使用者指南

Amazon Simple Storage Service

API 版本 2006-03-01

Amazon Simple Storage Service: 使用者指南

Table of Contents

什麼是 Amazon S3？

Amazon Simple Storage Service (Amazon S3) 是一項物件儲存服務，提供領先業界的可擴展性、資料可用性、安全性和效能。所有規模與產業的客戶都可使用 Amazon S3 來存放和保護適用於各種使用案例的任意資料量，例如資料湖、網站、行動應用程式、備份與還原、封存、企業應用程式、IoT 裝置和大數據分析。Amazon S3 提供管理功能，讓您可以最佳化、組織和設定對資料的存取，從而滿足您的特定業務、組織和合規要求。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱 什麼是 S3 Express One Zone？ 和 目錄值區。

主題

Amazon S3 的功能

儲存方案

Amazon S3 針對不同使用案例提供各式各樣的儲存類別。例如，您可以將任務關鍵型生產資料存放在 S3 Standard 或 S3 Express 單區域中以便經常存取；將不常存取的資料存放在 S3 Standard-IA 或 S3 One Zone-IA 中，以節省成本；以及以最低成本在 S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 中封存資料。

Amazon S3 Express One Zone 是一種高效能的單一區域 Amazon S3 儲存類別，專門為對延遲最敏感的應用程式提供一致的個位數毫秒資料存取。S3 Express One Zone 是目前可用的最低延遲雲端物件

儲存類別，資料存取速度提高了 10 倍，而且請求成本比 S3 標準低 50%。S3 Express One Zone 是第一款可讓您選取單一可用區域的 S3 儲存類別，還可選擇將物件儲存體與運算資源共置，藉此盡可能提供最高存取速度。此外，為了進一步提高存取速度並支援每秒數十萬個請求，資料會以新的儲存貯體類型儲存：Amazon S3 目錄儲存貯體。如需詳細資訊，請參閱 什麼是 S3 Express One Zone？ 及 目錄值區。

您可以在 S3 Intelligent-Tiering 中存放具有變更或未知存取模式的資料，而當您的存取模式變更時，會在四個存取層之間自動移動資料，從而最佳化儲存成本。這四個存取層包含兩個針對經常和不常存取最佳化的低延遲存取層，以及兩個針對極少存取的資料最佳化的非同步存取而設計的選用封存存取層。

如需詳細資訊，請參閱 使用 Amazon S3 儲存體方案。

儲存體管理

Amazon S3 具有儲存管理功能，可用於管理成本、滿足法規要求、降低延遲，並儲存多個不同的資料副本，以達到合規要求。

- S3 生命週期 - 設定生命週期組態，以便管理您的物件，並在其整個生命週期內以更符合成本效益的方式進行存放。您可以將物件轉換至其他 S3 儲存類別或使到達其生命週期結束日期的物件過期。
- S3 物件鎖定 – 在固定時間內或永遠避免刪除或覆寫 Amazon S3 物件。您可以使用「物件鎖定」來協助滿足需要 write-once-read-many(WORM) 儲存的法規要求，或者只是新增另一層防護來防止物件變更和刪除。
- S3 複寫 — 將物件及其各自的中繼資料和物件標籤複寫到相同或不 AWS 區域 同的一或多個目標儲存貯體，以減少延遲、合規、安全性和其他使用案例。
- S3 批次作業 – 透過單一 S3 API 請求或在 Amazon S3 主控台中按幾下滑鼠，即可大規模管理數十億個物件。您可以使用 Batch 作業對數百萬或數十億個物件執行複製、叫用 AWS Lambda 函數和還原等作業。

存取管理與安全性

Amazon S3 提供稽核和管理對儲存貯體和物件的存取的功能。根據預設，S3 儲存貯體與物件皆為私有。您僅可存取您建立的 S3 資源。若要授予可支援特定使用案例的精密資源使用權限，或稽核 Amazon S3 資源的許可，您可以使用下列功能。

- S3 封鎖公開存取 – 封鎖 S3 儲存貯體與物件的公開存取。根據預設，會在帳戶和儲存貯體層級開啟「封鎖公開存取」設定。建議您將所有「封鎖公開存取」設定保持啟用狀態，除非您知道需要針對特定使用案例關閉其中一或多個設定。如需詳細資訊，請參閱 為您的 S3 儲存貯體設定封鎖公開存取。

- AWS Identity and Access Management (IAM) — IAM 是一種 Web 服務，可協助您安全地控制資 AWS 源 (包括 Amazon S3 資源) 的存取。使用 IAM，您可以集中管理許可，以控制使用者可以存取的 AWS 資源。您可以使用 IAM 來控制能通過身分驗證 (登入) 和授權使用資源的 (具有許可) 的人員。
- 儲存貯體政策 – 使用以 IAM 為基礎的政策語言，為 S3 儲存貯體及其中的物件設定以資源為基礎的許可。
- Amazon S3 存取點— 使用專用存取政策設定名稱的網路端點，以大規模管理 Amazon S3 中的共用資料集的資料存取。
- 存取控制清單 (ACL) – 向授權的使用者授予讀取和寫入個別儲存貯體和物件的許可。一般而言，我們建議使用 S3 以資源為基礎的政策 (儲存貯體政策與存取點原則) 或 IAM 使用者政策來獲取存取控制而非 ACL。政策是一種簡化且更具彈性的存取控制選項。使用儲存貯體政策和存取點政策，您可以定義若干規則，廣泛應用於 Amazon S3 資源的所有請求。如需何時使用 ACL 而非資源型政策或 IAM 使用者政策之特定案例的詳細資訊，請參閱 使用 ACL 管理存取。
- S3 物件擁有權 - 取得儲存貯體中每個物件的擁有權，簡化存放在 Amazon S3 中資料的存取管理。S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來停用或啟用 ACL。根據預設，會停用 ACL。停用 ACL 後，儲存貯體擁有者會擁有儲存貯體中的所有物件，並使用存取管理政策專門管理對資料的存取。
- IAM Access Analyzer for S3 – 評估和監控您的 S3 儲存貯體存取政策，確保政策僅提供對 S3 資源的預期存取。

資料處理

若要轉換資料並觸發工作流程，以大規模自動化各種其他處理活動，則您可以使用下列功能。

- S3 Object Lambda – 將您自己的程式碼新增至 S3 GET、HEAD 和 LIST 請求，以在資料傳回至應用程式時對其進行修改和處理。執行資料列篩選、動態調整影像大小、修訂機密資料以及更多動作。
- 事件通知 — 觸發使用 Amazon Simple Notification Service (Amazon SNS)、Amazon Simple Queue Service (Amazon SQS) 的工作流程，以及對 S3 資源進行變更 AWS Lambda 時的工作流程。

儲存記錄和監控

Amazon S3 提供記錄和監控工具，您可以使用這些工具來監控和控制 Amazon S3 資源的使用方式。如需詳細資訊，請參閱監控工具。

自動化監控工具

- Amazon S3 的 Amazon CloudWatch 指標 — 追蹤 S3 資源的營運狀態，並在預估費用達到使用者定義閾值時設定帳單提醒。
- AWS CloudTrail— 記錄使用者、角色或 Amazon S3 AWS 服務 中所採取的動作。 CloudTrail 日誌可為您提供 S3 儲存貯體層級和物件層級作業的詳細 API 追蹤。

手動監控工具

- 伺服器存取記錄日誌 – 應儲存貯體要求，獲取的詳細記錄。您可以在許多使用案例中使用伺服器存取日誌，例如執行安全和存取稽核、了解您的客戶群以及掌握 Amazon S3 帳單。
- AWS Trusted Advisor — 使用 AWS 最佳實務檢查來評估您的帳戶，以識別最佳化 AWS 基礎架構、改善安全性和效能、降低成本，以及監控服務配額的方法。然後，您可以根據推薦來最佳化您的服務和資源。

分析與洞察

Amazon S3 提供相關功能，以協助您了解儲存用量，進而讓您能夠更好地了解、分析和大規模最佳化儲存。

- Amazon S3 Storage Lens – 了解、分析和最佳化儲存。S3 Storage Lens 提供 60 多個使用量和活動指標以及互動式儀表板，可彙總整個組織、特定帳戶 AWS 區域、儲存貯體或前置碼的資料。
- 儲存方案分析 – 分析儲存存取模式，以決定何時將資料移至更具成本效益的儲存類別。
- S3 清查與清查報告 – 稽核和報告物件及其對應的中繼資料，並設定其他 Amazon S3 功能以在清查報告中採取行動。例如，您可以報告物件的複寫和加密狀態。如需清查報告中每個物件可用的所有中繼資料清單，請參閱 Amazon S3 清查清單。

高度的一致性

Amazon S3 為您的 Amazon S3 儲存貯體中的物件放置和刪除請求提供了強大的 read-after-write 一致性 AWS 區域。這一行為適用於新物件的寫入，以及覆寫現有物件的 PUT 請求與 DELETE 請求。此外，Amazon S3 Select、Amazon S3 存取控制清單 (ACL)、Amazon S3 物件標籤和物件中繼資料 (例如 HEAD 物件) 上的讀取操作均高度一致。如需詳細資訊，請參閱 Amazon S3 資料一致性模式。

Amazon S3 的運作方式

Amazon S3 是將資料當做物件存放在儲存貯體中的物件儲存服務。物件是一個檔案和任何描述該檔案的中繼資料。儲存貯體是物件的容器。

要將資料存放在 Amazon S3 中，您應該先建立儲存貯體並指定儲存貯體名稱和 AWS 區域。然後，您可以將資料當做物件上傳到 Amazon S3 中的儲存貯體。每個物件都有金鑰 (或金鑰名稱），它是表示儲存貯體中物件的唯一識別符。

S3 提供您可設定的功能，以支援您的特定使用案例。例如，您可以使用 S3 版本控制將物件的多個版本保留在一個儲存貯體中，並可還原意外刪除或覆寫的物件。

儲存貯體和它們中的物件是私有的，只有在您明確授予存取許可時才能進行存取。您可以使用儲存貯體政策、 AWS Identity and Access Management (IAM) 政策、存取控制清單 (ACL) 和 S3 存取點來管理存取。

主題

- 儲存貯體
- 物件
- 鍵
- S3 版本控制
- 版本 ID
- 儲存貯體政策
- S3 存取點
- 存取控制清單 (ACL)
- 區域

儲存貯體

儲存貯體是 Amazon S3 中用於存放物件的容器。您可以在儲存貯體中存放任意數目的物件，並且帳戶中最多可有 100 個儲存貯體。若要請求增加，請造訪 Service Quotas 主控台。

每個物件都包含在儲存貯體中。例如，如果名為 `photos/puppy.jpg` 的物件存放在美國西部 (奧勒岡) 區域的 `DOC-EXAMPLE-BUCKET` 儲存貯體中，則可以使用 URL `https://DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com/photos/puppy.jpg` 定址。如需詳細資訊，請參閱存取儲存貯體。

建立儲存貯體時，請輸入儲存貯體名稱並選擇儲存貯體將駐留的 AWS 區域 。建立儲存貯體後，便無法變更儲存貯體的名稱或其區域。儲存貯體名稱必須遵循儲存貯體命名規則。您也可以將儲存貯體設定為使用 S3 版本控制或其他儲存管理功能。

儲存貯體還可：

- 以最高層級來組織 Amazon S3 命名空間。
- 識別負責儲存和數據傳輸費的帳戶。
- 提供存取控制選項，例如儲存貯體政策、存取控制清單 (ACL) 和 S3 存取點，且您可將其用來管理 Amazon S3 資源的存取。
- 充當用量報告中的彙總單位。

如需儲存貯體的詳細資訊，請參閱「儲存貯體概觀」。

物件

物件是存放在 Amazon S3 中的基本實體。物件是由物件資料與中繼資料構成。中繼資料是一組成對的名稱與數值，會說明該物件。其中包含一些預設中繼資料 (如上次修改日期) 以及標準 HTTP 中繼資料 (如 `Content-Type`)。您也可以在存放物件時指定自訂中繼資料。

在儲存貯體中，每個物件都是由金鑰 (名稱) 與版本 ID (如果已在儲存貯體啟用 S3 版本控制) 來唯一識別。如需物件的詳細資訊，請參閱 Amazon S3 物件概觀。

鍵

物件金鑰 (或金鑰名稱) 是儲存貯體內的物件的唯一識別符。儲存貯體中的每個物件只能有一個索引鍵。儲存貯體、物件金鑰以及選擇性的版本 ID (如果已針對儲存貯體啟用 S3 版本控制) 的組合可唯一識別每個物件。因此，您可以將 Amazon S3 視為「儲存貯體 + 金鑰 + 版本」與物件本身之間的基本資料對應。

Amazon S3 中的每個物件可透過 Web 服務端點、儲存貯體名稱、金鑰及版本 (選擇性) 的組合來唯一定址。例如，在 URL `https://DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com/photos/puppy.jpg` 中，`DOC-EXAMPLE-BUCKET` 是儲存貯體的名稱，而 `photos/puppy.jpg` 是金鑰。

如需物件金鑰的詳細資訊，請參閱「建立物件索引鍵名稱」。

S3 版本控制

您可以使用 S3 版本控制，以在相同的儲存貯體中保留物件的多個變形。使用 S3 版本控制功能，您即可保留、擷取和還原在儲存貯體中所存放每個物件的各個版本。您就可以輕鬆地復原失誤的使用者動作和故障的應用程式。

如需詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。

版本 ID

當您在儲存貯體中啟用 S3 版本控制時，Amazon S3 會為所有新增至儲存貯體的物件提供唯一的版本 ID。啟用版本控制時已存在於儲存貯體中的物件的版本 ID 為 `null`。如果您使用其他操作（例如和）修改這些（或任何其他）對象 CopyObjectPutObject，則新對象將獲得唯一的版本 ID。

如需詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。

儲存貯體政策

值區政策是以資源為基礎的 AWS Identity and Access Management (IAM) 政策，可用來授與值區及其中物件的存取權限。只有儲存貯體擁有者可建立政策與儲存貯體的關聯。連接到儲存貯體的許可會套用至儲存貯體擁有者帳戶擁有的所有儲存貯體物件。儲存貯體政策的大小限制為 20 KB。

儲存貯體政策使用 AWS標準的以 JSON 為基礎的存取政策語言。您可以使用儲存貯體政策來新增或拒絕儲存貯體中物件的許可。儲存貯體政策會根據政策中的元素來允許或拒絕要求，包括請求的請求者、S3 動作、資源以及其他方面或條件 (例如，用來傳送要求的 IP 地址)。例如，您可以建立儲存貯體政策，授予跨帳戶許可，以將物件上傳至 S3 儲存貯體，同時確保儲存貯體擁有者可完全控制上傳物件。如需詳細資訊，請參閱 Amazon S3 儲存貯體政策範例。

在儲存貯體政策中，您可以在 Amazon Resource Name (ARN) 上使用萬用字元和其他值上使用許可授予物件子集。例如，您可以控制對以常用字首或以給定的擴展名結束，例如`.html`。

S3 存取點

Amazon S3 存取點是含有專用存取政策的命名網路端點，其中說明了如何使用該端點存取資料。存取點會附加至儲存貯體，您可以用來執行 S3 物件操作，例如 GetObject 和PutObject。存取點針對 Amazon S3 中的共用資料集，簡化了大規模的資料存取管理。

每個存取點都有其自己的存取點政策。您也可以為每個存取點設定封鎖公開存取設定。若要限制只能透過私有網絡存取 Amazon S3 資料，您可以將任何存取點設定為僅接受來自 Virtual Private Cloud (VPC) 的請求。

如需詳細資訊，請參閱 使用 Amazon S3 存取點管理資料存取。

存取控制清單 (ACL)

您可以向授權的使用者授予讀取和寫入個別儲存貯體和物件的許可。每個儲存貯體與物件都會有一個與其連接的 ACL 作為子資源。ACL 會定義哪些 AWS 帳戶 或群組被授與存取權限，以及存取權的類型。ACL 是一種早於 IAM 的存取控制機制。如需 ACL 的詳細資訊，請參閱「存取控制清單 (ACL) 概觀」。

S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來同時控制上傳至儲存貯體之物件的擁有權，以及停用或啟用 ACL。根據預設，物件擁有權設定為「儲存貯體擁有者強制執行」設定，而且所有 ACL 都會停用。停用 ACL 時，儲存貯體擁有者會擁有儲存貯體中的所有物件，並使用存取管理政策專門管理對這些物件的存取。

Amazon S3 中的大多數新式使用案例不再需要使用 ACL。建議您將 ACL 保持停用狀態，除非在異常情況下必須個別控制每個物件的存取。停用 ACL 後，您可以使用政策來控制對儲存貯體中所有物件的存取，無論是誰將物件上傳到您的儲存貯體。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

區域

您可以選擇 Amazon S3 存放您建立的儲存貯體的地理 AWS 區域 位置。您可以選擇區域以最佳化延遲、降低成本或因應法規需求。存儲在一個對象 AWS 區域 永遠不會離開該區域，除非您明確地將它們轉移或複製到另一個區域。例如，存放在歐洲 (愛爾蘭) 區域的物件絕不會離開此區域。

> **Note**
>
> 您只能在已為您的帳戶啟用的功 AWS 區域 能中存取 Amazon S3 及其功能。如需有關啟用區域來建立和管理 AWS資源的詳細資訊，請參閱 AWS 區域中的管理AWS 一般參考。

如需 Amazon S3 區域與端點的清單，請參閱《AWS 一般參考》中的區域與端點。

Amazon S3 資料一致性模式

Amazon S3 為您的 Amazon S3 儲存貯體中的物件放置和刪除請求提供了強大的 read-after-write 一致性 AWS 區域。這一行為適用於寫入新的物件，以及覆寫現有物件的 PUT 請求與 DELETE 請求。此外，Amazon S3 Select、Amazon S3 存取控制清單 (ACL)、Amazon S3 物件標籤和物件中繼資料 (例如 HEAD 物件) 上的讀取操作均高度一致。

單一金鑰的更新不可部分完成。例如，如果您從一個執行緒中向現有的金鑰傳送 PUT 請求，並同時從第二個執行緒在相同的金鑰上執行 GET 請求，則會獲得舊資料或新資料，但絕不會獲得部分或損壞的資料。

Amazon S3 在 AWS 資料中心內的多部伺服器上複寫資料，達到高可用性。如果 PUT 要求成功，則您的資料已安全地存放。收到一個成功的 PUT 回應後啟動的任何讀取（GET 或 LIST）將傳回由 PUT 請求寫入的資料。下列是此行為的範例：

- 一個處理序將新物件寫入 Amazon S3，並立即列出其儲存貯體內的金鑰。新物件將出現在清單中。
- 一個程序會取代現有的物件，並立即嘗試讀取該物件。Amazon S3 會傳回新的資料。
- 一個程序會刪除現有的物件，並立即嘗試讀取該物件。Amazon S3 不會傳回任何資料，因為物件已被刪除。
- 一個程序會刪除現有的物件，並立即列出其儲存貯體內的金鑰。物件將不會出現在清單中。

> **Note**
>
> - Amazon S3 不支援並行寫入器的物件鎖定。如果同時對相同的金鑰提出兩個 PUT 要求，會優先使用具有最新時間戳記的要求。如果這是問題，您必須在應用程式中內建物件鎖定機制。
> - 更新是以金鑰為基礎。無法跨金鑰進行不可部分完成的更新。例如，您無法更新某個需要更新另一個金鑰的金鑰，除非您將這項功能設計到應用程式中。

儲存貯體組態具有最終一致性模式。具體來說，這表示：

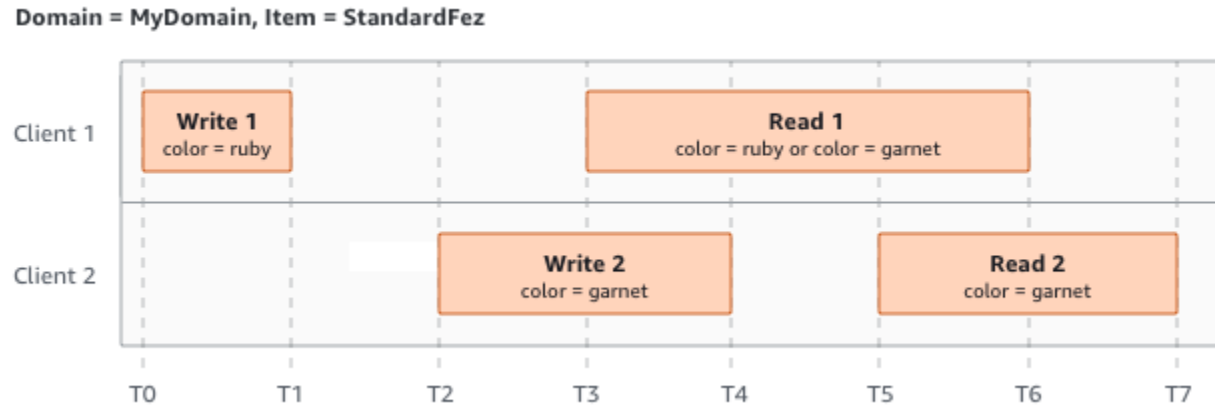
- 如果您刪除儲存貯體並立即列出所有儲存貯體，刪除的儲存貯體可能仍會出現在清單中。
- 如果您第一次在儲存貯體上啟用版本控制，則可能需要很短的時間才能完全傳播變更。我們建議您在啟用版本控制之後等待 15 分鐘，然後再對儲存貯體中的物件發出寫入作業 (PUT 或 DELETE 請求)。

並行應用程式

本節提供了一些範例，說明在多個用戶端寫入相同項目時，Amazon S3 可預期的行為。

在此範例中，會完成 W1 (寫入 1) 與 W2 (寫入 2)，再開始 R1 (讀取 1) 與 R2 (讀取 2)。由於 S3 高度一致，所以 R1 和 R2 均會傳回 `color = ruby`。

在下一個範例中，W2 會在開始 R1 之後才完成。因此，R1 可能會傳回 `color = ruby` 或 `color = garnet`。但是，因為 W1 和 W2 在 R2 開始之前完成，所以 R2 會傳回 `color = garnet`。



在最後一個範例中，W2 在 W1 收到確認之前即開始。因此，這些寫入會視為並行動作。Amazon S3 內部使用last-writer-wins 語意來判斷哪些寫入優先順序。然而，因為網路延遲等因素，Amazon S3 接收請求的順序和應用程式接收確認的順序則無法預測。例如，W2 可能由同一區域中的 Amazon EC2 執行個體啟動，而 W1 可能由較遠的主機啟動。確定最終值的最佳方法是在確認兩個寫入之後執行讀取。

相關服務

將資料載入 Amazon S3 之後，您可以將其與其他 AWS 服務搭配使用。以下為您可能最常使用的服務：

- Amazon Elastic Compute Cloud (Amazon EC2) – 在 AWS 雲端中提供安全且可擴展的運算容量。使用 Amazon EC2 可減少前期所需的硬體投資，讓您更快速開發並部署應用程式。您可使用 Amazon EC2 按需要啟動任意數量的虛擬伺服器，設定安全性和聯網功能以及管理儲存。
- Amazon EMR – 幫助企業、研究員、資料分析師與開發人員，輕鬆地以符合成本效益的方式來處理相當大量的資料。Amazon EMR 採用 Web 規模的 Amazon EC2 與 Amazon S3 基礎設施上運作的託管 Hadoop 框架。
- AWS S@@ now Family — 協助需要在嚴峻、非資料中心環境以及缺乏一致網路連線能力的地點執行作業的客戶。您可以使用 S AWS now Family 裝置， AWS 雲端 在可能無法選擇網際網路連線的地方，以符合成本效益的方式存取儲存空間和運算能力。
- AWS Transfer Family – 使用 Secure Shell (SSH) 檔案傳輸通訊協定 (SFTP)、透過 SSL 檔案傳輸通訊協定 (FTPS) 和檔案傳輸通訊協定 (FTP)，為 Amazon S3 或 Amazon Elastic File System (Amazon EFS) 直接傳輸檔案提供全受管支援。

存取 Amazon S3

您可以透過以下任何方式來使用 Amazon S3：

AWS Management Console

主控台是用於管理 Amazon S3 和 AWS 資源的網頁式使用者界面。如果您已註冊 AWS 帳戶，則可以登入 AWS Management Console 並從 AWS Management Console 首頁選擇 S3 來存取 Amazon S3 主控台。

AWS Command Line Interface

您可以使用命 AWS 令列工具在系統的命令列上發出命令或建置指令碼，以執行 AWS (包括 S3) 工作。

AWS Command Line Interface (AWS CLI) 為廣泛的集合提供指令 AWS 服務。在視窗、macOS 和 Linux 上支援。 AWS CLI 若要開始使用，請參閱《AWS Command Line Interface 使用者指南》。如需 Amazon S3 命令的詳細資訊，請參閱 AWS CLI 命令參考中的 s3api 和 s3control。

AWS 開發套件

AWS 提供包含各種程式設計語言和平台 (Java、Python、Ruby、.NET、iOS、安卓等) 的程式庫和範例程式碼的 SDK (軟體開發套件)。開 AWS 發套件提供方便的方式來建立 S3 和. AWS Amazon S3 是一個 REST 服務。您可以使用開發 AWS 套件程式庫將請求傳送到 Amazon S3，這些程式庫會包裝基礎 Amazon S3 REST API 並簡化您的程式設計任務。例如，開發套件會負責的工作諸如計算簽章、以密碼演算法簽署請求、管理錯誤以及自動重試請求。如需 AWS SDK 的相關資訊，包括如何下載和安裝它們，請參閱. AWS

與 Amazon S3 的每次互動，可以經過驗證身分或是匿名進行。如果您使用的是 AWS SDK，程式庫會從您提供的金鑰計算驗證的簽章。如需有關如何向 Amazon S3 提出請求的詳細資訊，請參閱 提出要求。

Amazon S3 REST API

Amazon S3 的架構設計成非程式設計語言相關，並使用 AWS支援的界面來存放與擷取物件。您可以透過使用 Amazon S3 REST API 以程式化設計方式存取 S3 和 AWS 。REST API 是 Amazon S3 的 HTTP 界面。藉助 REST API，您可以使用標準 HTTP 要求來建立、擷取與刪除儲存貯體與物件。

若要使用 REST API，您可以使用支援 HTTP 的任何工具組。您甚至可以使用瀏覽器來擷取物件，只要物件是可匿名讀取即可。

REST API 使用標準 HTTP 標頭與狀態碼，因此標準瀏覽器與工具組會如預期般運作。在某些區域中，我們已新增功能至 HTTP (例如，我們已新增標頭來支援存取控制)。在此情況下，我們會盡力以符合標準 HTTP 使用風格的方式來新增功能。

如果直接在應用程式中呼叫 REST API，您必須撰寫程式碼來運算簽章，並將其新增至要求。如需有關如何向 Amazon S3 提出請求的詳細資訊，請參閱 提出要求。

> **Note**
>
> HTTP 上的 SOAP API 支援已淘汰，但仍可透過 HTTPS 取得。SOAP 不支援較新的 Amazon S3 功能。我們建議您使用其他 API 或 AWS 開發套件。

支付 Amazon S3

Amazon S3 定價旨在讓您不需要規劃應用程式的儲存體需求。大部分的儲存提供者會要求您購買預先決定的儲存空間和網路傳輸容量。在此情況中，如果您超過容量，您的服務會關閉或必須支付很高的超額費用。如果未超過該容量，您仍會支付全額使用費用。

Amazon S3 只會按實際用量收費，不會有任何隱藏費用與超額費用。此模型提供可變成本的服務，可隨著您的業務成長，同時為您提供基礎架構的成本優勢。 AWS 如需詳細資訊，請參閱 Amazon S3 定價。

當您註冊時 AWS，系統會自動註冊中的所有服務 AWS，包括 Amazon S3。 AWS 帳戶 不過，您只需針對所使用的服務付費。若您是 Amazon S3 新客戶，可以免費試用 Amazon S3。如需詳細資訊，請參閱 AWS 免費方案。

若要查看您的帳單，請前往 AWS Billing and Cost Management 主控台中的帳單與成本管理儀表板。若要進一步了解 AWS 帳戶 帳單，請參閱使AWS Billing 用者指南。如果您對 AWS 帳單有任何疑問 AWS 帳戶，請聯絡 Sup AWS port 部門。

PCI DSS 合規

Amazon S3 支援處理、儲存、傳輸商家或服務供應商的信用卡資料，並且已驗證符合支付卡產業 (PCI) 資料安全標準 (DSS)。如需 PCI DSS 的詳細資訊，包括如何要求 AWS PCI 相容性 Package 的複本，請參閱 PCI DSS 等級 1。

Amazon S3 入門

您可以透過使用儲存貯體和物件來開始使用 Amazon S3。儲存貯體是物件的容器。物件是一個檔案和任何描述該檔案的中繼資料。

若要將物件存放在 Amazon S3 中，您需要建立儲存貯體，然後將物件上傳到儲存貯體。當物件在儲存貯體中時，您可以開啟、下載和移動它。當您不再需要物件或儲存貯體時，可以清理這些資源。

使用 Amazon S3，您只需按實際用量付費。如需 Amazon S3 功能和定價的詳細資訊，請參閱 Amazon S3。若您是 Amazon S3 新客戶，可以免費試用 Amazon S3。如需詳細資訊，請參閱 AWS 免費方案。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱 什麼是 S3 Express One Zone？ 和 目錄值區。

影片：Amazon S3 入門

必要條件

開始之前，請務必先完成「必要條件：設定 Amazon S3」中的步驟。

主題

必要條件：設定 Amazon S3

當您註冊時 AWS，系統會自動註冊中的所有服務 AWS，包括 Amazon S3。 AWS 帳戶 您只需支付實際使用服務的費用。

使用 Amazon S3，您只需按實際用量付費。如需 Amazon S3 功能和定價的詳細資訊，請參閱 Amazon S3。若您是 Amazon S3 新客戶，可以免費試用 Amazon S3。如需詳細資訊，請參閱 AWS 免費方案。

若要設定 Amazon S3，請依照以下各節中的步驟進行。

當您註冊 AWS 並設定 Amazon S3 時，您可以選擇性地變更中的顯示語言 AWS Management Console。如需詳細資訊，請參閱《AWS Management Console 入門指南》中的變更 AWS Management Console的語言。

主題

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時登錄 https://aws.amazon.com/ 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 選擇 Root 使用者並輸入您的 AWS 帳戶 電子郵件地址，以帳戶擁有者身分登入。AWS Management Console在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的以根使用者身分登入。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置 (主控台)。

建立具有管理權限的使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM 身分中心中，將管理存取權授予使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。

以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

  如需使用 IAM 身分中心使用者登入的說明，請參閱使用AWS 登入 者指南中的登入 AWS 存取入口網站。

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「建立權限集」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「新增群組」。

步驟 1：建立您的第一個 S3 儲存貯體

註冊之後 AWS，您就可 Amazon S3 用 AWS Management Console. Amazon S3 中的每個物件都會存放在儲存貯體中。您必須先建立儲存貯體，才能將資料存放至 Amazon S3。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱什麼是 S3 Express One Zone？ 和 目錄值區。

> **Note**
>
> 不會向您收取儲存貯體建立費用。只會向您收取在儲存貯體中存放物件以及物件進出儲存貯體的費用。遵循本指南中範例而產生的費用極小 (低於 1 USD)。如需儲存成本的詳細資訊，請參閱 Amazon S3 定價。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在頁面頂端的導覽列中，選擇目前顯示的名稱 AWS 區域。接下來，選擇您要在其中建立值區的「區域」。

   > **Note**
   >
   > 請選擇接近您的區域，以充分降低延遲及成本，並因應法規要求。除非您明確地將存放在區域中的物件傳輸到其他區域，否則物件絕對不會離開該區域。如需 Amazon S3 的清單 AWS 區域，AWS 服務 請參閱 Amazon Web Services 一般參考.

3. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
4. 選擇 Create bucket (建立儲存貯體)。

   Create bucket (建立儲存貯體) 頁面隨即開啟。
5. 在 [一般設定] 下方，檢視 AWS 區域 儲存貯體的建立位置。
6. 在「鏟斗類型」下選擇「一般用途」。
7. 針對 Bucket name (儲存貯體名稱)，輸入儲存貯體的名稱。

儲存貯體名稱必須；

- 在分割區內不重複。分割區是區域的群組。 AWS 目前有三個分割區：aws(標準區域)、aws-cn(中國區域) 和 aws-us-gov (AWS GovCloud (US) Regions)。
- 長度必須介於 3 與 63 個字元之間。
- 只能由小寫字母、數字、句點 (.) 和連字號 (-) 組成。為了獲得最佳相容性，建議您避免在儲存貯體名稱中使用句點 (.)，但僅用於靜態網站託管的儲存貯體除外。
- 開頭和結尾為字母或數字。

建立儲存貯體後，便無法變更其名稱。如需儲存貯體命名的詳細資訊，請參閱 [儲存貯體命名規則]。

> **Important**
>
> 避免在儲存貯體名稱中包含敏感資訊，例如帳戶號碼。在指向儲存貯體中之物件的 URL 中，會顯示儲存貯體名稱。

8. AWS Management Console 可讓您將現有值區的設定複製到新值區。如果您不想複製現有值區的設定，請跳至下一個步驟。

> **Note**
>
> 此選項：
>
> - 無法在中使用， AWS CLI 且只能在主控台中使用
> - 不適用於目錄值區
> - 不會將儲存貯體政策從現有值區複製到新值區

若要複製現有值區的設定，請在 [從現有值區複製設定] 底下，選取 [選擇值區]。「選擇時段」視窗即會開啟。找出包含您要複製之設定的值區，然後選取 [選擇值區]。「選擇時段」視窗即會關閉，且「建立時段」視窗會重新開啟。

在「從現有值區複製設定」 下方，您現在會看到所選值區的名稱。您也會看到 [還原預設值] 選項，可用來移除複製的值區設定。檢閱「建立值區」頁面上的剩餘值區設定。您將看到它們現在與您選擇的存儲桶的設置匹配。您可以跳到最後一步。

9. 在 Object Ownership (物件擁有權) 下，若要停用或啟用 ACL 並控制上傳在儲存貯體中物件的擁有權，請選擇下列其中一個設定：

   已停用 ACL

   - 儲存貯體擁有者強制執行 (預設) - 停用 ACL，儲存貯體擁有者會自動擁有並完全控制儲存貯體中的每個物件。ACL 不再影響 S3 儲存貯體中資料的存取許可。儲存貯體單獨使用政策來定義存取控制。

     根據預設，會停用 ACL。Amazon S3 中的大多數新式使用案例不再需要使用 ACL。建議您將 ACL 保時停用狀態，除非在異常情況下必須個別控制每個物件的存取。如需詳細資訊，請參閱控制物件的擁有權並停用儲存貯體的 ACL。

   已啟用 ACL

   - 儲存貯體擁有者偏好 – 儲存貯體擁有者擁有並完全控制其他帳戶使用 `bucket-owner-full-control` 標準 ACL 寫入儲存貯體的新物件。

     如果您套用儲存貯體擁有者偏好設定，以要求所有 Amazon S3 上傳都包含 `bucket-owner-full-control` 固定 ACL 時，您可以新增儲存貯體政策，只允許使用此 ACL 的物件上傳。
   - 物件寫入器 — 上傳物件的物件擁有物件、擁有物件的完全控制權，並且可以透過 ACL 授與其他使用者存取 AWS 帳戶 該物件。

   > **Note**
   >
   > 預設設定為儲存貯體擁有者強制執行。若要套用預設設定並將 ACL 保持停用狀態，只需要 `s3:CreateBucket` 許可。若要啟用 ACL，您必須具有 `s3:PutBucketOwnershipControls` 許可。

10. 在封鎖此儲存貯體的公開存取設定之下，選擇要套用至儲存貯體的封鎖公開存取設定。

    根據預設，會啟用全部四個「封鎖公開存取」設定。建議您將所有設定保持啟用狀態，除非您知道需要針對特定使用案例關閉其中一或多個設定。如需封鎖公開存取的詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

> **Note**
>
> 若要啟用所有「封鎖公用存取」設定，只需要 `s3:CreateBucket` 許可。若要關閉任何「封鎖公開存取」設定，您必須具有 `s3:PutBucketPublicAccessBlock` 許可。

11. (選用) 在 Bucket Versioning (儲存貯體版本控制) 下，您可以選擇是否要在儲存貯體中保留物件的變體。如需版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。

    若要對儲存貯體停用或啟用版本控制，請選擇 Disable (停用) 或 Enable (啟用)。

12. (選用) 在 Tags (標籤) 下，您可以選擇新增標籤至儲存貯體。標籤是用來分類儲存的鍵值對。

    若要新增儲存貯體標籤，請輸入 Key (金鑰) 並選擇性地輸入 Value (值)，然後選擇 Add tag (新增標籤)。

13. 在 Default encryption (預設加密) 底下，選擇 Edit (編輯)。

14. 若要設定預設加密，請在加密類型下，選擇下列其中一項：

    - Amazon S3 受管金鑰 (SSE-S3)
    - AWS Key Management Service 金鑰 (SSE-公里)

    > **Important**
    >
    > 如果您針對預設加密組態使用 SSE-KMS 選項，則受到 AWS KMS的每秒請求數目 (RPS) 限制。如需有關 AWS KMS 配額以及如何要求提高配額的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的配額。

    儲存貯體和新物件會以 Amazon S3 受管金鑰做為基本加密組態層級，使用伺服器端加密。如需預設加密的詳細資訊，請參閱對 Amazon S3 儲存貯體設定預設伺服器端加密行為。

    如需有關使用 Amazon S3 伺服器端加密來加密資料的詳細資訊，請參閱「使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密」。

15. 若您選擇 AWS Key Management Service 金鑰 (SSE-KMS)，請執行下列操作：

    a. 在 AWS KMS 金鑰下，使用下列其中一種方式指定 KMS 金鑰：

       - 若要從可用 KMS 金鑰清單中選擇，請選擇從您的金鑰中選擇 AWS KMS keys，然後從可用金鑰清單中選擇您的 KMS 金鑰。

AWS 受管金鑰 (aws/s3) 和您的客戶管理金鑰都會出現在此清單中。如需詳細了解客戶受管金鑰，請參閱《AWS Key Management Service 開發人員指南》中的客戶金鑰和 AWS 金鑰。

- 若要輸入 KMS 金鑰 ARN，請選擇輸入 AWS KMS key ARN，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。
- 若要在 AWS KMS 主控台中建立新的客戶管理金鑰，請選擇 [建立 KMS 金鑰]。

  如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。

> **Important**
>
> 您只能使用與值區相同 AWS 區域 的 KMS 金鑰。Amazon S3 主控台僅會列出與儲存貯體位於相同區域的前 100 個 KMS 金鑰。若要使用未列出的 KMS 金鑰，必須輸入 KMS 金鑰 ARN。若您想要使用其他帳戶的 KMS 金鑰，您必須先具有該金鑰的使用權限，然後輸入 KMS 金鑰 ARN。如需詳細了解 KMS 金鑰跨帳戶權限，請參閱《AWS Key Management Service 開發人員指南》中的建立其他帳戶可使用的 KMS 金鑰。如需 SSE-KMS 的詳細資訊，請參閱 使用 AWS KMS (SSE-KMS) 指定伺服器端加密。
>
> 當您在 Amazon S3 中使 AWS KMS key 用伺服器端加密時，您必須選擇對稱加密 KMS 金鑰。Amazon S3 僅支援對稱加密 KMS 金鑰，而不支援非對稱 KMS 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的識別對稱和非對稱 KMS 金鑰。

如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。如需 AWS KMS 搭配 Amazon S3 搭配使用的詳細資訊，請參閱使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)。

b. 當您將儲存貯體設定為使用 SSE-KMS 的預設加密時，您還可以啟用 S3 儲存貯體金鑰。S3 儲存貯體金鑰透過將 Amazon S3 的請求流量減少到，從而降低加密成本 AWS KMS。如需詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本。

   若要使用 S3 儲存貯體金鑰，在 Bucket Key (儲存貯體金鑰) 下選擇 Enable (啟用)。

16. (選用) 如果您想要啟用 S3 物件鎖定，請執行下列動作：

    a. 選擇 Advanced settings (進階設定)。

> **Important**
>
> 啟用物件鎖定也會啟用儲存貯體的版本控制。啟用後，您必須設定「物件鎖定」預設保留和合法保留設定，以防止新物件遭到刪除或覆寫。

b. 如果想要啟用物件鎖定，請選擇 Enable (啟用)、讀取出現的警告並確認。

如需詳細資訊，請參閱 使用 S3 物件鎖定。

> **Note**
>
> 若要建立已啟用物件鎖定的儲存貯體，您必須具備下列許可：`s3:CreateBucket`、`s3:PutBucketVersioning` 和 `s3:PutBucketObjectLockConfiguration`。

17. 選擇 Create bucket (建立儲存貯體)。

您已在 Amazon S3 中建立儲存貯體。

下一步驟

若要將物件新增至儲存貯體，請參閱「步驟 2：將物件上傳至您的儲存貯體」。

步驟 2：將物件上傳至您的儲存貯體

在 Amazon S3 中建立儲存貯體後，即可將物件上傳至儲存貯體。物件可以是任何類型的檔案：文字檔、相片、影片等等。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱什麼是 S3 Express One Zone？ 和 目錄值區。

將物件上傳至儲存貯體

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。

2. 在儲存貯體清單中，選擇您要上傳物件的目標儲存貯體名稱。
3. 在儲存貯體的物件索引標籤上，選擇上傳。
4. 在檔案和資料夾下，選擇新增檔案。
5. 選擇要上傳的檔案，然後選擇 Open (開啟)。
6. 選擇上傳。

您已經成功將物件上傳至您的儲存貯體中。

下一步驟

若要檢視您的物件，請參閱 步驟 3：下載物件。

步驟 3：下載物件

上傳物件到儲存貯體後，您可以檢視物件的相關資訊，並將物件下載至您的本機電腦。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱 什麼是 S3 Express One Zone？ 和 目錄值區。

使用 S3 主控台

本節說明如何使用 Amazon S3 主控台從 S3 儲存貯體下載物件。

> **Note**
>
> - 您一次只能下載一個物件。
> - 如果您使用 Amazon S3 主控台下載的物件，且其金鑰名稱結尾為句號 (.)，則會移除所下載物件的金鑰名稱中的句號。若要保留所下載物件名稱結尾的句號，您必須使用 AWS Command Line Interface (AWS CLI)、AWS SDK 或 Amazon S3 REST API。

從 S3 儲存貯體下載物件

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在 Buckets (儲存貯體) 清單中，選擇您要從中下載物件的儲存貯體名稱。
3. 您可使用下列任一方式從 S3 儲存貯體下載物件：
   - 勾選物件旁的核取方塊，然後選擇下載。如果您要將物件下載到特定資料夾，請在動作選單上選擇下載為。
   - 如果您要下載特定版本的物件，請開啟顯示版本 (位於搜尋方塊旁)。勾選您要的物件版本旁的核取方塊，然後選擇下載。如果您要將物件下載到特定資料夾，請在動作選單上選擇下載為。

您已成功下載您的物件。

下一步驟

若要在 Amazon S3 中複製和貼上物件，請參閱「步驟 4：將物件複製到資料夾」。

步驟 4：將物件複製到資料夾

您已將物件新增至儲存貯體，並已下載了物件。現在，您將建立資料夾並複製物件，並將其貼到資料夾中。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱什麼是 S3 Express One Zone？ 和 目錄值區。

將物件複製到資料夾

1. 在 Buckets (儲存貯體) 清單中，選擇您的儲存貯體名稱。
2. 選擇 Create folder (建立資料夾) 並設定新資料夾：
   a. 輸入資料夾名稱 (例如 `favorite-pics`)。
   b. 如為資料夾加密設定，請選擇 Disable (停用)。

c. 選擇儲存。

3. 導覽至 Amazon S3 儲存貯體或資料夾，其中包含您要複製的物件。
4. 選取物件名稱左側的核取方塊，以複製這些物件。
5. 選擇 Actions (動作)，然後從出現的選項清單中選擇 Copy (複製)。

   或者，從右上角的選項中選擇 Copy (複製)。
6. 選擇目的地資料夾：

   a. 選擇 Browse S3 (瀏覽 S3)。

   b. 選擇資料夾名稱左側的選項按鈕。

      若要導覽至資料夾並選擇子資料夾做為目的地，請選擇資料夾名稱。

   c. 選擇 Choose destination (選擇目的地)。

   目的地資料夾的路徑會出現在 Destination (目的地) 方塊中。在 Destination (目的地) 中，您可以間隔地輸入目的地路徑，例如 s3://*bucket-name*/*folder-name*/。
7. 選擇右下角的 Copy (複製)。

   Amazon S3 會將您的物件複製到目的地資料夾。

下一步驟

若要刪除 Amazon S3 中的物件和儲存貯體，請參閱「步驟 5：刪除物件和儲存貯體」。

步驟 5：刪除物件和儲存貯體

當您不再需要物件或儲存貯體時，我們建議您將其刪除，以免繼續產生費用。如果您已完成此入門演練作為學習練習，但不打算使用儲存貯體或物件，建議您將其刪除，以免繼續產生費用。

刪除儲存貯體之前，必須清空儲存貯體或刪除儲存貯體中的物件。刪除物件和儲存貯體後，就無法再使用這些物件。

如果您想要繼續使用相同的儲存貯體名稱，建議您刪除物件或清空儲存貯體，但不要刪除該儲存貯體。刪除儲存貯體之後，該名稱就可以重複使用。但是，在您有機會重複使用之前，另一個 AWS 帳戶 可能會建立同名的儲存貯體。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱什麼是 S3 Express One Zone？ 和 目錄值區。

主題

刪除物件

如果您想要選擇刪除的物件，而不清空儲存貯體中的所有物件，則可以刪除物件。

1. 在 Bucket (儲存貯體) 清單中，選擇您要從中刪除物件的儲存貯體名稱。
2. 選取您要刪除的物件。
3. 從右上角的選項中選擇刪除。
4. 在刪除物件頁面上，輸入 **`delete`** 以確認刪除物件。
5. 選擇 Delete objects (刪除物件)。

將儲存貯體清空

如果您打算刪除儲存貯體，則必須首先將其清空，從而刪除其中的所有物件。

清空儲存貯體

1. 在 Bucket (儲存貯體) 清單中，選取要清空的儲存貯體，然後選擇 Empty (清空)。
2. 若要確認您想清空儲存貯體並刪除其中的所有物件，請在 Empty bucket (清空儲存貯體) 中輸入 **`permanently delete`**。

> **Important**
>
> 清空儲存貯體無法復原。清空儲存貯體動作正在進行時在儲存貯體中新增的物件將會遭到刪除。

3. 若要清空儲存貯體並刪除其中的所有物件，請選擇 Empty (清空)。

   Empty bucket: Status (清空儲存貯體：狀態) 頁面隨即開啟，您可以使用此頁面來檢閱失敗和成功物件刪除的摘要。

4. 若要返回儲存貯體清單，請選擇 Exit (結束)。

刪除儲存貯體

清空儲存貯體或刪除儲存貯體中的所有物件後，您就可以刪除儲存貯體。

1. 若要刪除儲存貯體，請在 Buckets (儲存貯體) 清單中進行選取。
2. 選擇 Delete (刪除)。
3. 若要確認刪除，請在 Delete bucket (刪除儲存貯體) 中輸入儲存貯體的名稱。

   Important

   刪除儲存貯體無法復原。儲存貯體名稱是唯一的。如果您刪除儲存貯體，其他 AWS 使用者就可以使用該名稱。若希望繼續使用相同的儲存貯體名稱，請勿刪除該儲存貯體。反之，請清空並保留儲存貯體。

4. 若要刪除儲存貯體，請選擇 Delete bucket (刪除儲存貯體)。

後續步驟

在前面各範例中，您已了解如何執行一些基本 Amazon S3 任務。

下列各主題說明您可用於更深入了解 Amazon S3 以在應用程式中進行實作的學習路徑。

Note

如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱什麼是 S3 Express One Zone？ 和 目錄值區。

主題

- 了解常用使用案例

- 控制對儲存貯體與物件的存取
- 管理和監控您的儲存
- 使用 Amazon S3 進行開發
- 從教學課程學習
- 探索培訓與支援

了解常用使用案例

您可以使用 Amazon S3 來支援您的特定使用案例。AWS 解決方案程式庫和 AWS 部落格提供使用案例特定資訊和教學課程。以下是 Amazon S3 的一些常用使用案例：

- 備份與儲存 – 使用 Amazon S3 儲存管理功能以管理成本、滿足法規要求、降低延遲，並儲存多個不同的資料複本，以達到合規要求。
- 應用程式託管 – 部署、安裝與管理可靠、高度可擴展且成本低廉的 Web 應用程式。例如，您可以設定您的 Amazon S3 儲存貯體處理靜態網站託管。如需詳細資訊，請參閱 使用 Amazon S3 託管靜態網站。
- 媒體託管 – 建置可託管影片、相片或音樂上傳與下載的高可用基礎設施。
- 軟體交付 – 託管供客戶下載的軟體應用程式。

控制對儲存貯體與物件的存取

Amazon S3 提供了各種安全性功能和工具。如需概觀，請參閱 存取管理。

根據預設，S3 儲存貯體與物件皆為私有。您只能存取您建立的 S3 資源。您可以使用下列功能授予可支援特定使用案例的精密資源使用權限，或稽核 Amazon S3 資源的許可。

- S3 封鎖公開存取 – 封鎖 S3 儲存貯體與物件的公開存取。根據預設，會在帳戶和儲存貯體層級開啟「封鎖公開存取」設定。
- AWS Identity and Access Management (IAM) 身分識別 — 使用 IAM 或 AWS IAM Identity Center 在您的中建立 IAM 身分， AWS 帳戶 以管理對 Amazon S3 資源的存取。例如，您可以將 IAM 與 Amazon S3 搭配使用，控制使用者或使用者群組對您擁有的 Amazon S3 儲存貯體所 AWS 帳戶 擁有的存取類型。如需 IAM 身分與最佳實務的詳細資訊，請參閱《IAM 使用者指南》中的 IAM 身分 (使用者、群組和角色)。
- 儲存貯體政策 – 使用以 IAM 為基礎的政策語言，為 S3 儲存貯體及其中的物件設定以資源為基礎的許可。

- 存取控制清單 (ACL) – 向授權的使用者授予讀取和寫入個別儲存貯體和物件的許可。一般而言，我們建議使用 S3 以資源為基礎的政策 (儲存貯體政策與存取點原則) 或 IAM 使用者政策來獲取存取控制而非 ACL。政策是一種簡化且更具彈性的存取控制選項。使用儲存貯體政策和存取點政策，您可以定義若干規則，廣泛應用於 Amazon S3 資源的所有請求。如需何時使用 ACL 而非資源型政策或 IAM 使用者政策之特定案例的詳細資訊，請參閱 適用於 Amazon S3 的 Identity and Access Management。
- S3 物件擁有權 - 取得儲存貯體中每個物件的擁有權，簡化存放在 Amazon S3 中資料的存取管理。S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來停用或啟用 ACL。根據預設，會停用 ACL。停用 ACL 後，儲存貯體擁有者會擁有儲存貯體中的所有物件，並使用存取管理政策專門管理對資料的存取。
- IAM Access Analyzer for S3 – 評估和監控您的 S3 儲存貯體存取政策，確保政策僅提供對 S3 資源的預期存取。

管理和監控您的儲存

- 管理儲存 – 在 Amazon S3 中建立儲存貯體並上傳物件之後，您可以管理物件儲存。例如，您可以使用 S3 版本控制和 S3 複寫進行災難復原，使用 S3 生命週期管理儲存成本，以及使用 S3 物件鎖定來達成合規要求。
- 監控儲存 — 監控是維護 Amazon S3 和 AWS 解決方案的可靠性、可用性和效能的重要組成部分。您可以監控儲存的活動和成本。另外，我們建議您全面收集 AWS 解決方案的監控資料，以便在發生多點故障時更容易偵錯。
- 分析與洞見 – 您還可以在 Amazon S3 中使用分析和洞見功能來了解、分析和最佳化儲存使用量。例如，使用 Amazon S3 Storage Lens 來了解、分析和最佳化儲存。S3 Storage Lens 提供超過 29 種用量和活動指標以及互動式儀表板，可彙總整個組織、特定帳戶、區域、儲存貯體或字首的資料。使用儲存方案分析分析儲存存取模式，以決定何時將資料移至更具成本效益的儲存類別。

使用 Amazon S3 進行開發

Amazon S3 是一個 REST 服務。您可以使用 REST API 或開發 AWS 套件程式庫將請求傳送到 Amazon S3，這些程式庫會包裝基礎的 Amazon S3 REST API，以簡化您的程式設計任務。您也可以使用 AWS Command Line Interface (AWS CLI) 來進行 Amazon S3 API 呼叫。如需詳細資訊，請參閱提出要求。

Amazon S3 REST API 是 Amazon S3 的 HTTP 介面。藉助 REST API，您可以使用標準 HTTP 要求來建立、擷取與刪除儲存貯體與物件。若要使用 REST API，您可以使用支援 HTTP 的任何工具組。

您甚至可以使用瀏覽器來擷取物件，只要物件是可匿名讀取即可。如需詳細資訊，請參閱 使用 REST API 與 Amazon S3 進行開發。

為協助您使用所選擇的語言來建置應用程式，我們會提供下列資源：

AWS CLI

您可以使用 AWS CLI來存取 Amazon S3 的功能。若要下載和設定 AWS CLI，請參閱使用 AWS CLI 來透過 Amazon S3 進行開發。

提 AWS CLI 供兩層用於存取 Amazon S3 的命令：高階 (s3) 命令和 API 層級 (s3api 和 s3 control) 命令。高階 S3 命令可簡化執行常見任務，如建立、操作和刪除物件和儲存貯體。s3api 和 s3control 命令可公開直接存取所有 Amazon S3 API 操作，而您可以使用它來執行單獨的高階命令可能無法實現的進階操作。

如需 Amazon S3 AWS CLI 命令的清單，請參閱 s3、s3api 和 s3 控制。

AWS 軟體開發套件和探險者

使用 Amazon S3 開發應用程式時，您可以使用這 AWS 些開發套件。 AWS SDK 透過包裝基礎 REST API 來簡化您的程式設計任務。行 AWS 動開發套件和 Amplify 程式 JavaScript 庫也可用來建置連線的行動和 Web 應用程式。 AWS

除了軟件開 AWS 發套件， AWS 瀏覽器可用於視覺工作室和日食 Java IDE。在這種情況下，SDK 和探險者將作為工具包捆綁在一 AWS 起。

如需詳細資訊，請參閱 使用開發 AWS 套件使用 Amazon S3 進行開發。

範本程式碼與程式庫

AWS 開發人員中心與 AWS 程式碼範例目錄具有專門針對 Amazon S3 撰寫的範本程式碼和程式庫。您可以使用這些程式碼範例來了解如何實作 Amazon S3 API。您也可以檢視《Amazon Simple Storage Service API 參考》，以詳細了解 Amazon S3 API 操作。

從教學課程學習

您可以開始使用 step-by-step 教學課程來進一步了解 Amazon S3。這些教學課程適用於實驗室類型的環境，且它們會使用虛構公司名稱、使用者名稱等。他們的目的是提供一般指導。他們不能直接用於您的生產環境，需要仔細審查更動以符合組織環境的獨特需求。

開始使用

- 教學課程：使用 Amazon S3 存放和擷取檔案

- 教學課程：開始使用 S3 智慧型分層服務
- 教學課程：開始使用 Amazon S3 Glacier 儲存體類別

將儲存體成本最佳化

- 教學課程：開始使用 S3 智慧型分層服務
- 教學課程：開始使用 Amazon S3 Glacier 儲存體類別
- 教學課程：使用 S3 Storage Lens 將成本最佳化並了解使用情況

管理儲存體

- 教學課程：開始使用 Amazon S3 多區域存取點
- 教學課程：使用 S3 批次複寫複寫 Amazon S3 儲存貯體中的現有物件

託管影片和網站

- 教學課程：使用 Amazon S3、Amazon 和 Amazon Route 53 託管隨選串流視訊 CloudFront
- 教學課程：在 Amazon S3 上設定靜態網站
- 教學課程：使用向 Route 53 註冊的自訂網域設定靜態網站

處理資料

- 教學課程：使用 S3 Object Lambda 轉換應用程式的資料
- 教學課程：使用 S3 Object Lambda 和 Amazon Comprehend 來偵測和編輯 PII 資料
- 教學課程：使用 S3 Object Lambda 在擷取影像時動態加上浮水印
- 教學課程：使用 S3 Batch 操作進行 Batch 轉碼影片，以及 AWS LambdaAWS Elemental MediaConvert

保護資料

- 教學課程：使用其他總和檢查來檢查 Amazon S3 中資料的完整性
- 教學課程： AWS 區域 使用 S3 複寫在內部和之間複寫資料

- 教學課程：使用 S3 版本控制、S3 物件鎖定和 S3 複寫，保護 Amazon S3 上的資料，以防意外刪除或發生應用程式錯誤
- 教學課程：使用 S3 批次複寫複寫 Amazon S3 儲存貯體中的現有物件

探索培訓與支援

您可以向 AWS 專家學習，以提升您的技能，並獲得達成目標的專家協助。

- 培訓 – 培訓資源可提供了解 Amazon S3 的實作方式。如需詳細資訊，請參閱 AWS 培訓和認證及 AWS 線上技術講座。
- 開發論壇 – 在論壇上，您可以檢閱貼文，進而了解可以使用 Amazon S3 執行與不可執行的操作。您也可以提出您的問題。如需詳細資訊，請參閱開發論壇。
- 技術支援 – 如果您有其他問題，則可以聯絡技術支援。

教學課程

下列教學提供常見 Amazon S3 任務的完整 end-to-end 程序。這些教學課程適用於實驗室類型的環境，且它們會使用虛構公司名稱、使用者名稱等。他們的目的是提供一般指導。他們不能直接用於您的生產環境，需要仔細審查更動以符合組織環境的獨特需求。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱 什麼是 S3 Express One Zone？ 和 目錄值區。

開始使用

- 教學課程：使用 Amazon S3 存放和擷取檔案
- 教學課程：開始使用 S3 智慧型分層服務
- 教學課程：開始使用 Amazon S3 Glacier 儲存體類別

將儲存體成本最佳化

- 教學課程：開始使用 S3 智慧型分層服務
- 教學課程：開始使用 Amazon S3 Glacier 儲存體類別
- 教學課程：使用 S3 Storage Lens 將成本最佳化並了解使用情況

管理儲存體

- 教學課程：開始使用 Amazon S3 多區域存取點
- 教學課程：使用 S3 批次複寫複寫 Amazon S3 儲存貯體中的現有物件

託管影片和網站

- 教學課程：使用 Amazon S3、Amazon 和 Amazon Route 53 託管隨選串流視訊 CloudFront
- 教學課程：在 Amazon S3 上設定靜態網站

- 教學課程：使用向 Route 53 註冊的自訂網域設定靜態網站

處理資料

- 教學課程：使用 S3 Object Lambda 轉換應用程式的資料
- 教學課程：使用 S3 Object Lambda 和 Amazon Comprehend 來偵測和編輯 PII 資料
- 教學課程：使用 S3 Object Lambda 在擷取影像時動態加上浮水印
- 教學課程：使用 S3 Batch 操作進行 Batch 轉碼影片，以及 AWS LambdaAWS Elemental MediaConvert

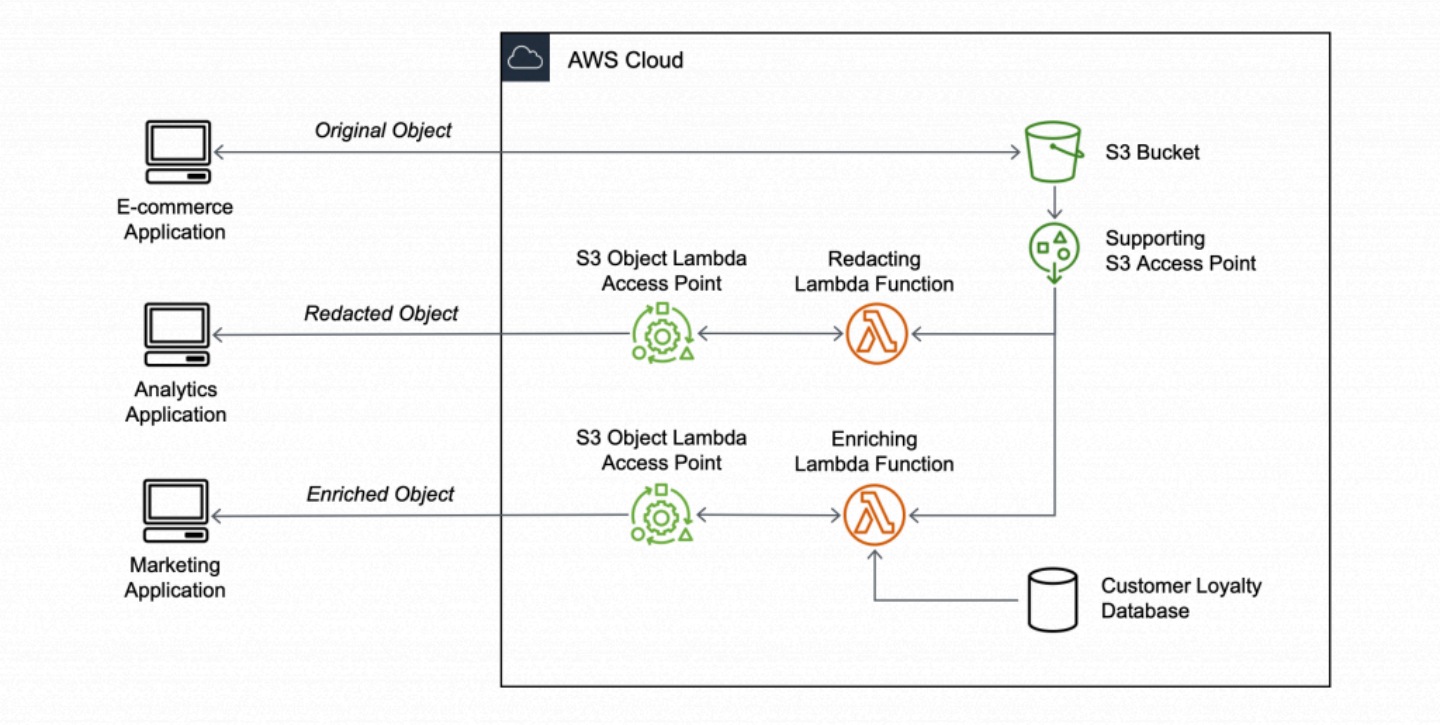
保護資料

- 教學課程：使用其他總和檢查來檢查 Amazon S3 中資料的完整性
- 教學課程： AWS 區域 使用 S3 複寫在內部和之間複寫資料
- 教學課程：使用 S3 版本控制、S3 物件鎖定和 S3 複寫，保護 Amazon S3 上的資料，以防意外刪除或發生應用程式錯誤
- 教學課程：使用 S3 批次複寫複寫 Amazon S3 儲存貯體中的現有物件

教學課程：使用 S3 Object Lambda 轉換應用程式的資料

將資料存放至 Amazon S3 時，您可以輕鬆地進行共用，以供多個應用程式使用。不過，每個應用程式可能有唯一的資料格式需求，而且可能需要針對特定使用案例修改或處理您的資料。例如，電子商務應用程式建立的資料集可能會包含個人身分識別資訊 (PII)。當處理相同的資料以進行分析時，不需要此 PII，而且應該加以修訂。不過，如果同一個資料集用於行銷活動，您可能需要使用其他詳細資訊 (例如來自客戶忠誠度資料庫的資訊) 來豐富資料。

利用 S3 Object Lambda，您可以新增自己的程式碼，以處理從 S3 擷取的資料，然後再將其傳回應用程式。具體而言，您可以設定 AWS Lambda 函數並將其附加至 S3 物件 Lambda 存取點。當應用程式透過 S3 Object Lambda 存取點傳送標準 S3 GET 請求時，則會叫用指定的 Lambda 函數，以處理透過支援 S3 存取點從 S3 儲存貯體擷取的任何資料。然後，S3 Object Lambda 存取點會將轉換的結果傳回至應用程式。您可以編寫和執行自己的自訂 Lambda 函數，從而根據您的特定使用案例，量身定製 S3 Object Lambda 資料轉換，並且全程無需對應用程式進行任何變更。



目標

在本教學課程中，您將學習如何將自訂程式碼新增至標準 S3 GET 請求，以修改從 S3 擷取的請求物件，如此一來，物件方能符合請求用戶端或應用程式的需求。具體而言，您將學習如何透過 S3 Object Lambda 將存放在 S3 中的原始物件中的所有文字轉換為大寫。

主題

必要條件

在開始本教學課程之前，您必須擁有 AWS 帳戶 可以使用正確許可的 AWS Identity and Access Management (IAM) 使用者身分登入的權限。您還必須安裝 Python 3.8 版或更新版本。

子步驟

- 在您的 AWS 帳戶 (主控台) 建立一個擁有許可的 IAM 使用者
- 在您的本機機器上安裝 Python 3.8 或更新版本

在您的 AWS 帳戶 (主控台) 建立一個擁有許可的 IAM 使用者

您可以為該教學課程建立 IAM 使用者。若要完成本教學課程，您的 IAM 使用者必須附加下列 IAM 政策，才能存取相關 AWS 資源並執行特定動作。如需如何建立 IAM 使用者的詳細資訊，請參閱《IAM 使用者指南》中的建立 IAM 政策 (主控台)。

您的 IAM 使用者需要下列政策：

- 亞馬遜 S3 FullAccess — 授予所有 Amazon S3 動作的許可，包括建立和使用物件 Lambda 存取點的許可。
- AWSLambda_FullAccess— 授予所有 Lambda 動作的權限。
- IAM FullAccess — 授予所有 IAM 動作的許可。
- IAM AccessAnalyzerReadOnlyAccess — 授予許可以讀取 IAM 存取分析器提供的所有存取資訊。
- CloudWatchLogsFullAccess— 授與 CloudWatch記錄檔的完整存取權限。

> **Note**
>
> 為了簡單起見，本教學課程會建立和使用 IAM 使用者。完成本教學課程後，請記得 刪除 IAM 使用者。針對生產使用，我們建議您遵循《IAM 使用者指南》中的 IAM 中的安全最佳實務。其中一項最佳實務是，要求人類使用者搭配身分提供者使用聯合功能，以便使用暫時性憑證存取 AWS 。另一項最佳實務要求工作負載使用臨時性憑證和 IAM 角色來存取 AWS。若要瞭解如 AWS IAM Identity Center 何使用建立具有臨時登入資料的使用者，請參閱《使用AWS IAM Identity Center 者指南》中的入門。
> 本教學課程也使用完整存取的 AWS 受管政策。為供生產使用，我們建議您改為僅授予使用案例所需的最低許可，以符合安全最佳實務。

在您的本機機器上安裝 Python 3.8 或更新版本

使用以下程序在您的本機機器上安裝 Python 3.8 或更新版本。如需安裝說明的詳細資訊，請參閱《Python 入門指南》中的下載 Python。

1. 開啟您的本機終端機或 shell 並執行以下命令，以確定是否已經安裝 Python，如果是，那安裝的是哪個版本。

   ```
   python --version
   ```

2. 如果您沒有 Python 3.8 或更新版本，請下載使用您的本機機器的 Python 3.8 的官方安裝程式。
3. 按兩下下載的檔案來執行安裝程式，然後依照步驟完成安裝。

   若為 Windows 使用者，利用安裝精靈選擇 Add Python 3.X to PATH (新增 Python 3.X 至 PATH)，然後選擇 Install Now (立即安裝)。
4. 透過關閉並重新開啟終端機來重新啟動。
5. 執行以下命令，以驗證已正確安裝 Python 3.8 或更新版本。

   若為 macOS 使用者，請執行此命令：

   ```
   python3 --version
   ```

   若為 Windows 使用者，請執行此命令：

   ```
   python --version
   ```

6. 執行下列命令，以來驗證已安裝 pip3 套件管理工具。如果您在命令回應中看到 pip 版本編號和 python 3.8 或更新版本，則意味著已成功安裝 pip3 套件管理工具。

   ```
   pip --version
   ```

步驟 1：建立 S3 儲存貯體

建立儲存貯體來存放您計劃要轉換的原始資料。

建立儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇 Create bucket (建立儲存貯體)。

   Create bucket (建立儲存貯體) 頁面隨即開啟。
4. 針對 Bucket name (儲存貯體名稱)，輸入儲存貯體的名稱 (例如 **`tutorial-bucket`**)。

   如需有關在 Amazon S3 中的命名儲存貯體的詳細資訊，請參閱 儲存貯體命名規則。
5. 在「區域」中，選擇您 AWS 區域 要儲存貯體的位置。

   如需有關儲存貯體區域的詳細資訊，請參閱 儲存貯體概觀 。
6. 針對 Block Public Access settings for this bucket (此儲存貯體的封鎖公開存取設定)，將保留預設設定 (已啟用封鎖所有公開存取)。

   除非您需要針對使用案例關閉一或多個設定，否則建議您將所有封鎖公開存取設定保持啟用狀態。如需封鎖公開存取的詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。
7. 對於其他設定，請保留預設值。

   (選用) 如果您想要為您的特定使用案例設定其他儲存貯體設定，請參閱 建立儲存貯體。
8. 選擇建立儲存貯體。

步驟 2：將檔案上傳至 S3 儲存貯體

上傳文字檔案至 S3 儲存貯體。此文字檔案包含您將在本教學課程稍後部分轉換為大寫的原始資料。

例如，您可以上傳 `tutorial.txt` 檔案，其中包含以下文字：

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

上傳檔案至儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇您在步驟 1 中建立的且要將檔案上傳至的儲存貯體的名稱 (例如，**tutorial-bucket**)。
4. 在儲存貯體的物件索引標籤上，選擇上傳。
5. 在 Upload (上傳) 頁面上的 Files and folders (檔案和資料夾) 下，選擇 Add files (新增檔案)。
6. 選擇要上傳的檔案，然後選擇 Open (開啟)。舉例而言，您可以上傳之前提及的 `tutorial.txt` 檔案範例。
7. 選擇上傳。

步驟 3：建立 S3 存取點

若要使用 S3 Object Lambda 存取點來存取和轉換原始資料，您必須建立 S3 存取點，並將其與您在步驟 1 中建立的 S3 儲存貯體建立關聯。存取點必須與您要轉換 AWS 區域 的物件位於相同的位置。

在本教學課程稍後的部分，您將使用此存取點做為您 Object Lambda 存取點的支援存取點。

建立存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Points (存取點)。
3. 在 Access Points (存取點) 頁面上，選擇 Create access point (建立存取點)。
4. 在 Access point name (存取點名稱) 欄位中，輸入存取點的名稱 (例如，**tutorial-access-point**)。

   如需存取點命名的詳細資訊，請參閱「命名 Amazon S3 存取點的規則」。
5. 在 Bucket name (儲存貯體名稱) 欄位，輸入您在步驟 1 中建立的儲存貯體的名稱 (例如，**tutorial-bucket**)。S3 將存取點連接至此儲存貯體。

   (選用) 您可以選擇 Browse S3 (瀏覽 S3) 來瀏覽並搜尋您帳戶中的儲存貯體。如果您選擇 Browse S3 (瀏覽 S3)，請先選擇所需的儲存貯體，然後選擇 Choose path (選擇路徑)，系統即會在 Bucket name (儲存貯體名稱) 欄位中填入該儲存貯體的名稱。
6. 針對 Network origin (網路來源)，選擇 Internet (網際網路)。

   如需存取點網路來源的詳細資訊，請參閱「建立受限於 Virtual Private Cloud 的存取點」。

7. 依預設，存取點的所有封鎖公開存取設定都會開啟。我們建議您將封鎖所有公開存取保持啟用的狀態。

   如需詳細資訊，請參閱 管理存取點的公開存取。
8. 對於所有其他存取點設定，保留預設設定。

   (選用) 您可以修改存取點設定，以支援您的使用案例。在本教學課程中，我們建議您保留預設設定。

   (選用) 如果您需要管理存取點的存取，您可以指定存取點政策。如需詳細資訊，請參閱 存取點政策範例。
9. 選擇 Create access point (建立新的存取點)。

步驟 4：建立 Lambda 函數

若要轉換原始資料，請建立一個 Lambda 函數，以與您的 S3 Object Lambda 存取點搭配使用。

子步驟

- 使用虛擬環境撰寫 Lambda 函數程式碼並建立部署套件
- 使用執行角色建立 Lambda 函數 (主控台)
- 使用 .zip 檔案封存部署 Lambda 函數程式碼，並設定 Lambda 函數 (主控台)

使用虛擬環境撰寫 Lambda 函數程式碼並建立部署套件

1. 在本機機器上，為虛擬環境建立資料夾名稱為 `object-lambda` 的資料夾，以便用於本教學課程的稍後部分。
2. 在 `object-lambda` 資料夾中，建立具有 Lambda 函數的檔案，而該函數會將原始物件中的所有文字變更為大寫。例如，您可以使用以 Python 撰寫的下列函數。將此函數儲存在名為 `transform.py` 的檔案中。

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

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

```
    # from S3
    s3_url = object_context["inputS3Url"]
    # Extract the route and request token from the input context
    request_route = object_context["outputRoute"]
    request_token = object_context["outputToken"]

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

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

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

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

> **Note**
>
> 上述範例 Lambda 函數將整個請求的文件載入到記憶體中，然後予以轉換並將其傳回給用戶端。或者，您可以從 S3 串流對象，以避免將整個文件載入到記憶體中。處理大型物件時，此方法非常有用。如需使用 Object Lambda 存取點串流回應的詳細資訊，請參閱 使用 Lambda 中的 GetObject 請求 中的串流範例。

當您撰寫與 S3 Object Lambda 存取點搭配使用的 Lambda 函數時，函數會以 S3 Object Lambda 提供給 Lambda 函數的輸入事件內容為基礎。事件內容提供了從 S3 Object Lambda 傳遞給 Lambda 事件中提出之請求的相關資訊。它會包含您用來建立 Lambda 函數的參數。

用來建立前述 Lambda 函數的欄位如下所示：

`getObjectContext` 的欄位意指連線至 Amazon S3 和 S3 Object Lambda 的輸入和輸出詳細資訊。其欄位如下：

- `inputS3Url` – Lambda 函數可用來從支援存取點下載原始物件的預先簽章 URL。透過使用預先簽章 URL，Lambda 函數不需要擁有 Amazon S3 讀取許可即可擷取原始物件，而且只能存取每次叫用處理的物件。
- `outputRoute` – 當 Lambda 函數呼叫 `WriteGetObjectResponse` 以傳回轉換的物件時，會新增至 S3 Object Lambda URL 的路由字符。
- `outputToken` – 當傳回轉換的物件時，S3 Object Lambda 用來將 `WriteGetObjectResponse` 呼叫與原始呼叫者比對的字符。

如需事件內容中的所有欄位的詳細資訊，請參閱 事件內容格式和用量 和 撰寫 S3 Object Lambda 存取點的 Lambda 函數。

3. 在本機終端機中，輸入下列命令來安裝 `virtualenv` 套件：

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

4. 在您的本機終端機中，開啟您之前建立的 `object-lambda` 資料夾，然後輸入下列命令以建立並初始化稱為 `venv` 的虛擬環境。

```
python -m virtualenv venv
```

5. 如要啟用虛擬環境，請輸入下列命令來從環境的檔案局中執行 `activate` 檔案：

   若為 macOS 使用者，請執行此命令：

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

   若為 Windows 使用者，請執行此命令：

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

   現在，您的命令提示字元會變更，以顯示 (venv)，表示虛擬環境作用中。

6. 若要安裝所需的程式庫，請在 `venv` 虛擬環境中逐行執行下列命令。

   這些命令會安裝 `lambda_handler` Lambda 函數相依性的更新版本。這些相依性是 AWS SDK for Python (Boto3) 和請求模組。

```
pip3 install boto3
```

```
pip3 install requests
```

7. 若要停用虛擬環境，請執行下列命令：

```
deactivate
```

8. 若要在 object-lambda 目錄根中將含有已安裝程式庫的部署套件建立為名為 lambda.zip 的 .zip 檔案，請在本機終端機中逐行執行以下命令。

> **Tip**
>
> 下列命令可能需要調整，才能在您的特定環境中運作。例如，程式庫可能會出現在 site-packages 或 dist-packages 中，並且第一個資料夾可能是 lib 或 lib64。此外，python 資料夾可能會使用不同的 Python 版本命名。若要尋找特定套件，使用 pip show 命令。

若為 macOS 使用者，請執行這些命令：

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

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

若為 Windows 使用者，請執行這些命令：

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

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

最後一個命令會將部署套件儲存至 object-lambda 目錄的根目錄。

9. 將函數程式碼檔案 transform.py 新增至部署套件的根目錄。

若為 macOS 使用者，請執行這些命令：

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

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

若為 Windows 使用者，請執行這些命令：

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

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

完成此步驟後，您應具有下列目錄結構：

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

使用執行角色建立 Lambda 函數 (主控台)

1. 請登入 AWS Management Console 並開啟 AWS Lambda 主控台，網址為 https://console.aws.amazon.com/lambda/。
2. 在左側導覽窗格中，選擇 Functions (函數)。
3. 選擇建立函數。
4. 選擇 Author from scratch (從頭開始撰寫)。
5. 在基本資訊下，請執行下列動作：
   a. 針對 函數名稱 ，請輸入 **tutorial-object-lambda-function**。
   b. 針對 Runtime (執行時間)，選擇 Python 3.8 或更新版本。
6. 展開 Change default execution role (變更預設執行角色) 區段。在 Execution role (執行角色) 下，選擇 Create a new role with basic Lambda permissions (建立具備基本 Lambda 許可的新角色)。

在本教學課程稍後的步驟 5 中，您將 AmazonS3 附加ObjectLambdaExecutionRolePolicy到此 Lambda 函數的執行角色。

7. 將其餘設定保持為預設值。
8. 選擇建立函數。

使用 .zip 檔案封存部署 Lambda 函數程式碼，並設定 Lambda 函數 (主控台)

1. 在 https://console.aws.amazon.com/lambda/ 的 AWS Lambda 主控台中，選擇左側導覽窗格中的「函數」。
2. 選擇您之前建立的 Lambda 函數 (例如，**`tutorial-object-lambda-function`**)。
3. 在 Lambda 函數的詳細資訊頁面上，選擇 Code (程式碼) 標籤。在 Code Source (程式碼來源) 區段中，選擇 Upload from (上傳來源)，然後選擇 .zip file (.zip 檔案)。
4. 選擇 Upload (上傳) 以選取您的本機 `.zip` 檔案。
5. 選擇您之前建立的 `lambda.zip` 檔案，然後選擇 Open (開啟)。
6. 選擇儲存。
7. 在 Runtime settings (執行時間設定) 區段中，選擇 Edit (編輯)。
8. 在 Edit runtime settings (編輯執行時間設定) 頁面上，確認 Runtime (執行時間) 已設定為 Python 3.8 或更新版本。
9. 若要告知 Lambda 執行時間要叫用 Lambda 函數程式碼中的處理常式方法，請針對 Handler (處理常式) 輸入 **`transform.lambda_handler`**。

   當您以 Python 設定函式時，處理常式的設定值就是檔案名稱，以及已匯出的處理常式模組名稱，並且以點分隔。例如，`transform.lambda_handler` 會呼叫 `transform.py` 檔案中定義的 `lambda_handler` 方法。
10. 選擇儲存。
11. (選用) 在 Lambda 函數的詳細資訊頁面上，選擇 Configuration (組態) 標籤。在左側導覽窗格中，選擇 General configuration (一般組態)，然後選擇 Edit (編輯)。在 Timeout (逾時) 欄位中，輸入 **`1`** 分 **`0`** 秒。將其餘設定設定為預設值，然後選擇 Save (儲存)。

    Timeout (逾時) 是 Lambda 在停用函數前允許函數執行叫用的時間。預設為 3 秒。S3 Object Lambda 使用的 Lambda 函數的持續時間上限為 60 秒。定價是根據設定的記憶體數量和程式碼執行的時間量而定。

步驟 5：為 Lambda 函數的執行角色設定 IAM 政策

若要啟用 Lambda 函數，將自訂資料和回應標頭提供給 `GetObject` 呼叫者，您 Lambda 函數的執行角色必須具有 IAM 許可，才能呼叫 `WriteGetObjectResponse` API。

若要將 IAM 政策連接到您的 Lambda 函數角色

1. 在 https://console.aws.amazon.com/lambda/ 的 AWS Lambda 主控台中，選擇左側導覽窗格中的「函數」。
2. 選擇您在步驟 4 中建立的函數 (例如，**`tutorial-object-lambda-function`**)。
3. 在 Lambda 函數的詳細資訊頁面上，選擇 Configuration (組態) 標籤，然後在左側導覽窗格選擇Permissions (許可)。
4. 在 Execution role (執行角色)，選擇 Role name (角色名稱)。開啟 IAM 主控台。
5. 在 IAM 主控台的 Summary (摘要) 頁面上，針對您 Lambda 函數的執行角色，選擇 Permissions (許可) 索引標籤。然後，從 Add Permissions (新增許可) 功能表中選擇 Attach policies (附加政策)。
6. 在 Attach permissions (連接許可) 頁面的搜尋方塊中，輸入 **`AmazonS3ObjectLambdaExecutionRolePolicy`**，以篩選政策清單。選取 AmazonS3 ObjectLambdaExecutionRolePolicy 政策名稱旁邊的核取方塊。
7. 選擇連接政策。

步驟 6：建立 S3 Object Lambda 存取點

S3 Object Lambda 存取點提供了直接從 S3 GET 請求叫用 Lambda 函數的靈活性，以便函數可以處理從 S3 存取點擷取的資料。建立及設定 S3 Object Lambda 存取點時，您必須指定要叫用的 Lambda 函數，並以 JSON 格式提供事件內容做為自訂參數以供 Lambda 使用。

建立 S3 Object Lambda 存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側的導覽窗格中，選擇 Object Lambda Access Points (Object Lambda 存取點)。
3. 在 Object Lambda Access Points (Object Lambda 存取點) 頁面上，選擇 Create Object Lambda access point (建立 Object Lambda 存取點)。

4. 對於 Object Lambda 存取點名稱，請輸入您要用於 Object Lambda 存取點的名稱 (例如，**`tutorial-object-lambda-accesspoint`**)。
5. 針對 Supporting Access Point (支援存取點)，輸入或瀏覽您在步驟 3 中建立的標準存取點 (例如，**`tutorial-access-point`**)，然後選擇 Choose supporting Access Point (選擇支援存取點)。
6. 對於 S3 API，若要從 S3 儲存貯體擷取物件以供 Lambda 函數處理，請選取GetObject。
7. 針對 Invoke Lambda function (叫用 Lambda 函數)，您可以為本教學課程選擇以下兩個選項中的任意一個。
   - 從 Lambda function (Lambda 函數) 下拉式清單，選擇 Choose from functions in your account (從您帳戶中的函數選擇)，並選擇您在步驟 4 中建立的 Lambda 函數 (例如，**`tutorial-object-lambda-function`**)。
   - 選擇 Enter ARN (輸入 ARN)，然後輸入您在步驟 4 中建立的 Lambda 函數的 Amazon 資源名稱 (ARN)。
8. 針對 Lambda function version (Lambda 函數版本)，選擇 $LATEST (您在步驟 4 中建立的 Lambda 函數的最新版本)。
9. (選用) 如果您需要 Lambda 函數來識別和處理具有範圍和組件編號標頭的 GET 請求，請選取 Lambda function supports requests using range (Lambda 函數支援使用範圍的請求) 和 Lambda function supports requests using part numbers (Lambda 函數支援使用組件編號的請求)。否則，請清除這兩個核取方塊。

   如需如何藉助 S3 Object Lambda 使用範圍或組件編號的詳細資訊，請參閱「使用 Range 和 partNumber 標頭」。
10. (選用) 在 Payload - optional (酬載 - 選用) 下，新增 JSON 文字，以提供 Lambda 函數的其他資訊。

    承載是可選的 JSON 文本，您可以將其做為來自特定 S3 Object Lambda 存取點的所有叫用的輸入來提供給您的 Lambda 函數。若要針對叫用相同 Lambda 函數的不同 Object Lambda 存取點自訂行為，您可以使用不同參數設定承載，藉此擴充 Lambda 函數的靈活性。

    如需承載的詳細資訊，請參閱「事件內容格式和用量」。
11. (選用) 對於 請求指標 - 選用，請選擇停用或啟用，將 Amazon S3 監控新增至您的 Object Lambda 存取點。請求指標以標準 Amazon 費 CloudWatch 率計費。如需詳細資訊，請參閱 CloudWatch 定價。
12. 在 Object Lambda Access Point policy - optional (Object Lambda 存取點政策 - 選用) 下，請保留預設設定。

(選用) 您可以設定資源政策。此資源政策會授予 `GetObject` API 許可 ，以使用指定的 Object Lambda 存取點。

13. 將其餘設定保持為預設值，並選擇 Create Object Lambda Access Point (建立 Object Lambda 存取點)。

步驟 7：檢視轉換後的資料

現在，S3 Object Lambda 已經準備好為您的使用案例轉換資料。在本教學課程中，S3 Object Lambda 會將物件中的所有文字轉換為大寫。

子步驟

- 在 S3 Object Lambda 存取點中檢視轉換後的資料
- 執行 Python 指令碼，以列印原始資料和轉換的資料

在 S3 Object Lambda 存取點中檢視轉換後的資料

當您請求透過 S3 Object Lambda 存取點擷取檔案時，您要對 S3 Object Lambda 進行 `GetObject` API 呼叫。S3 Object Lambda 會叫用 Lambda 函數來轉換您的資料，然後傳回轉換後的資料，作為對標準 S3 `GetObject` API 呼叫的回應。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側的導覽窗格中，選擇 Object Lambda Access Points (Object Lambda 存取點)。
3. 在 Object Lambda 存取點頁面上，選擇您在步驟 6 中建立的 S3 Object Lambda 存取點 (例如，**`tutorial-object-lambda-accesspoint`**)。
4. 在您的 S3 Object Lambda 存取點的物件標籤上，選取與您在 步驟 2. 中上傳至 S3 儲存貯體的檔案同名的檔案 (例如，`tutorial.txt`)。

   此檔案應包含所有轉換的資料。
5. 若要檢視轉換的資料，選擇 Open (開啟) 或 Download (下載)。

執行 Python 指令碼，以列印原始資料和轉換的資料

您可以將 S3 Object Lambda 與您現有的應用程式搭配使用。若要執行這項操作，請更新您的應用程式組態，以使用您在步驟 6 中建立的新的 S3 Object Lambda 存取點 ARN，進而從 S3 擷取資料。

下列範例 Python 指令碼會同時列印來自 S3 儲存貯體的原始資料，以及來自 S3 Object Lambda 存取點的轉換資料。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側的導覽窗格中，選擇 Object Lambda Access Points (Object Lambda 存取點)。
3. 在 Object Lambda 存取點頁面上，選擇您在步驟 6 中建立的 S3 Object Lambda 存取點左側的選項按鈕 (例如，**`tutorial-object-lambda-accesspoint`**)。
4. 選擇 Copy ARN (複製 ARN)。
5. 儲存 ARN 以供稍後使用。
6. 在本機機器上編寫 Python 指令碼，以列印來自 S3 儲存貯體的原始資料 (例如，`tutorial.txt`) 和來自 S3 Object Lambda 存取點的轉換後的資料 (例如，`tutorial.txt`)。您可使用下列範例指令碼。

```
import boto3
from botocore.config import Config

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

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

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

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

7. 在您本機機器上使用自訂名稱將您的 Python 指令碼 (例如，`tutorial_print.py`) 儲存至您在步驟 4 中建立的資料夾中 (例如，`object-lambda`)。
8. 在本機終端機中，從您在步驟 4 中建立的目錄的根中執行下列命令 (例如，`object-lambda`)。

```
python3 tutorial_print.py
```

您應該透過終端機查看原始資料和轉換的資料 (所有文字均為大寫)。例如，您應該會看到類似下列文字的內容。

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

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

步驟 8：清除

如果透過 S3 Object Lambda 轉換的資料僅供學習練習之用，請先刪除已配置的 AWS 資源，如此即不會再產生費用。

子步驟

- 刪除 Object Lambda 存取點
- 刪除 S3 存取點
- 為您的 Lambda 函數刪除執行角色
- 刪除 Lambda 函數
- 刪除記 CloudWatch 錄群組
- 刪除 S3 來源儲存貯體中的原始檔案
- 刪除 S3 來源儲存貯體
- 刪除 IAM 使用者

刪除 Object Lambda 存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側的導覽窗格中，選擇 Object Lambda Access Points (Object Lambda 存取點)。
3. 在 Object Lambda 存取點頁面上，選擇您在步驟 6 中建立的 S3 Object Lambda 存取點左側的選項按鈕 (例如，**tutorial-object-lambda-accesspoint**)。
4. 選擇刪除。
5. 在出現的文字欄位中，輸入存取點名稱，以確認您要刪除 Object Lambda 存取點，然後選擇刪除。

刪除 S3 存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Points (存取點)。
3. 導覽至您在步驟 3 中建立的存取點 (例如，**tutorial-access-point**)，然後選擇存取點名稱旁的選項按鈕。
4. 選擇刪除。
5. 在出現的文字欄位中，輸入存取點名稱，以確認您要刪除此存取點，然後選擇 Delete (刪除)。

為您的 Lambda 函數刪除執行角色

1. 請登入 AWS Management Console 並開啟 AWS Lambda 主控台，網址為 https://console.aws.amazon.com/lambda/。
2. 在左側導覽窗格中，選擇 Functions (函數)。
3. 選擇您在步驟 4 中建立的函數 (例如，**tutorial-object-lambda-function**)。
4. 在 Lambda 函數的詳細資訊頁面上，選擇 Configuration (組態) 標籤，然後在左側導覽窗格選擇Permissions (許可)。
5. 在 Execution role (執行角色)，選擇 Role name (角色名稱)。開啟 IAM 主控台。
6. 在 IAM 主控台的 Lambda 函數的執行角色的 Summary (摘要) 頁面，選擇 Delete role (刪除角色)。
7. 在 Delete role (刪除角色) 對話方塊中，選擇 Yes, delete (是，刪除)。

刪除 Lambda 函數

1. 在 https://console.aws.amazon.com/lambda/ 的 AWS Lambda 主控台中，選擇左側導覽窗格中的「函數」。
2. 選取您在步驟 4 中建立的函數名稱的左側的核取方塊 (例如，**`tutorial-object-lambda-function`**)。
3. 選擇動作，然後選擇刪除。
4. 在 Delete function (刪除函數) 對話方塊中，選擇 Delete (刪除)。

刪除記 CloudWatch 錄群組

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在左側導覽窗格中，選擇 Log groups (日誌群組)。
3. 尋找您在步驟 4 中建立的且名稱以 Lambda 函數結尾的日誌群組 (例如，**`tutorial-object-lambda-function`**)。
4. 選取日誌群組名稱左側的核取方塊。
5. 選擇 Actions (動作)，然後選擇 Delete log group(s) (刪除日誌群組)。
6. 在 刪除日誌群組 對話方塊中，選擇 刪除 。

刪除 S3 來源儲存貯體中的原始檔案

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Bucket name (儲存貯體名稱) 清單中，選擇您在步驟 2 中將原始檔案上傳到的儲存貯體的名稱 (例如，**`tutorial-bucket`**)。
4. 選取要刪除之物件名稱左側的核取方塊 (例如，`tutorial.txt`)。
5. 選擇刪除。
6. 在 Delete objects (刪除物件) 頁面上的 Permanently delete objects? (永久刪除物件？) 區段中，在文字方塊中輸入 **`permanently delete`**，以確認您要刪除此物件。
7. 選擇 Delete objects (刪除物件)。

刪除 S3 來源儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇您在步驟 1 中建立的儲存貯體名稱旁的選項按鈕 (例如，**`tutorial-bucket`**)。
4. 選擇刪除。
5. 在 Delete bucket (刪除儲存貯體) 頁面上，在文字欄位中輸入儲存貯體名稱以確認您要刪除該儲存貯體，然後選擇 Delete bucket (刪除儲存貯體)。

刪除 IAM 使用者

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中，選擇 Users (使用者)，然後選取您要刪除之使用者名稱旁的核取方塊。
3. 在頁面頂端，選擇 Delete (刪除)。
4. 在 Delete ***`user name`***? (刪除使用者名稱？) 對話方塊中，在文字輸入欄位中輸入使用者名稱以確認刪除使用者。選擇刪除。

後續步驟

完成本教學課程後，您可以針對您的使用案例自訂 Lambda 函數，進而修改標準 S3 GET 請求傳回的資料。

以下是 S3 Object Lambda 的常用使用案例清單：

- 遮罩敏感資料以確保安全與合規。

  如需詳細資訊，請參閱 教學課程：使用 S3 Object Lambda 和 Amazon Comprehend 來偵測和編輯 PII 資料。
- 篩選某些資料列，以傳遞特定資訊。
- 使用來自其他服務或資料庫的資訊增強資料。
- 跨資料格式轉換，例如將 XML 轉換為 JSON 以獲得應用程式的相容性。
- 在下載檔案時壓縮或解壓縮檔案。

- 調整圖像大小和為其浮水印。

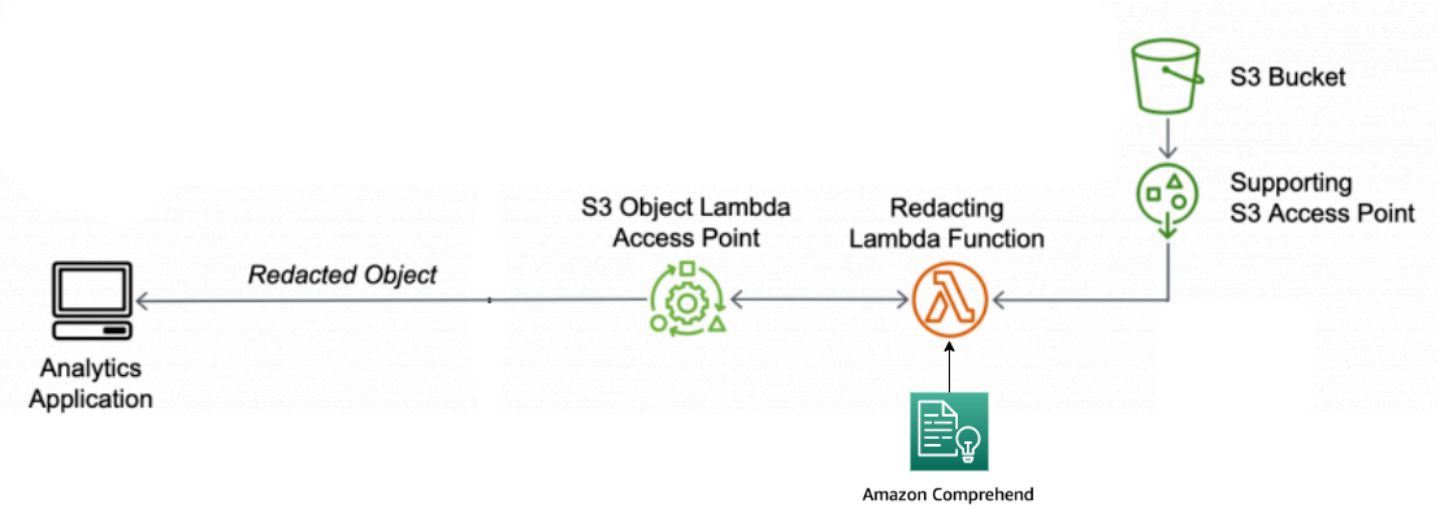
  如需詳細資訊，請參閱教學課程：使用 S3 Object Lambda 在擷取影像時動態加上浮水印。
- 實作自訂授權規則以存取資料。

如需 S3 Object Lambda 的詳細資訊，請參閱 使用 S3 Object Lambda 轉換物件。

教學課程：使用 S3 Object Lambda 和 Amazon Comprehend 來偵測和編輯 PII 資料

當您將 Amazon S3 用於多個應用程式和使用者存取的共用資料集時，請務必將特許資訊 (例如個人身分識別資訊 (PII)) 限制為授權的實體。例如，當行銷應用程式使用某些包含 PII 的資料時，可能需要先遮罩 PII 資料，才能符合資料隱私權需求。此外，當分析應用程式使用生產訂單清查資料集時，可能需要先修訂客戶信用卡資訊，以防止意外的資料外洩。

搭配使用 S3 Object Lambda 和由 Amazon Comprehend 提供的預先建置的 AWS Lambda 函數，您可以保護從 S3 擷取的 PII 資料，然後再將其傳回應用程式。具體而言，您可以使用預先建置的 Lambda 函數做為修訂函數，並將其連接到 S3 Object Lambda 存取點。當應用程式 (例如，分析應用程式) 傳送標準 S3 GET 請求時，這些透過 S3 Object Lambda 存取點提出的請求會叫用預先建置的修訂 Lambda 函數，進而偵測並修訂透過支援 S3 存取點從 S3 儲存貯體擷取的 PII 資料。然後，S3 Object Lambda 存取點會將修訂的結果傳回至應用程式。



在該過程中，預先建置的 Lambda 函數使用自然語言處理 (NLP) 服務的 Amazon Comprehend，擷取 PII 表示方式的變化，而不論 PII 在文字中的存在方式 (例如數字或文字與數字的組合)。Amazon Comprehend 甚至使用文字中的內容來了解一個 4 位數的數字是一個 PIN、社會安全號碼 (SSN) 的最後四個數字還是年份。Amazon Comprehend 處理 UTF-8 格式的任何文字檔案，並可以大規模保護

PII，而不會影響準確性。如需詳細資訊，請參閱《Amazon Comprehend 開發人員指南》中的什麼是 Amazon Comprehend？。

目標

在本教學課程中，您將學習如何搭配使用 S3 Object Lambda 和預先建置的 Lambda 函數 `ComprehendPiiRedactionS3ObjectLambda`。此函數使用 Amazon Comprehend 來檢測 PII 實體。然後，它會以星號取代這些實體，進而予以修訂。透過修訂 PII，您可以隱藏敏感資料，而這有助於維持安全與合規。

您也可以在中學習如何使用和設定預先建置的 AWS Lambda 函數，AWS Serverless Application Repository以便與 S3 物件 Lambda 搭配使用，以便輕鬆部署。

主題

先決條件：建立具有許可的 IAM 使用者

在開始本教學課程之前，您必須擁有一個可以使用正確許可的 AWS Identity and Access Management 使用者 (IAM 使用者) 身分登入的 AWS 帳戶。

您可以為該教學課程建立 IAM 使用者。若要完成本教學課程，您的 IAM 使用者必須附加下列 IAM 政策，才能存取相關 AWS 資源並執行特定動作。

> **Note**
>
> 為了簡單起見，本教學課程會建立和使用 IAM 使用者。完成本教學課程後，請記得 刪除 IAM 使用者。針對生產使用，我們建議您遵循《IAM 使用者指南》中的 IAM 中的安全最佳實務。其中一項最佳實務是，要求人類使用者搭配身分提供者使用聯合功能，以便使用暫時性憑證存

取 AWS 。另一項最佳實務要求工作負載使用臨時性憑證和 IAM 角色來存取 AWS。若要瞭解如 AWS IAM Identity Center 何使用建立具有臨時登入資料的使用者，請參閱《使用AWS IAM Identity Center 者指南》中的入門。
本教學課程也使用完整存取的政策。為供生產使用，我們建議您改為僅授予使用案例所需的最低許可，以符合安全最佳實務。

您的 IAM 使用者需要下列 AWS 受管政策：

- 亞馬遜 S3 FullAccess — 授予所有 Amazon S3 動作的許可，包括建立和使用物件 Lambda 存取點的許可。
- AWSLambda_FullAccess— 授予所有 Lambda 動作的權限。
- AWSCloudFormationFullAccess— 授予所有 AWS CloudFormation 動作的權限。
- IAM FullAccess — 授予所有 IAM 動作的許可。
- IAM AccessAnalyzerReadOnlyAccess — 授予許可以讀取 IAM 存取分析器提供的所有存取資訊。

您可以在建立 IAM 使用者時直接連接這些現有政策。如需如何建立 IAM 使用者的詳細資訊，請參閱《IAM 使用者指南》中的建立 IAM 政策 (主控台)。

此外，您的 IAM 使用者需要客戶受管政策。若要將 IAM 使用者許可授與所有 AWS Serverless Application Repository 資源和動作，您必須建立 IAM 政策並將該政策附加到 IAM 使用者。

建立 IAM 政策並將其連接至您的 IAM 使用者

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中選擇 Policies (政策)。
3. 選擇建立政策。
4. 在 Visual editor (視覺化編輯器) 索引標籤上，針對 Service (服務)，選擇 Choose a service (選擇服務)。然後，選擇 Serverless Application Repository。
5. 針對 Actions (動作)，在 Manual actions (手動動作) 下，為本教學課程選取 All Serverless Application Repository actions (serverlessrepo:*) (所有 Serverless Application Repository 動作 (serverlessrepo:*)。

   作為安全最佳實務，您應該根據您的使用案例，僅允許使用者需要的那些動作和資源的許可。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的安全最佳實務。

6. 針對 Resources (資源)，為本教學課程選擇 All resources (所有資源)。

   作為最佳實務，您應該僅為特定帳戶中的特定資源定義許可。或者，您也可以使用條件金鑰授予最低權限。如需詳細資訊，請參閱《IAM 使用者指南》中的授予最低權限。

7. 選擇下一步：標籤。
8. 選擇下一步：檢閱。
9. 在 Review policy (檢閱政策) 頁面上，為您正在建立的政策輸入 Name (名稱) (例如，**tutorial-serverless-application-repository**) 與 Description (描述) (選用)。檢閱政策摘要以確認您已授予所需的許可，然後選擇 Create policy (建立政策) 來儲存您的新政策。
10. 在左側導覽窗格中，選擇 Users (使用者)。然後，選擇本教學課程的 IAM 使用者。
11. 在所選使用者的 Summary (摘要) 頁面上，選擇 Permissions (許可) 標籤，然後選擇 Add permissions (新增許可)。
12. 在 Grant permissions (授予許可) 下，選擇 Attach existing policies directly (直接連接現有政策)。
13. 選取您剛建立之政策旁的核取方塊 (例如，**tutorial-serverless-application-repository**)，然後選擇 Next: Review (下一步：檢閱)。
14. 在 Permissions summary (許可摘要) 下，檢閱摘要以確認您已連接想要的政策。然後，選擇 Add permissions (新增許可)。

步驟 1：建立 S3 儲存貯體

建立儲存貯體來存放您計劃要轉換的原始資料。

建立儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇 Create bucket (建立儲存貯體)。

   Create bucket (建立儲存貯體) 頁面隨即開啟。

4. 針對 Bucket name (儲存貯體名稱)，輸入儲存貯體的名稱 (例如 **tutorial-bucket**)。

   如需有關在 Amazon S3 中的命名儲存貯體的詳細資訊，請參閱 儲存貯體命名規則。

5. 針對 Region (區域)，選擇希望存放儲存貯體的 AWS 區域 。

   如需有關儲存貯體區域的詳細資訊，請參閱 儲存貯體概觀 。

6. 針對 Block Public Access settings for this bucket (此儲存貯體的封鎖公開存取設定)，將保留預設設定 (已啟用封鎖所有公開存取)。

   除非您需要針對使用案例關閉一或多個設定，否則建議您將所有封鎖公開存取設定保持啟用狀態。如需封鎖公開存取的詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。
7. 對於其他設定，請保留預設值。

   (選用) 如果您想要為您的特定使用案例設定其他儲存貯體設定，請參閱 建立儲存貯體。
8. 選擇建立儲存貯體。

步驟 2：將檔案上傳至 S3 儲存貯體

將包含各種類型的已知 PII 資料 (例如姓名、銀行資訊、電話號碼和 SSN) 的文字檔案作為原始資料上傳至 S3 儲存貯體，而您將在本教學課程的之後部分修訂 PII。

例如，您可以上傳下列 `tutorial.txt` 檔案。這是 Amazon Comprehend 的輸入檔案範例。

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

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

上傳檔案至儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇您在步驟 1 中建立的且要將檔案上傳至的儲存貯體的名稱 (例如，**`tutorial-bucket`**)。
4. 在儲存貯體的物件索引標籤上，選擇上傳。

5. 在 Upload (上傳) 頁面上的 Files and folders (檔案和資料夾) 下，選擇 Add files (新增檔案)。
6. 選擇要上傳的檔案，然後選擇 Open (開啟)。舉例而言，您可以上傳之前提及的 `tutorial.txt` 檔案範例。
7. 選擇上傳。

步驟 3：建立 S3 存取點

若要使用 S3 Object Lambda 存取點來存取和轉換原始資料，您必須建立 S3 存取點，並將其與您在步驟 1 中建立的 S3 儲存貯體建立關聯。存取點必須與您要轉換的物件位於相同 AWS 區域 的位置。

在本教學課程稍後的部分，您將使用此存取點做為您 Object Lambda 存取點的支援存取點。

建立存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Points (存取點)。
3. 在 Access Points (存取點) 頁面上，選擇 Create access point (建立存取點)。
4. 在 Access point name (存取點名稱) 欄位中，輸入存取點的名稱 (例如，**`tutorial-pii-access-point`**)。

   如需存取點命名的詳細資訊，請參閱「命名 Amazon S3 存取點的規則」。
5. 在 Bucket name (儲存貯體名稱) 欄位，輸入您在步驟 1 中建立的儲存貯體的名稱 (例如，**`tutorial-bucket`**)。S3 將存取點連接至此儲存貯體。

   (選用) 您可以選擇 Browse S3 (瀏覽 S3) 來瀏覽並搜尋您帳戶中的儲存貯體。如果您選擇 Browse S3 (瀏覽 S3)，請先選擇所需的儲存貯體，然後選擇 Choose path (選擇路徑)，系統即會在 Bucket name (儲存貯體名稱) 欄位中填入該儲存貯體的名稱。
6. 針對 Network origin (網路來源)，選擇 Internet (網際網路)。

   如需存取點網路來源的詳細資訊，請參閱「建立受限於 Virtual Private Cloud 的存取點」。
7. 依預設，存取點的所有封鎖公開存取設定都會開啟。我們建議您將封鎖所有公開存取保持啟用的狀態。如需詳細資訊，請參閱 管理存取點的公開存取。
8. 對於所有其他存取點設定，保留預設設定。

   (選用) 您可以修改存取點設定，以支援您的使用案例。在本教學課程中，我們建議您保留預設設定。

(選用) 如果您需要管理存取點的存取，您可以指定存取點政策。如需詳細資訊，請參閱 存取點政策範例。

9. 選擇 Create access point (建立新的存取點)。

步驟 4：設定及部署預先建置的 Lambda 函數

若要修訂 PII 資料，請設定並部署預先建置的 AWS Lambda 函數 `ComprehendPiiRedactionS3ObjectLambda`，以與您的 S3 Object Lambda 存取點搭配使用。

設定和部署 Lambda 函數

1. 登入 AWS Management Console 並檢視中的`ComprehendPiiRedactionS3ObjectLambda`函數 AWS Serverless Application Repository。
2. 針對 Application settings (應用程式設定)，在 Application name (應用程式名稱) 下，為本教學課程保留預設值 (`ComprehendPiiRedactionS3ObjectLambda`)。

   (選用) 您可以輸入您要給予此應用程式的名稱。如果您計劃針對相同共用資料集的不同存取需求，設定多個 Lambda 函數，您可能會想要這麼做。
3. 對於 MaskCharacter，保留預設值 (**)。遮罩字元會取代已修訂 PII 實體中的每個字元。
4. 對於 MaskMode，保留預設值 (遮罩)。此MaskMode值指定 PII 實體是使用`MASK`字元還是值來編輯。`PII_ENTITY_TYPE`
5. 若要編輯指定類型的資料 PiiEntityTypes，請保留預設值 ALL。該PiiEntityTypes值指定要考慮進行密文的 PII 實體類型。

   如需支援的 PII 實體類型清單的詳細資訊，請參閱《Amazon Comprehend 開發人員指南》中的偵測個人身分識別資訊 (PII)。
6. 將其餘設定保持為預設值。

   (選用) 如果您想要為您的特定使用案例設定其他設定，請參閱位於頁面左側的 Readme 檔案。
7. 選取 I acknowledge that this app creates custom IAM roles (我認可此應用程式建立自訂的 IAM 角色) 旁的核取方塊。
8. 選擇部署。
9. 在新應用程式的頁面，在 Resources (資源) 下，選擇您部署的 Lambda 函數的 Logical ID (邏輯 ID)，以檢閱 Lambda function (Lambda 函數) 頁面上的函數。

步驟 5：建立 S3 Object Lambda 存取點

S3 Object Lambda 存取點提供了直接從 S3 GET 請求叫用 Lambda 函數的靈活性，以便函數可以修訂從 S3 存取點擷取的 PII 資料。建立及設定 S3 Object Lambda 存取點時，您必須指定要叫用的修訂 Lambda 函數，並以 JSON 格式提供事件內容做為自訂參數以供 Lambda 使用。

事件內容提供了從 S3 Object Lambda 傳遞給 Lambda 事件中提出之請求的相關資訊。如需事件內容中所有欄位的詳細資訊，請參閱 事件內容格式和用量。

建立 S3 Object Lambda 存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側的導覽窗格中，選擇 Object Lambda Access Points (Object Lambda 存取點)。
3. 在 Object Lambda Access Points (Object Lambda 存取點) 頁面上，選擇 Create Object Lambda access point (建立 Object Lambda 存取點)。
4. 對於 Object Lambda 存取點名稱，請輸入您要用於 Object Lambda 存取點的名稱 (例如，**`tutorial-pii-object-lambda-accesspoint`**)。
5. 針對 Supporting Access Point (支援存取點)，輸入或瀏覽您在步驟 3 中建立的標準存取點 (例如，**`tutorial-pii-access-point`**)，然後選擇 Choose supporting Access Point (選擇支援存取點)。
6. 對於 S3 API，若要從 S3 儲存貯體擷取物件以供 Lambda 函數處理，請選取GetObject。
7. 針對 Invoke Lambda function (叫用 Lambda 函數)，您可以為本教學課程選擇以下兩個選項中的任意一個。
   - 從 Lambda function (Lambda 函數) 下拉式清單，選擇 Choose from functions in your account (從您帳戶中的函數選擇)，並選擇您在步驟 4 中部署的 Lambda 函數 (例如，**`serverlessrepo-ComprehendPiiRedactionS3ObjectLambda`**)。
   - 選擇 Enter ARN (輸入 ARN)，然後輸入您在步驟 4 中建立的 Lambda 函數的 Amazon 資源名稱 (ARN)。
8. 針對 Lambda function version (Lambda 函數版本)，選擇 $LATEST (您在步驟 4 中部署的 Lambda 函數的最新版本)。
9. (選用) 如果您需要 Lambda 函數來識別和處理具有範圍和組件編號標頭的 GET 請求，請選取 Lambda function supports requests using range (Lambda 函數支援使用範圍的請求) 和 Lambda function supports requests using part numbers (Lambda 函數支援使用組件編號的請求)。否則，請清除這兩個核取方塊。

如需如何藉助 S3 Object Lambda 使用範圍或組件編號的詳細資訊，請參閱「使用 Range 和 partNumber 標頭」。

10. (選用) 在 Payload - optional (酬載 - 選用) 下，新增 JSON 文字，以提供 Lambda 函數的其他資訊。

    承載是可選的 JSON 文本，您可以將其做為來自特定 S3 Object Lambda 存取點的所有叫用的輸入來提供給您的 Lambda 函數。若要針對叫用相同 Lambda 函數的不同 Object Lambda 存取點自訂行為，您可以使用不同參數設定承載，藉此擴充 Lambda 函數的靈活性。

    如需承載的詳細資訊，請參閱「事件內容格式和用量」。

11. (選用) 對於 請求指標 - 選用，請選擇停用或啟用，將 Amazon S3 監控新增至您的 Object Lambda 存取點。請求指標以標準 Amazon 費 CloudWatch 率計費。如需詳細資訊，請參閱 CloudWatch 定價。

12. 在 Object Lambda Access Point policy - optional (Object Lambda 存取點政策 - 選用) 下，請保留預設設定。

    (選用) 您可以設定資源政策。此資源政策會授予 `GetObject` API 許可，以使用指定的 Object Lambda 存取點。

13. 將其餘設定保持為預設值，並選擇 Create Object Lambda Access Point (建立 Object Lambda 存取點)。

步驟 6：使用 S3 Object Lambda 存取點擷取已修訂的檔案

現在，S3 Object Lambda 已經準備好從您的原始檔案修訂 PII 資料。

若要使用 S3 Object Lambda 存取點擷取修訂的檔案

當您請求透過 S3 Object Lambda 存取點擷取檔案時，您要對 S3 Object Lambda 進行 `GetObject` API 呼叫。S3 Object Lambda 會叫用 Lambda 函數來修改您的 PII 資料，並傳回轉換後的資料，作為對標準 S3 `GetObject` API 呼叫的回應。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側的導覽窗格中，選擇 Object Lambda Access Points (Object Lambda 存取點)。
3. 在 Object Lambda 存取點頁面上，選擇您在步驟 5 中建立的 S3 Object Lambda 存取點 (例如，**`tutorial-pii-object-lambda-accesspoint`**)。

4. 在您的 S3 Object Lambda 存取點的物件標籤上，選取與您在 步驟 2. 中上傳至 S3 儲存貯體的檔案同名的檔案 (例如，`tutorial.txt`)。

   此檔案應包含所有轉換的資料。

5. 若要檢視轉換的資料，選擇 Open (開啟) 或 Download (下載)。

   您應能夠看到已修訂的檔案，如下方範例所示。

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

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

步驟 7：清除

如果您只是透過 S3 Object Lambda 編輯資料作為學習練習，請刪除您分配的 AWS 資源，以免再產生費用。

子步驟

- 刪除 Object Lambda 存取點
- 刪除 S3 存取點
- 刪除 Lambda 函數
- 刪除記 CloudWatch錄群組
- 刪除 S3 來源儲存貯體中的原始檔案
- 刪除 S3 來源儲存貯體
- 為您的 Lambda 函數刪除 IAM 角色
- 刪除 IAM 使用者的客戶受管政策
- 刪除 IAM 使用者

刪除 Object Lambda 存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側的導覽窗格中，選擇 Object Lambda Access Points (Object Lambda 存取點)。
3. 在 Object Lambda 存取點頁面上，選擇您在步驟 5 中建立的 S3 Object Lambda 存取點左側的選項按鈕 (例如，**tutorial-pii-object-lambda-accesspoint**)。
4. 選擇刪除。
5. 在出現的文字欄位中，輸入存取點名稱，以確認您要刪除 Object Lambda 存取點，然後選擇刪除。

刪除 S3 存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Points (存取點)。
3. 導覽至您在步驟 3 中建立的存取點 (例如，**tutorial-pii-access-point**)，然後選擇存取點名稱旁的選項按鈕。
4. 選擇刪除。
5. 在出現的文字欄位中，輸入存取點名稱，以確認您要刪除此存取點，然後選擇 Delete (刪除)。

刪除 Lambda 函數

1. 在 https://console.aws.amazon.com/lambda/ 的 AWS Lambda 主控台中，選擇左側導覽窗格中的「函數」。
2. 選擇您在步驟 4 中建立的函數 (例如，**serverlessrepo-ComprehendPiiRedactionS3ObjectLambda**)。
3. 選擇動作，然後選擇刪除。
4. 在 Delete function (刪除函數) 對話方塊中，選擇 Delete (刪除)。

刪除記 CloudWatch錄群組

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在左側導覽窗格中，選擇 Log groups (日誌群組)。

3. 尋找您在步驟 4 中建立的且名稱以 Lambda 函數結尾的日誌群組 (例如，**serverlessrepo-ComprehendPiiRedactionS3ObjectLambda**)。
4. 選擇 Actions (動作)，然後選擇 Delete log group(s) (刪除日誌群組)。
5. 在 刪除日誌群組 對話方塊中，選擇 刪除 。

刪除 S3 來源儲存貯體中的原始檔案

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Bucket name (儲存貯體名稱) 清單中，選擇您在步驟 2 中將原始檔案上傳到的儲存貯體的名稱 (例如，**tutorial-bucket**)。
4. 選取要刪除之物件名稱左側的核取方塊 (例如，`tutorial.txt`)。
5. 選擇刪除。
6. 在 Delete objects (刪除物件) 頁面上的 Permanently delete objects? (永久刪除物件？) 區段中，在文字方塊中輸入 **permanently delete**，以確認您要刪除此物件。
7. 選擇 Delete objects (刪除物件)。

刪除 S3 來源儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇您在步驟 1 中建立的儲存貯體名稱旁的選項按鈕 (例如，**tutorial-bucket**)。
4. 選擇刪除。
5. 在 Delete bucket (刪除儲存貯體) 頁面上，在文字欄位中輸入儲存貯體名稱以確認您要刪除該儲存貯體，然後選擇 Delete bucket (刪除儲存貯體)。

為您的 Lambda 函數刪除 IAM 角色

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。

2. 在左側導覽窗格中，選擇 Roles (角色)，然後選取您要刪除之角色名稱旁的核取方塊。角色名稱以您在步驟 4 中部署的 Lambda 函數的名稱為開頭 (例如，**serverlessrepo-ComprehendPiiRedactionS3ObjectLambda**)。
3. 選擇刪除。
4. 在 Delete (刪除) 對話方塊中，在文字輸入欄位中輸入角色名稱以確認刪除。再選擇 Delete (刪除)。

刪除 IAM 使用者的客戶受管政策

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中選擇 Policies (政策)。
3. 在 Policies (政策) 頁面上，在搜尋方塊中輸入您在 Prerequisites (先決條件) 中建立的客戶受管政策的名稱 (例如，**tutorial-serverless-application-repository**)，以篩選政策清單。選取您要刪除的政策名稱旁的選項按鈕。
4. 選擇動作，然後選擇刪除。
5. 在顯示的文字欄位中，輸入本政策的名稱，以確認您要刪除此政策，然後選擇 Delete (刪除)。

刪除 IAM 使用者

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中，選擇 Users (使用者)，然後選取您要刪除之使用者名稱旁的核取方塊。
3. 在頁面頂端，選擇 Delete (刪除)。
4. 在 Delete ***user name***? (刪除使用者名稱？) 對話方塊中，在文字輸入欄位中輸入使用者名稱以確認刪除使用者。選擇刪除。

後續步驟

完成本教學課程之後，您可以進一步探索下列相關的使用案例：

- 您可以建立多個 S3 Object Lambda 存取點，並使用預先建置的 Lambda 函數啟用這些函數，其中這些函數的設定不同，以根據資料存取者的業務需求修訂特定類型的 PII。

每種類型的使用者都具有 IAM 角色，且只能存取一個 S3 Object Lambda 存取點 (透過 IAM 政策管理)。然後，您連接每個 `ComprehendPiiRedactionS3ObjectLambda` 針對不同 S3 Object Lambda 存取點的不同修訂使用案例設定 Lambda 函數。對於每個 S3 Object Lambda 存取點，您可以擁有支援的 S3 存取點，以便從存放共用資料集的 S3 儲存貯體讀取資料。

如需如何建立 S3 儲存貯體政策，以允許使用者僅透過 S3 存取點讀取儲存貯體的詳細資訊，請參閱「配置使用存取點的 IAM 原則」。

如需如何授予使用者存取 Lambda 函數、S3 存取點及 S3 Object Lambda 存取點之許可的相關資訊，請參閱 設定 Object Lambda 存取點的 IAM 政策。

- 您可以建置自己的 Lambda 函數，並將 S3 Object Lambda 搭配您自訂的 Lambda 函數使用，以滿足您的特定資料需求。

  例如，若要探索各種資料值，您可以使用 S3 Object Lambda 和您自己的 Lambda 函數，其中該函數會使用其他 Amazon Comprehend 功能 (例如實體辨識、金鑰片語辨識、情感分析和文件分類) 來處理資料。您也可以將 S3 Object Lambda 與 Amazon Comprehend Medical 搭配使用，後者是符合 HIPAA 資格的 NLP 服務，以內容感知的方式分析和擷取資料。

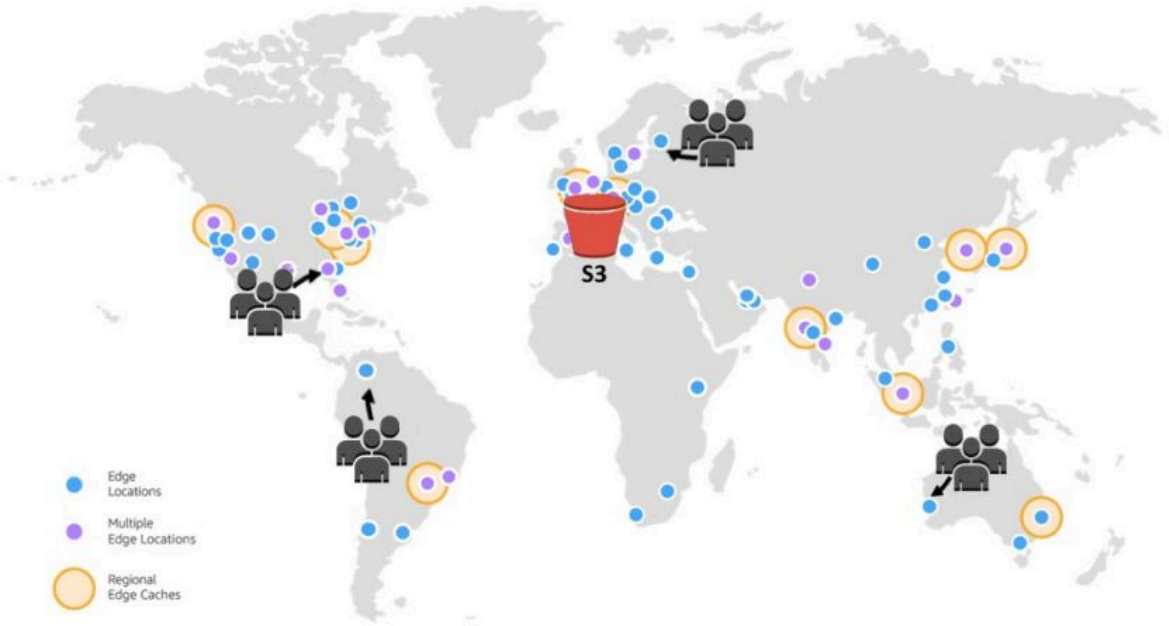
  如需如何使用 S3 Object Lambda 和您自己的 Lambda 函數轉換資料的詳細資訊，請參閱「教學課程：使用 S3 Object Lambda 轉換應用程式的資料」。

教學課程：使用 Amazon S3、Amazon 和 Amazon Route 53 託管隨選串流視訊 CloudFront

您可以 CloudFront 將 Amazon S3 與 Amazon 搭配使用，以安全且可擴展的方式託管視訊以供隨選檢視。隨需影片 (VOD) 串流意指您的影片內容會存放在伺服器上，瀏覽者隨時都可以觀看。

CloudFront 是一種快速、高度安全且可程式化的內容傳遞網路 (CDN) 服務。 CloudFront 可以從全球所有 CloudFront 邊緣位置透過 HTTPS 安全地傳遞您的內容。如需詳細資訊 CloudFront，請參閱什麼是 Amazon CloudFront？ 在 Amazon 開 CloudFront 發人員指南。

CloudFront 快取會減少原始伺服器必須直接回應的要求數目。當檢視者 (一般使用者) 要求您提供服務的視訊時 CloudFront，會將要求路由到離檢視者所在位置較近的附近節點。 CloudFront 從其緩存中提供視頻，只有在尚未緩存的情況下才從 S3 存儲桶中檢索它。如此一來，此快取管理功能可透過低延遲、高輸送量和高傳輸速度，加快您的影片交付給全球瀏覽者的速度。如需 CloudFront 快取管理的詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的最佳化快取和可用性。



目標

在本教學中，您將 S3 儲存貯體設定為託管用 CloudFront 於交付的隨選視訊串流，使用 Amazon Route 53 進行網域名稱系統 (DNS) 和自訂網域管理。

主題

先決條件：使用 Route 53 註冊並設定自訂網域

在開始本教學課程之前，您必須使用 Route 53 註冊並設定自訂網域 (例如**example.com**)，以便稍後可以將 CloudFront 分發設定為使用自訂網域名稱。

如果沒有自訂網域名稱，您的 S3 影片就可以公開存取，並透過 CloudFront 類似下列的 URL 進行託管：

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

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

CloudFront 在您將 CloudFront 分發設定為使用 Route 53 設定的自訂網域名稱之後，您的 S3 影片可以公開存取，並透過類似下列的 URL 進行託管：

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

例如 **https://www.example.com/sample.mp4**。自訂網域名稱對瀏覽者而言更簡單，也更直觀。

若要註冊自訂網域，請參閱《Amazon Route 53 開發人員指南》中的使用 Route 53 註冊新網域。

當您使用 Route 53 註冊網域名稱時，Route 53 會為您建立託管區域，而您將在本教學課程稍後使用該區域。您可以在此託管區域存放有關如何將網域流量路由到 Amazon EC2 執行個體或 CloudFront 分發的資訊。

網域註冊、託管區域以及您網域收到的 DNS 查詢會有相關費用。如需詳細資訊，請參閱 Amazon Route 53 定價。

> **Note**
>
> 當您註冊網域時，會立即支付費用，而且無法復原。您可以選擇不自動續約網域，但您已預先付款且該年度擁有該網域。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》中的註冊新網域。

步驟 1：建立 S3 儲存貯體

建立儲存貯體來存放您計劃串流的原始影片。

建立儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇 Create bucket (建立儲存貯體)。

   Create bucket (建立儲存貯體) 頁面隨即開啟。
4. 針對 Bucket name (儲存貯體名稱)，輸入儲存貯體的名稱 (例如 **`tutorial-bucket`**)。

   如需有關在 Amazon S3 中的命名儲存貯體的詳細資訊，請參閱 儲存貯體命名規則。
5. 在「區域」中，選擇您 AWS 區域 要儲存貯體的位置。

   如果可能的話，您應該挑選最接近大多數瀏覽者的區域。如需有關儲存貯體區域的詳細資訊，請參閱 儲存貯體概觀 。
6. 針對 Block Public Access settings for this bucket (此儲存貯體的封鎖公開存取設定)，將保留預設設定 (已啟用封鎖所有公開存取)。

   即使啟用了阻止所有公共訪問，觀眾仍然可以通過訪問上傳的視頻 CloudFront。此功能是用 CloudFront 來託管存放在 S3 中的視訊的主要優勢。

   除非您需要針對使用案例關閉一或多個設定，否則建議您將所有設定保持啟用狀態。如需封鎖公開存取的詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。
7. 對於其他設定，請保留預設值。

   (選用) 如果您想要為您的特定使用案例設定其他儲存貯體設定，請參閱 建立儲存貯體。
8. 選擇建立儲存貯體。

步驟 2：將影片上傳至 S3 儲存貯體

下列程序說明如何使用主控台將影片檔案上傳至 S3 儲存貯體。如果您正在上傳許多大型影片到 S3，您會想要使用 Amazon S3 Transfer Acceleration 來設定快速且安全的檔案傳輸。傳輸加速可以加快影片上傳到 S3 儲存貯體的速度，以便長途傳輸較大的影片。如需詳細資訊，請參閱 使用 Amazon S3 Transfer Acceleration 設定快速安全的檔案傳輸。

將檔案上傳至儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇您在步驟 1 中建立的且要將檔案上傳至的儲存貯體的名稱 (例如，**`tutorial-bucket`**)。
4. 在儲存貯體的物件索引標籤上，選擇上傳。
5. 在 Upload (上傳) 頁面上的 Files and folders (檔案和資料夾) 下，選擇 Add files (新增檔案)。
6. 選擇要上傳的檔案，然後選擇 Open (開啟)。

   例如，您可以上傳名為 `sample.mp4` 的影片檔案。
7. 選擇上傳。

步驟 3：建立 CloudFront 原始存取身分

若要限制從 S3 儲存貯體直接存取影片，請建立名為來源存取身分 (OAI) 的特殊 CloudFront 使用者。您稍後將在本教學課程中將 OAI 與您的分佈建立關聯。透過使用 OAI，您可以確保觀眾無法略過 CloudFront 並直接從 S3 儲存貯體取得影片。只有 CloudFront OAI 可以存取 S3 儲存貯體中的檔案。如需詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的使用 OAI 限制對 Amazon S3 內容的存取。

若要建立 CloudFront OAI

1. 登入 AWS Management Console 並開啟 CloudFront 主控台，位於https://console.aws.amazon.com/cloudfront/v4/home。
2. 在左側導覽窗格中的安全區段下，選擇原始存取。
3. 在身分索引標籤底下，選擇建立原始存取身分。
4. 輸入新原始存取身分的名稱 (例如，**`S3-OAI`**)。
5. 選擇建立。

步驟 4：建立 CloudFront分發

若 CloudFront 要用於在 S3 儲存貯體中提供和散發影片，您必須建立 CloudFront 分發。

子步驟

- 創建一個 CloudFront分佈
- 檢閱儲存貯體政策

創建一個 CloudFront分佈

1. 登入 AWS Management Console 並開啟 CloudFront 主控台，位於https://console.aws.amazon.com/cloudfront/v4/home。
2. 在左側導覽窗格中，選擇 Distributions (分佈)。
3. 選擇 Create Distribution (建立分佈)。
4. 在 Origin (來源) 區段中，針對 Origin domain (原始網域)，選擇 S3 的網域名稱，其中該名稱會以您在 步驟 1 中建立的 S3 儲存貯體的名稱為開頭 (例如，**tutorial-bucket**)。
5. 對於原始存取，選擇舊版存取身分。
6. 在 Origin access identity (原始存取身分) 下，選擇您在步驟 3 中建立的原始存取身分 (例如，**S3-OAI**)。
7. 在 Bucket policy (儲存貯體政策) 下，選擇 Yes, update the bucket policy (是，更新儲存貯體政策)。
8. 在 Default cache behavior (預設快取行為) 區段中，Viewer protocol policy (瀏覽者通訊協定政策) 下方，選擇 Redirect HTTP to HTTPS (從 HTTP 重新引導到 HTTPS)。

   當您選擇此功能，HTTP 請求會自動重新引導至 HTTPS，以確保您網站的安全並保護瀏覽者的資料。
9. 針對 Default cache behaviors (預設快取行為) 區段的其他設定，保留預設值。

   (選擇性) 您可以控制檔案在 CloudFront 快取中保留的時間長度，然後再將另一個要求 CloudFront 轉寄至您的來源。降低持續時間允許您提供動態內容。增加持續時間表示您的瀏覽者取得更好的效能，因為檔案更有可能是直接透過節點快取提供。較長的持續時間也能減少的原始伺服器的負載。如需詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的管理內容在快取中停留的時間 (到期)。
10. 針對其他區段，請將其餘設定保持為預設值。

    如需有關不同設定選項的詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的建立或更新分發時指定的值。
11. 在頁面底部，選擇 Create distribution (建立分佈)。
12. 在散發的 [一般] 索引標籤上的 [詳細資料] 底下， CloudFront 發佈的 [上次修改] 資料行的值會從 [部署] 變更為上次修改發佈時的時間戳記。通常此程序需要幾分鐘的時間。

檢閱儲存貯體政策

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在「值區」清單中，選擇您先前用來作為 CloudFront 分配來源的值區名稱 (例如，**tutorial-bucket**)。
4. 選擇許可索引標籤標籤。
5. 在 Bucket policy (儲存貯體政策) 區段中，確認您在儲存貯體政策文字看到了類似於下列的陳述式：

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

這是當您選擇 「是，請稍早更新儲存貯體政策」時，您的 CloudFront 分配新增至儲存貯體政策的聲明。

此儲存貯體政策更新表示您已成功設定 CloudFront 分發，以限制對 S3 儲存貯體的存取。由於此限制，值區中的物件只能透過您的 CloudFront發行版存取。

步驟 5：通過 CloudFront分發訪問視頻

現在， CloudFront 可以提供存儲在 S3 存儲桶中的視頻。若要透過存取視訊 CloudFront，您必須將 CloudFront 分發網域名稱與 S3 儲存貯體中視訊的路徑結合在一起。

使用 CloudFront 分發網域名稱建立 S3 影片的 URL

1. 登入 AWS Management Console 並開啟 CloudFront 主控台，位於https://console.aws.amazon.com/cloudfront/v4/home。
2. 在左側導覽窗格中，選擇 Distributions (分佈)。
3. 若要取得分佈網域名稱，請執行下列動作：
   a. 在 Ori gins 資料行中，尋找其來源名稱以您在步驟 1 中建立的 S3 儲存貯體開頭 (例如**tutorial-bucket**)，以尋找正確的 CloudFront分佈。
   b. 在清單中找到發行版之後，請擴大 [網域名稱] 欄，以複製您 CloudFront分發的網域名稱值。
4. 在新的瀏覽器標籤中，貼上您複製的分佈網域名稱。
5. 返回上一個瀏覽器標籤，並開啟位於 https://console.aws.amazon.com/s3/ 的 S3 主控台。
6. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
7. 在 Buckets (儲存貯體) 清單中，選擇您在步驟 1 中建立的儲存貯體名稱 (例如，**tutorial-bucket**)。
8. 在 Objects (物件) 清單中，選擇您在步驟 2 中上傳的影片名稱 (例如，`sample.mp4`)。
9. 在物件詳細資料頁面上，Object overview (物件概觀) 區段中，複製 Key (索引鍵) 值。此值是 S3 儲存貯體中上傳影片物件的路徑。
10. 返回您先前貼上分佈網域名稱的瀏覽器索引標籤，輸入斜線 (**/**)，然後貼上您先前複製的影片路徑 (例如，`sample.mp4`)。

    現在，您的 S3 影片可公開存取，並透過 CloudFront 類似下列的 URL 託管：

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

    將*CloudFront ######*和*S3 ########*適當的值。範例 URL 為 **https://d111111abcdef8.cloudfront.net/sample.mp4**。

步驟 6：設定您的 CloudFront 分發以使用您的自訂網域名稱

若要使用您自己的網域名稱，而不是 URL 中的 CloudFront 網域名稱來存取 S3 影片，請在您的 CloudFront 分發中新增替代網域名稱。

子步驟

- 請求 SSL 憑證

- 將替代域名添加到您的 CloudFront 分發中
- 建立 DNS 記錄，將流量從您的備用網域名稱路由到您的 CloudFront 分發網域名稱
- 檢查您的分佈是否已啟用 IPv6，並視需要建立另一個 DNS 記錄

請求 SSL 憑證

若要允許檢視者在影片串流的 URL 中使用 HTTPS 和您的自訂網域名稱，請使用 AWS Certificate Manager (ACM) 要求安全通訊端層 (SSL) 憑證。SSL 憑證會建立與網站之間的加密網路連線。

1. 登入 AWS Management Console 並開啟 ACM 主控台，網址為 https://console.aws.amazon.com/acm/。
2. 如果出現簡介頁面，請在 Provision certificates (佈建憑證) 中，選擇 Get Started (入門)。
3. 在 Request a certificate (請求憑證) 頁面上，選擇 Request a public certificate (請求公有憑證)，然後選擇 Request a certificate (請求憑證)。
4. 在 Add domain names (新增網域名稱) 頁面，輸入要使用 SSL/TLS 憑證保護的網站的完整網域名稱 (FQDN)。您可以使用星號 (*) 請求萬用字元憑證，以保護相同網域中的數個網站名稱。在本教學課程中，輸入 *****，以及您在先決條件中設定的自訂網域名稱。例如，輸入 ***.example.com**，然後選擇 Next (下一步)。

   如需詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的請求 ACM 公有憑證 (主控台)。
5. 在 Select validation method (選取驗證方法) 頁面上，選擇 DNS validation (DNS 驗證)。然後選擇下一步。

   如果您能編輯 DNS 組態，我們建議您使用 DNS 網域驗證，而不使用電子郵件驗證。與電子郵件驗證相比，DNS 驗證有多個優點。如需詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的選項 1：DNS 驗證。
6. (選用) 在 Add tags (新增標籤) 頁面上，使用中繼資料標記您的憑證。
7. 選擇檢閱。
8. 在 Review (檢閱) 頁面上，驗證 Domain name (網域名稱) 和 Validation method (驗證方法) 下面的資訊正確無誤。然後，選擇 Confirm and request (確認和請求)。

   Validation (驗證) 頁面會顯示您的請求正在處理中，且憑證網域正在進行驗證。等待驗證的憑證會處於 Pending validation (待定驗證) 狀態。
9. 在 Validation (驗證) 頁面上，選擇自訂網域名稱左側的向下箭頭，然後選擇 Create record in Route 53 (在 Route 53 中建立記錄) 以透過 DNS 驗證您的網域所有權。

這樣做會將提供的 CNAME 記錄新增 AWS Certificate Manager 至您的 DNS 組態。

10. 在 Create record in Route 53 (在 Route 53 中建立記錄) 對話方塊中，選擇 Create (建立)。

    Validation (驗證) 頁面應該會在底部顯示 Success (成功) 狀態通知。

11. 選擇 Continue (繼續) 以檢視 Certificates (憑證) 清單頁面。

    您的新憑證的 Status (狀態) 會在 30 分鐘內從 Pending validation (待定驗證) 變更為 Issued (已發行)。

將替代域名添加到您的 CloudFront 分發中

1. 登入 AWS Management Console 並開啟 CloudFront 主控台，位於https://console.aws.amazon.com/cloudfront/v4/home。
2. 在左側導覽窗格中，選擇 Distributions (分佈)。
3. 為您在步驟 4 中建立的分佈選擇 ID。
4. 在General (一般) 索引標籤上，移至 Settings (設定) 區段，然後選擇 Edit (編輯)。
5. 在 [編輯設定] 頁面上，對於替代網域名稱 (CNAME)-選用，選擇 [新增項目] 以新增您要在此 CloudFront 發行版所提供之 S3 影片的 URL 中使用的自訂網域名稱。

   例如，在此教學課程中，如果您想要路由子網域的流量，例如 `www.example.com`，輸入含有網域名稱 (`example.com`) 的子網域名稱 (`www`)。具體而言，輸入 **`www.example.com`**。

   > **Note**
   >
   > 您新增的替代網域名稱 (CNAME) 必須涵蓋您先前附加到 CloudFront發行版本的 SSL 憑證。

6. 針對 Custom SSL certificate - optional (自訂 SSL 憑證 - 選用)，選擇您在之前請求的 SSL 憑證 (例如，**`*.example.com`**)。

   > **Note**
   >
   > 如果您在請求之後沒有立即看到 SSL 憑證，請等待 30 分鐘，然後重新整理清單，直到 SSL 憑證可供您選取為止。

7. 將其餘設定保持為預設值。選擇儲存變更。

8. 在分佈的 General (一般) 標籤上，等待 Last modified (上次修改日期) 的值從 Deploying (正在部署) 變更為上次修改分佈的時間戳記。

建立 DNS 記錄，將流量從您的備用網域名稱路由到您的 CloudFront 分發網域名稱

1. 登入 AWS Management Console 並開啟路線 53 主控台，網址為 https://console.aws.amazon.com/route53/。
2. 在左側導覽窗格中，選擇 Hosted zones (託管區域)。
3. 在 Hosted zones (託管區域) 頁面上，選擇 Route 53 在先決條件中為您建立的託管區域名稱 (例如，**example.com**)。
4. 選擇 Create record (建立記錄)，然後再使用 Quick create record (快速建立記錄) 方法。
5. 對於 [記錄名稱]，請保持記錄名稱的值與您先前新增之 CloudFront 發行版的替代網域名稱相同。

   在本教學課程中，若要將流量路由至子網域 (例如 `www.example.com`)，請輸入不含網域名稱的子網域名稱。例如，僅在您自訂網域名稱之前的文字欄位中輸入 **www**。
6. 在 [記錄類型] 中，選擇 [A]-將流量路由至 IPv4 位址和某些 AWS 資源。
7. 針對 Value (值)，選擇 Alias (別名) 切換，以啟用別名資源。
8. 在路由流量到，從下拉列表中選擇要 CloudFront 分發的別名。
9. 在顯示「選擇分發」的搜索框中，選擇您在步驟 4 中創建的 CloudFront 分發的域名。

   要查找分 CloudFront 發的域名，請執行以下操作：

   a. 在新的瀏覽器索引標籤中，登入 AWS Management Console 並在開啟 CloudFront 主控台https://console.aws.amazon.com/cloudfront/v3/home。
   b. 在左側導覽窗格中，選擇 Distributions (分佈)。
   c. 在 Ori gins 資料行中，尋找其來源名稱以您在步驟 1 中建立的 S3 儲存貯體開頭 (例如**tutorial-bucket**)，以尋找正確的 CloudFront分佈。
   d. 在清單中找到發行版之後，請擴大 [網域名稱] 欄，以查看您 CloudFront 分發的網域名稱值。
10. 在 Route 53 主控台的 Create record (建立記錄) 頁面上，針對剩餘的設定，請保留預設值。
11. 選擇建立記錄。

檢查您的分佈是否已啟用 IPv6，並視需要建立另一個 DNS 記錄

如果您的分佈已啟用 IPv6，您必須建立另一個 DNS 記錄。

1. 若要檢查您的分佈是否已啟用 IPv6，請執行下列動作：

   a. 登入 AWS Management Console 並開啟 CloudFront 主控台，位於https://console.aws.amazon.com/cloudfront/v4/home。

   b. 在左側導覽窗格中，選擇 Distributions (分佈)。

   c. 選擇您在步驟 4 中建立的 CloudFront 分發 ID。

   d. 在 General (一般) 選項卡上的 Settings (設定) 下，檢查 IPv6 是否設定為 Enabled (已啟用)。

   如果您的分佈已啟用 IPv6，您必須建立另一個 DNS 記錄。

2. 如果您的分佈已啟用 IPv6，請執行以下動作建立一個 DNS 記錄：

   a. 登入 AWS Management Console 並開啟路線 53 主控台，網址為 https://console.aws.amazon.com/route53/。

   b. 在左側導覽窗格中，選擇 Hosted zones (託管區域)。

   c. 在 Hosted zones (託管區域) 頁面上，選擇 Route 53 在先決條件中為您建立的託管區域名稱 (例如，**example.com**)。

   d. 選擇 Create record (建立記錄)，然後再使用 Quick create record (快速建立記錄) 方法。

   e. 針對 Record name (記錄名稱)，在自訂網域名稱前面的文字欄位中，輸入您先前建立 IPv4 DNS 記錄時所輸入的相同數值。例如，在本教學課程中，要為子網域 `www.example.com` 路由流量，僅須輸入 **www**。

   f. 針對 Record type (記錄類型)，選擇 AAAA - Routes traffic to an IPv6 address and some AWS resources (AAAA – 將流量路由至 IPv6 地址和某些 AWS 資源)。

   g. 針對 Value (值)，選擇 Alias (別名) 切換，以啟用別名資源。

   h. 在路由流量到，從下拉列表中選擇要 CloudFront 分發的別名。

   i. 在顯示「選擇分發」的搜索框中，選擇您在步驟 4 中創建的 CloudFront分發的域名。

   j. 對於其他設定，請保留預設值。

   k. 選擇建立記錄。

步驟 7：使用自定義域名通過 CloudFront 分發訪問 S3 視頻

要使用自訂 URL 存取 S3 影片，您必須將您的備用網域名稱與 S3 儲存貯體中的影片路徑合併。

若要建立透過 CloudFront 分發存取 S3 影片的自訂 URL

1. 登入 AWS Management Console 並開啟 CloudFront 主控台，位於https://console.aws.amazon.com/cloudfront/v4/home。
2. 在左側導覽窗格中，選擇 Distributions (分佈)。
3. 要獲取 CloudFront分發的替代域名，請執行以下操作：
   a. 在 Ori gins 資料行中，尋找正確的 CloudFront分佈，尋找其來源名稱，該名稱以您在步驟 1 中建立的儲存貯體的 S3 儲存貯體名稱開頭 (例如**`tutorial-bucket`**)。
   b. 在清單中找到發行版本後，請擴大 [替代網域名稱] 欄，以複製您 CloudFront 分發的替代網域名稱的值。
4. 在新的瀏覽器選項卡中，粘貼 CloudFront分發的替代域名。
5. 返回上一個瀏覽器標籤，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
6. 按照 步驟 5 的解釋說明，查找 S3 影片的路徑。
7. 返回您先前貼上備用網域名稱的瀏覽器索引標籤，輸入斜線 (**/**)，然後貼上 S3 影片的路徑 (例如，`sample.mp4`)。

   現在，您的 S3 影片可公開存取，並透過 CloudFront 類似下列的自訂 URL 託管：

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

   *`#CloudFront ########## S3 ####`*取代為適當的值。範例 URL 為 **`https://www.example.com/sample.mp4`**。

(選擇性) 步驟 8：檢視 CloudFront 分發所收到之請求的相關資料

若要檢視您的 CloudFront發佈所收到請求的相關資料

1. 登入 AWS Management Console 並開啟 CloudFront 主控台，位於https://console.aws.amazon.com/cloudfront/v4/home。
2. 在左側導覽窗格中，在 Reports & analytics (報告與分析) 下，從主控台選擇報告，其中範圍包括 Cache statistics (快取統計資料)、Popular Objects (熱門物件)、Top Referrers (最佳推薦網站)、Usage (用量) 和 Viewers (瀏覽者)。

您可以篩選每個報告儀表板。如需詳細資訊，請參閱 Amazon CloudFront 開發人員指南中主控台中的CloudFront 報告。

3. 若要篩選資料，請選擇您在步驟 4 中建立的 CloudFront 分佈 ID。

步驟 9：清除

如果您僅使用 CloudFront Route 53 作為學習練習託管 S3 串流影片，請刪除您分配的 AWS 資源，以免再產生費用。

> **Note**
>
> 當您註冊網域時，會立即支付費用，而且無法復原。您可以選擇不自動續約網域，但您已預先付款且該年度擁有該網域。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》中的註冊新網域。

子步驟

- 刪除分 CloudFront配
- 刪除 DNS 記錄
- 針對您的自訂網域刪除公有託管區域
- 從 Route 53 刪除自訂網域名稱
- 刪除 S3 來源儲存貯體中的原始影片
- 刪除 S3 來源儲存貯體

刪除分 CloudFront配

1. 登入 AWS Management Console 並開啟 CloudFront 主控台，位於https://console.aws.amazon.com/cloudfront/v4/home。
2. 在左側導覽窗格中，選擇 Distributions (分佈)。
3. 在 Ori gins 資料行中，尋找正確的 CloudFront分佈，尋找其來源名稱，該名稱以您在步驟 1 中建立的儲存貯體的 S3 儲存貯體名稱開頭 (例如**`tutorial-bucket`**)。
4. 若要刪除 CloudFront 分發，您必須先將其停用。

- 如果 Status (狀態) 欄位的值為 Enabled (已啟用) 且 Last modified (上次修改日期) 的值是上次修改分佈的時間戳記，請先繼續停用分佈，然後再將其刪除。
- 如果 Status (狀態) 的值為 Enabled (已啟用) 且 Last modified (上次修改日期) 的值為 Deploying (正在部署)，請等待直至 Status (狀態) 的值變更為上次修改分佈的時間戳記。然後請先繼續停用分佈，然後再將其刪除。

5. 若要停用 CloudFront 散佈，請執行下列動作：

   a. 在 Distributions (分佈) 清單中，選取要刪除的分佈 ID 旁邊的核取方塊。

   b. 若要停用分佈，選擇 Disable (停用)，然後選擇 Disable (停用) 以進行確認。

      如果您 CloudFront 停用了與其相關聯的替代網域名稱的發行版，即使另一個發行版具有萬用字元 (`www.example.com`) 符合相同網域 (例如) 的替代網域名稱，也會停止接受該網域名稱的流量 (例如`*.example.com`)。*

   c. Status (狀態) 的值立即變更為 Disabled (已停用)。等待 Last modified (上次修改日期) 的值從 Deploying (正在部署) 變更為上次修改分佈的時間戳記。

      由於 CloudFront 必須將此變更傳播到所有邊位置，因此更新完成之前可能需要幾分鐘的時間，而且您可以使用「刪除」(Delete) 選項來刪除分佈。

6. 若要刪除已停用的分佈，請執行下列動作：

   a. 選擇要刪除的分佈 ID 旁邊的核取方塊。

   b. 選擇 Delete (刪除)，然後選擇 Delete (刪除) 以進行確認。

刪除 DNS 記錄

如果您想要刪除網域的公有託管區域 (包括 DNS 記錄)，請參閱《Amazon Route 53 開發人員指南》中的 針對您的自訂網域刪除公有託管區域。如果您只想刪除在步驟 6 中建立的 DNS 記錄，請執行下列動作：

1. 登入 AWS Management Console 並開啟路線 53 主控台，網址為 https://console.aws.amazon.com/route53/。
2. 在左側導覽窗格中，選擇 Hosted zones (託管區域)。
3. 在 Hosted zones (託管區域) 頁面上，選擇 Route 53 在先決條件中為您建立的託管區域名稱 (例如，**`example.com`**)。
4. 在記錄清單中，選取要刪除的記錄旁邊的核取方塊 (您在步驟 6 中建立的記錄)。

> **Note**
>
> 您無法刪除 Type (類型) 為 NS 或 SOA 值的記錄。

5. 選擇 Delete records (刪除記錄)。
6. 如要確認刪除，請選擇 Delete (刪除)。

對記錄的變更需要一些時間傳播到 Route 53 DNS 伺服器。目前，驗證變更是否已傳播的唯一方法是使用 GetChange API 動作。變更通常會在 60 秒內傳播至所有 Route 53 名稱伺服器。

針對您的自訂網域刪除公有託管區域

> **Warning**
>
> 如果您想要保留您的網域註冊，但停止將網際網路流量路由到您的網站或 Web 應用程式，則建議您刪除託管區域中的記錄(如前一個區段所述)，而不是刪除託管區域。
> 如果您刪除託管區域，有心人士可能會使用網域，並使用您的網域名稱將流量路由到他們自己的資源。
> 此外，如果您刪除託管區域，就無法取消刪除。您必須建立新的託管區域，並更新您網域註冊的名稱伺服器；這最多需要 48 小時才能生效。
> 如果您想要讓網域無法在網際網路上使用，您可以先將您的 DNS 服務傳輸至免費的 DNS 服務，然後刪除 Route 53 託管區域。這可避免往後的 DNS 查詢發生路由錯誤。
>
> 1. 如果此網域已向 Route 53 註冊，請參閱《Amazon Route 53 開發人員指南》中的新增或變更網域的名稱伺服器和黏附記錄，以了解如何將 Route 53 名稱伺服器取代為新 DNS 服務的名稱伺服器。
> 2. 如果此網域已向其他註冊商註冊，請使用註冊商提供的方法變更網域的名稱伺服器。
>
> > **Note**
> >
> > 如果您刪除子網域 (`www.example.com`) 的託管區域，則不需要變更網域 (`example.com`) 的名稱伺服器。

1. 登入 AWS Management Console 並開啟路線 53 主控台，網址為 https://console.aws.amazon.com/route53/。
2. 在左側導覽窗格中，選擇 Hosted zones (託管區域)。
3. 在 Hosted zones (託管區域) 頁面上，選擇要刪除的託管區域的名稱。
4. 在您託管區域的 Records (記錄) 標籤上，確認要刪除的託管區域只包含 NS 和 SOA 紀錄。

   如果包含其他記錄，請先刪除它們。

   如果您在託管區域中為子網域建立任何 NS 記錄，也請刪除這些記錄。
5. 在您託管區域的 DNSSEC signing (DNSSEC 簽署) 標籤上，請停用 DNSSEC 簽署 (如果已啟用)。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》中的停用 DNSSEC 簽署。
6. 在託管區域的詳細資訊頁面頂端，選擇 Delete zone (刪除區域)。
7. 若要確認刪除，輸入 **delete**，然後選擇 Delete (刪除)。

從 Route 53 刪除自訂網域名稱

對於大多數頂層網域 (TLD)，可在不再需要網域時刪除註冊。如果您在註冊計劃到期前從 Route 53 刪除網域名稱註冊，註冊費 AWS 不會退還。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》中的刪除網域名稱註冊。

> **Important**
>
> 如果您想要在網域之間轉移 AWS 帳戶 或將網域轉移到其他註冊商，請勿刪除該網域，並希望立即重新註冊該網域。反之，請參閱《Amazon Route 53 開發人員指南》中的適用文件：
>
> - 將網域轉移到其他網域 AWS 帳戶
> - 將網域從 Amazon Route 53 轉移到其他註冊商

刪除 S3 來源儲存貯體中的原始影片

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Bucket name (儲存貯體名稱) 清單中，選擇您在步驟 2 中將影片上傳到的儲存貯體的名稱 (例如，**tutorial-bucket**)。

4. 在 Objects (物件) 索引標籤上，選取要刪除之物件名稱左側的核取方塊 (例如，`sample.mp4`)。
5. 選擇刪除。
6. 在 Permanently delete objects? (要永久刪除物件？) 下，輸入 **`permanently delete`** 以確認要刪除此物件。
7. 選擇 Delete objects (刪除物件)。

刪除 S3 來源儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選取您在步驟 1 中建立的儲存貯體名稱旁的選項按鈕 (例如，**`tutorial-bucket`**)。
4. 選擇刪除。
5. 在 Delete bucket (刪除儲存貯體) 頁面上，在文字欄位中輸入儲存貯體名稱以確認您要刪除該儲存貯體，然後選擇 Delete bucket (刪除儲存貯體)。

後續步驟

完成本教學課程後，您可以進一步探索下列相關的使用案例：

- 將 S3 視頻轉碼為特定電視或連接設備所需的流格式，然後再使用 CloudFront 分發託管這些視頻。

  若要使用 Amazon S3 Batch 操作， AWS Elemental MediaConvert 以 AWS Lambda 及將影片集合批次轉碼為各種輸出媒體格式，請參閱。教學課程：使用 S3 Batch 操作進行 Batch 轉碼影片，以及 AWS LambdaAWS Elemental MediaConvert
- 使用和 Route 53 託管存儲在 S3 中的其他對象，例如圖像，音頻 JavaScript，動態圖形，樣式表，HTML，React 應用程序 CloudFront 等。

  如需範例，請參閱 教學課程：使用向 Route 53 註冊的自訂網域設定靜態網站 和 加快您的網站與 Amazon CloudFront。
- 使用 Amazon S3 Transfer Acceleration 來設定快速且安全的檔案傳輸。傳輸加速可以加快影片上傳到 S3 儲存貯體的速度，以便長途傳輸較大的影片。傳輸加速會將流量路由到 CloudFront 全球分散的邊緣位置和 AWS 骨幹網路，藉此改善傳輸效能。它還使用了網路通訊協定最佳化。如需詳細資訊，請參閱 使用 Amazon S3 Transfer Acceleration 設定快速安全的檔案傳輸。

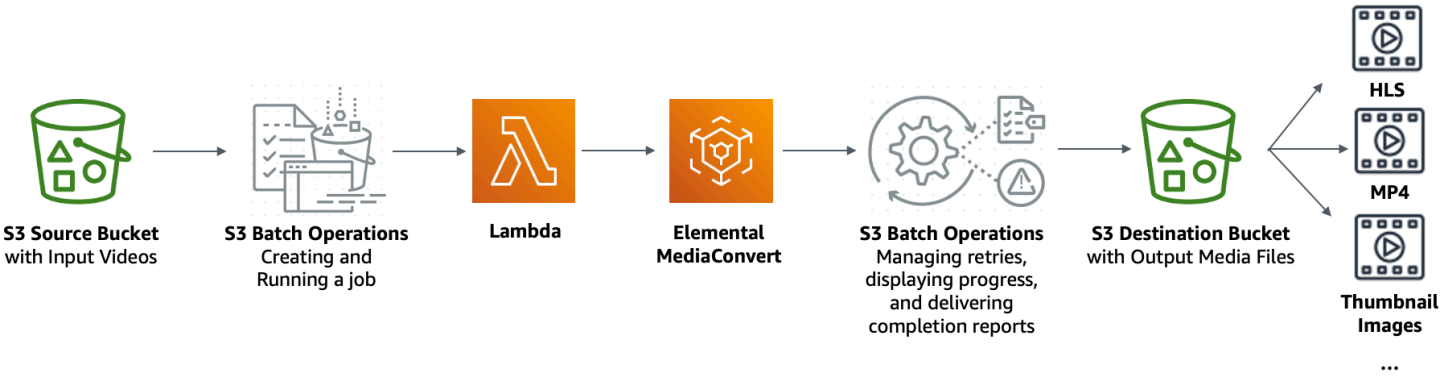
教學課程：使用 S3 Batch 操作進行 Batch 轉碼影片，以及 AWS LambdaAWS Elemental MediaConvert

影片消費者使用各種類型、尺寸和年份的裝置來觀賞媒體內容。各式各樣的裝置對內容創作者和發行者來說是一項挑戰。視頻必須轉換，而不是 one-size-fits-all 格式，以便它們可以跨越廣泛的大小，格式和比特率。當您有大量必須轉換的影片時，這項轉換任務更具挑戰性。

AWS 為您提供了一種構建可擴展的分佈式架構的方法，該架構可執行以下操作：

- 擷取輸入影片
- 處理影片，以在各種裝置上播放
- 存放轉碼後的媒體檔案
- 交付輸出媒體檔案，以滿足需求

當您在 Amazon S3 中存放了大量的影片儲存庫時，您可以將這些影片從其來源格式轉碼為特定影片播放器或裝置所需的大小、解析度和格式各異的多種檔案類型。具體來說，S3 Batch 操作提供了一個解決方案，可為 S3 來源儲存貯體中的現有輸入影片叫用 AWS Lambda 函數。然後，Lambda 函數會呼叫 AWS Elemental MediaConvert 以執行大規模的影片轉碼任務。轉換後的輸出媒體檔案會存放在 S3 目的地儲存貯體中。



目標

在本教學課程中，您將學習如何設定 S3 批次操作來叫用 Lambda 函數，以對存放在 S3 來源儲存貯體中的影片進行批次轉碼。Lambda 函數會呼叫 MediaConvert 將視訊轉碼。S3 來源儲存貯體中的每個影片輸出如下：

- HTTP 即時串流 (HLS) 彈性位元速率串流，可在多個大小的裝置上播放，並可適應不同的頻寬
- MP4 影片檔案

- 每隔一段時間收集的縮圖影像

主題

必要條件

在您開始此教學課程之前，您必須有 Amazon S3 來源儲存貯體 (例如 **`tutorial-bucket-1`**），並且其中已存放了要轉碼的影片。

如果您想要，可以為儲存貯體命名另一個名稱。如需 Amazon S3 中儲存貯體名稱的詳細資訊，請參閱儲存貯體命名規則。

對於 S3 來源儲存貯體，請將與 Block Public Access settings for this bucket (此儲存貯體的封鎖公開存取設定) 相關的設定保留為預設值 (已啟用 Block all public access (封鎖所有公開存取))。如需詳細資訊，請參閱 建立儲存貯體。

如需有關將影片上傳至 S3 來源儲存貯體的詳細資訊，請參閱 上傳物件。如果您正在上傳許多大型影片到 S3，您會想要使用 Amazon S3 Transfer Acceleration 來設定快速且安全的檔案傳輸。傳輸加速可以加快影片上傳到 S3 儲存貯體的速度，以便長途傳輸較大的影片。如需詳細資訊，請參閱 使用 Amazon S3 Transfer Acceleration 設定快速安全的檔案傳輸。

步驟 1：為您的輸出媒體檔案建立 S3 儲存貯體

在此步驟中，您要建立 S3 目的地儲存貯體，以存放轉換後的輸出媒體檔案。您也可以建立一個跨來源資源共享 (CORS) 組態，以允許跨來源存取存放在 S3 目的地儲存貯體中的轉碼媒體檔案。

子步驟

- 為您的輸出媒體檔案建立儲存貯體
- 將 CORS 組態新增至 S3 輸出儲存貯體

為您的輸出媒體檔案建立儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇建立儲存貯體。
4. 針對 Bucket name (儲存貯體名稱)，輸入儲存貯體的名稱 (例如 **`tutorial-bucket-2`**)。
5. 在「區域」中，選擇您 AWS 區域 要儲存貯體的位置。
6. 若要確保公開存取您的輸出媒體檔案，請在 Block Public Access settings for this bucket (此儲存貯體的封鎖公開存取設定) 中，清除 Block all public access (封鎖所有公開存取)。

   > **Warning**
   >
   > 在完成此步驟之前，請檢閱 封鎖對 Amazon S3 儲存體的公開存取權 以確保您了解並接受允許公開存取所涉及的風險。當您關閉 Block Public Access (封鎖公開存取) 設定以公開儲存貯體時，網際網路上的任何人都可以存取您的儲存貯體。我們建議您封鎖對儲存貯體的所有公用存取權。
   >
   > 如果您不想清除「封鎖公開存取」設定，可以使用 Amazon CloudFront 將轉碼的媒體檔案交付給檢視者 (最終使用者)。如需詳細資訊，請參閱 教學課程：使用 Amazon S3、Amazon 和 Amazon Route 53 託管隨選串流視訊 CloudFront。

7. 選取 I acknowledge that the current settings might result in this bucket and the objects within becoming public (我確認目前的設定可能會導致此儲存貯體及其中的物件變為公開) 旁的核取方塊。
8. 將其餘設定保持為預設值。
9. 選擇建立儲存貯體。

將 CORS 組態新增至 S3 輸出儲存貯體

JSON CORS 組態會定義一種方式，讓單一網域載入的用戶端 Web 應用程式 (即影片播放器)，能播放不同網域中的轉碼後的輸出媒體檔案。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇您先前建立的儲存貯體名稱 (例如，**`tutorial-bucket-2`**)。
4. 選擇許可索引標籤標籤。
5. 在 Cross-origin resource sharing (CORS) (跨來源資源分享 (CORS)) 區段中，選擇 Edit (編輯)。
6. 在 CORS 組態文字方塊中，複製並貼上下列 CORS 組態。

   CORS 組態必須是 JSON 格式。在此範例中，`AllowedOrigins` 屬性使用萬用字元 (*) 來指定所有來源伺服器。如果您知道您的特定來源，則可以將 `AllowedOrigins` 屬性限制為您的特定播放器 URL。如需有關設定此屬性和其他屬性的詳細資訊，請參閱 CORS 組態。

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

       }
   ]
   ```

7. 選擇儲存變更。

步驟 2：為下列項目建立 IAM 角色 MediaConvert

若 AWS Elemental MediaConvert 要使用對存放在 S3 儲存貯體中的輸入影片進行轉碼，您必須具有 AWS Identity and Access Management (IAM) 服務角色，才能授與從 S3 來源和目標儲存貯體讀取和寫入影片檔案的 MediaConvert 許可。當您執行轉碼工作時， MediaConvert 主控台會使用此角色。

若要為以下項目建立 IAM 角色 MediaConvert

1. 使用您選擇的角色名稱建立 IAM 角色 (例如，**tutorial-mediaconvert-role**)。若要建立此角色，請按照AWS Elemental MediaConvert 使用者指南中的在 IAM (主控台) 中建立 MediaConvert 角色中的步驟進行操作。
2. 建立 IAM 角色之後 MediaConvert，在 [角色] 清單中，選擇您為 MediaConvert 其建立的角色名稱 (例如**tutorial-mediaconvert-role**)。
3. 在 Summary (摘要) 頁面上，複製 Role ARN (角色 ARN) (以 `arn:aws:iam::` 為開頭)，並儲存 ARN 以供稍後使用。

   如需 ARN 的詳細資訊，請參閱《AWS 一般參考》中的 Amazon Resource Name (ARN)。

步驟 3：為您的 Lambda 函數建立 IAM 角色

若要使用 MediaConvert 和 S3 Batch 操作對影片進行批次轉碼，您可以使用 Lambda 函數連接這兩個服務以轉換影片。此 Lambda 函數必須具有 IAM 角色，可授與 Lambda 函數存取權限 MediaConvert 和 S3 Batch 操作的權限。

子步驟

- 為您的 Lambda 函數建立 IAM 角色
- 針對 Lambda 函數的 IAM 角色嵌入內嵌政策

為您的 Lambda 函數建立 IAM 角色

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 選擇 AWS service (AWS 服務) 角色類型，然後在 Common use cases (常用使用案例) 下，選擇 Lambda。
4. 選擇下一步：許可。

5. 在 Attach Permissions policies (連接許可政策) 頁面上，於 Filter policies (篩選政策) 方塊中輸入 **AWSLambdaBasicExecutionRole**。若要將受管政策附加AWSLambdaBasicExecutionRole到此角色以授與寫入權限給 Amazon CloudWatch Logs，請選取旁邊的核取方塊AWSLambdaBasicExecutionRole。
6. 選擇下一步：標籤。
7. (選用) 新增標籤至受管政策。
8. 選擇下一步：檢閱。
9. 在角色名稱中，輸入 **tutorial-lambda-transcode-role**。
10. 選擇建立角色。

針對 Lambda 函數的 IAM 角色嵌入內嵌政策

若要將權限授與執行 Lambda 函數所需的 MediaConvert 資源，您必須使用內嵌政策。

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 在 Roles (角色) 清單中，選擇您在先前為您的 Lambda 函數建立的 IAM 角色的名稱 (例如，**tutorial-lambda-transcode-role**)。
4. 選擇 Permissions (許可) 標籤。
5. 選擇 Add inline policy (新增內嵌政策)。
6. 選擇 JSON 標籤，然後將下列 JSON 政策複製後貼上。

   在 JSON 政策中，將的範例 ARN 值取代為您在步驟 2 中建立的 IAM 角色 MediaConvert 的角色 ARN (例如)。`Resource` **tutorial-mediaconvert-role**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*",
            "Effect": "Allow",
```

```
            "Sid": "Logging"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/tutorial-mediaconvert-role"
            ],
            "Effect": "Allow",
            "Sid": "PassRole"
        },
        {
            "Action": [
                "mediaconvert:*"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "MediaConvertService"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "S3Service"
        }
    ]
}
```

7. 選擇檢閱政策。
8. 對於 Name (名稱)，輸入 **tutorial-lambda-policy**。
9. 選擇建立政策。

   在您建立內嵌政策後，它會自動嵌入您的 Lambda 函數的 IAM 角色中。

步驟 4：為影片轉碼建立 Lambda 函數

在本教學課程的這一節中，您將使用適用於 Python 的開發套件建立 Lambda 函數，以便與 S3 Batch 操作和 MediaConvert. 若要開始轉碼已存放在 S3 來源儲存貯體中的影片，您需要執行 S3 批次操作任務，該任務會直接為 S3 來源儲存貯體中的每個影片叫用 Lambda 函數。然後，Lambda 函數會將每個視訊的轉碼工作提交給。 MediaConvert

子步驟

- 撰寫 Lambda 函數程式碼並建立部署套件
- 使用執行角色 (主控台) 建立 Lambda 函數
- 使用 .zip 檔案封存部署您的 Lambda 函數，並設定 Lambda 函數 (主控台)

撰寫 Lambda 函數程式碼並建立部署套件

1. 在本機電腦上，建立名為 `batch-transcode` 的資料夾。
2. 在 `batch-transcode` 資料夾中，使用 JSON 任務設定建立檔案。例如，您可以使用本區段提供的設定，並將檔案命名為 `job.json`。

   指定下列項目的 `job.json` 檔案：

   - 要轉碼的檔案
   - 您想要如何轉碼輸入影片
   - 您想要建立的輸出媒體檔案
   - 如何命名轉碼後的檔案
   - 轉碼後的檔案的儲存位置
   - 要套用的進階功能等等

   在此教學課程中，我們使用下列 `job.json` 檔案為 S3 來源儲存貯體中的每個影片建立以下輸出：

   - HTTP 即時串流 (HLS) 彈性位元速率串流，可在多個不同大小的裝置上播放，並可適應不同的頻寬
   - MP4 影片檔案
   - 每隔一段時間收集的縮圖影像

此範例 job.json 檔案使用品質定義的可變位元率 (QVBR) 來最佳化影片品質。HLS 輸出符合 Apple 標準 (影片未混合音訊、區段持續時間為 6 秒，以及透過自動 QVBR 優化影片品質)。

如果想要不使用此處提供的範例設定，您可以根據您的使用案例產生 job.json 規格。為了確保輸出的一致性，請確定您的輸入檔案具有類似的影片和音訊組態。針對任何具有不同影片和音訊組態的輸入檔案，建立單獨的自動化 (唯一的 job.json 設定)。如需詳細資訊，請參閱《AWS Elemental MediaConvert 使用者指南》中的 JSON 中的範例 AWS Elemental MediaConvert 任務設定。

```
{
  "OutputGroups": [
    {
      "CustomName": "HLS",
      "Name": "Apple HLS",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "VideoPid": 481,
              "AudioPids": [
                482,
                483,
                484,
                485,
                486,
                487,
                488,
                489,
                490,
                491,
                492
              ]
```

```
            }
          },
          "VideoDescription": {
            "Width": 640,
            "ScalingBehavior": "DEFAULT",
            "Height": 360,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "GopSize": 2,
                "Slices": 1,
                "GopBReference": "DISABLED",
                "MaxBitrate": 1200000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
```

```
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_360"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "TimedMetadataPid": 502,
              "VideoPid": 481,
              "AudioPids": [
                482,
                483,
                484,
                485,
                486,
                487,
                488,
                489,
                490,
                491,
                492
              ]
            }
          },
```

```
            "VideoDescription": {
              "Width": 960,
              "ScalingBehavior": "DEFAULT",
              "Height": 540,
              "TimecodeInsertion": "DISABLED",
              "AntiAlias": "ENABLED",
              "Sharpness": 50,
              "CodecSettings": {
                "Codec": "H_264",
                "H264Settings": {
                  "InterlaceMode": "PROGRESSIVE",
                  "NumberReferenceFrames": 3,
                  "Syntax": "DEFAULT",
                  "Softness": 0,
                  "GopClosedCadence": 1,
                  "GopSize": 2,
                  "Slices": 1,
                  "GopBReference": "DISABLED",
                  "MaxBitrate": 3500000,
                  "SlowPal": "DISABLED",
                  "SpatialAdaptiveQuantization": "ENABLED",
                  "TemporalAdaptiveQuantization": "ENABLED",
                  "FlickerAdaptiveQuantization": "DISABLED",
                  "EntropyEncoding": "CABAC",
                  "FramerateControl": "INITIALIZE_FROM_SOURCE",
                  "RateControlMode": "QVBR",
                  "CodecProfile": "MAIN",
                  "Telecine": "NONE",
                  "MinIInterval": 0,
                  "AdaptiveQuantization": "HIGH",
                  "CodecLevel": "AUTO",
                  "FieldEncoding": "PAFF",
                  "SceneChangeDetect": "TRANSITION_DETECTION",
                  "QualityTuningLevel": "SINGLE_PASS_HQ",
                  "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                  "UnregisteredSeiTimecode": "DISABLED",
                  "GopSizeUnits": "SECONDS",
                  "ParControl": "INITIALIZE_FROM_SOURCE",
                  "NumberBFramesBetweenReferenceFrames": 2,
                  "RepeatPps": "DISABLED"
                }
              },
              "AfdSignaling": "NONE",
              "DropFrameTimecode": "ENABLED",
```

```
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_540"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "VideoPid": 481,
              "AudioPids": [
                482,
                483,
                484,
                485,
                486,
                487,
                488,
                489,
                490,
                491,
                492
              ]
            }
          },
          "VideoDescription": {
            "Width": 1280,
            "ScalingBehavior": "DEFAULT",
```

```
            "Height": 720,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "GopSize": 2,
                "Slices": 1,
                "GopBReference": "DISABLED",
                "MaxBitrate": 5000000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
```

```
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_720"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {}
          },
          "AudioDescriptions": [
            {
              "AudioSourceName": "Audio Selector 1",
              "CodecSettings": {
                "Codec": "AAC",
                "AacSettings": {
                  "Bitrate": 96000,
                  "CodingMode": "CODING_MODE_2_0",
                  "SampleRate": 48000
                }
              }
            }
          ],
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioTrackType": "ALTERNATE_AUDIO_AUTO_SELECT_DEFAULT"
            }
          },
          "NameModifier": "_audio"
        }
      ],
      "OutputGroupSettings": {
        "Type": "HLS_GROUP_SETTINGS",
        "HlsGroupSettings": {
          "ManifestDurationFormat": "INTEGER",
          "SegmentLength": 6,
          "TimedMetadataId3Period": 10,
          "CaptionLanguageSetting": "OMIT",
```

```
          "Destination": "s3://EXAMPLE-BUCKET/HLS/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          },
          "TimedMetadataId3Frame": "PRIV",
          "CodecSpecification": "RFC_4281",
          "OutputSelection": "MANIFESTS_AND_SEGMENTS",
          "ProgramDateTimePeriod": 600,
          "MinSegmentLength": 0,
          "DirectoryStructure": "SINGLE_DIRECTORY",
          "ProgramDateTime": "EXCLUDE",
          "SegmentControl": "SEGMENTED_FILES",
          "ManifestCompression": "NONE",
          "ClientCache": "ENABLED",
          "StreamInfResolution": "INCLUDE"
        }
      }
    },
    {
      "CustomName": "MP4",
      "Name": "File Group",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "MP4",
            "Mp4Settings": {
              "CslgAtom": "INCLUDE",
              "FreeSpaceBox": "EXCLUDE",
              "MoovPlacement": "PROGRESSIVE_DOWNLOAD"
            }
          },
          "VideoDescription": {
            "Width": 1280,
            "ScalingBehavior": "DEFAULT",
            "Height": 720,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 100,
            "CodecSettings": {
              "Codec": "H_264",
```

```
            "H264Settings": {
              "InterlaceMode": "PROGRESSIVE",
              "ParNumerator": 1,
              "NumberReferenceFrames": 3,
              "Syntax": "DEFAULT",
              "Softness": 0,
              "GopClosedCadence": 1,
              "HrdBufferInitialFillPercentage": 90,
              "GopSize": 2,
              "Slices": 2,
              "GopBReference": "ENABLED",
              "HrdBufferSize": 10000000,
              "MaxBitrate": 5000000,
              "ParDenominator": 1,
              "EntropyEncoding": "CABAC",
              "RateControlMode": "QVBR",
              "CodecProfile": "HIGH",
              "MinIInterval": 0,
              "AdaptiveQuantization": "AUTO",
              "CodecLevel": "AUTO",
              "FieldEncoding": "PAFF",
              "SceneChangeDetect": "ENABLED",
              "QualityTuningLevel": "SINGLE_PASS_HQ",
              "UnregisteredSeiTimecode": "DISABLED",
              "GopSizeUnits": "SECONDS",
              "ParControl": "SPECIFIED",
              "NumberBFramesBetweenReferenceFrames": 3,
              "RepeatPps": "DISABLED",
              "DynamicSubGop": "ADAPTIVE"
            }
          },
          "AfdSignaling": "NONE",
          "DropFrameTimecode": "ENABLED",
          "RespondToAfd": "NONE",
          "ColorMetadata": "INSERT"
        },
        "AudioDescriptions": [
          {
            "AudioTypeControl": "FOLLOW_INPUT",
            "AudioSourceName": "Audio Selector 1",
            "CodecSettings": {
              "Codec": "AAC",
              "AacSettings": {
                "AudioDescriptionBroadcasterMix": "NORMAL",
```

```
                "Bitrate": 160000,
                "RateControlMode": "CBR",
                "CodecProfile": "LC",
                "CodingMode": "CODING_MODE_2_0",
                "RawFormat": "NONE",
                "SampleRate": 48000,
                "Specification": "MPEG4"
              }
            },
            "LanguageCodeControl": "FOLLOW_INPUT",
            "AudioType": 0
          }
        ]
      }
    ],
    "OutputGroupSettings": {
      "Type": "FILE_GROUP_SETTINGS",
      "FileGroupSettings": {
        "Destination": "s3://EXAMPLE-BUCKET/MP4/",
        "DestinationSettings": {
          "S3Settings": {
            "AccessControl": {
              "CannedAcl": "PUBLIC_READ"
            }
          }
        }
      }
    }
  },
  {
    "CustomName": "Thumbnails",
    "Name": "File Group",
    "Outputs": [
      {
        "ContainerSettings": {
          "Container": "RAW"
        },
        "VideoDescription": {
          "Width": 1280,
          "ScalingBehavior": "DEFAULT",
          "Height": 720,
          "TimecodeInsertion": "DISABLED",
          "AntiAlias": "ENABLED",
          "Sharpness": 50,
```

```
            "CodecSettings": {
              "Codec": "FRAME_CAPTURE",
              "FrameCaptureSettings": {
                "FramerateNumerator": 1,
                "FramerateDenominator": 5,
                "MaxCaptures": 500,
                "Quality": 80
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          }
        }
      ],
      "OutputGroupSettings": {
        "Type": "FILE_GROUP_SETTINGS",
        "FileGroupSettings": {
          "Destination": "s3://EXAMPLE-BUCKET/Thumbnails/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          }
        }
      }
    }
  ],
  "AdAvailOffset": 0,
  "Inputs": [
    {
      "AudioSelectors": {
        "Audio Selector 1": {
          "Offset": 0,
          "DefaultSelection": "DEFAULT",
          "ProgramSelection": 1
        }
      },
      "VideoSelector": {
        "ColorSpace": "FOLLOW"
      },
```

```
      "FilterEnable": "AUTO",
      "PsiControl": "USE_PSI",
      "FilterStrength": 0,
      "DeblockFilter": "DISABLED",
      "DenoiseFilter": "DISABLED",
      "TimecodeSource": "EMBEDDED",
      "FileInput": "s3://EXAMPLE-INPUT-BUCKET/input.mp4"
    }
  ]
}
```

3. 在 batch-transcode 資料夾中，使用 Lambda 函數建立檔案。您可使用下列 Python 範例並將檔案命名為 convert.py。

   S3 批次操作會將特定的任務資料傳送至 Lambda 函數，並需要傳回結果資料。如需 Lambda 函數的請求和回應範例、回應和結果程式碼的資訊，以及 S3 批次操作的範例 Lambda 函數，請參閱調用 AWS Lambda 函數。

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

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

Input parameter "event": The S3 Batch Operations event as a request
                         for the Lambda function.

Input parameter "context": Context about the event.

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

    invocation_schema_version = event['invocationSchemaVersion']
    invocation_id = event['invocationId']
```

```
    task_id = event['tasks'][0]['taskId']

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

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

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

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

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

        # The path of each output video is constructed based on the values of
        # the attributes in each object of OutputGroups in the job.json file.
        destination_s3 = 's3://{0}/{1}/{2}' \
            .format(os.environ['DestinationBucket'],
                    os.path.splitext(os.path.basename(source_s3_key))[0],
                    os.path.splitext(os.path.basename(job_name))[0])

        for output_group in job_settings['OutputGroups']:
            output_group_type = output_group['OutputGroupSettings']['Type']
            if output_group_type in output_group_type_dict.keys():
                output_group_type = output_group_type_dict[output_group_type]
                output_group['OutputGroupSettings'][output_group_type]
['Destination'] = \
                    "{0}{1}".format(destination_s3,
                                    urlparse(output_group['OutputGroupSettings']
[output_group_type]['Destination']).path)
```

```
            else:
                raise ValueError("Exception: Unknown Output Group Type {}."
                                 .format(output_group_type))

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

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

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

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

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

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

4. 若要將含有 `convert.py` 和 `job.json` 的部署套件建立為名稱為 `lambda.zip` 的 `.zip` 檔案，在您的本機終端機中，開啟您先前建立的 `batch-transcode` 資料夾，然後執行下列命令。

   若為 macOS 使用者，請執行下列命令：

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

   若為 Windows 使用者，請執行下列命令：

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

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

使用執行角色 (主控台) 建立 Lambda 函數

1. 請在以下位置開啟 AWS Lambda 主控台。 https://console.aws.amazon.com/lambda/
2. 在左側導覽窗格中，選擇 Functions (函數)。
3. 選擇建立函數。
4. 選擇 Author from scratch (從頭開始撰寫)。
5. 在基本資訊下，請執行下列動作：

   a. 針對 函數名稱 ，請輸入 **`tutorial-lambda-convert`**。

   b. 針對 Runtime (執行時間)，選擇 Python 3.8 或更新的 Python 版本。

6. 選擇 Change default execution role (變更預設執行角色)，並在 Execution role (執行角色) 下，選擇 Use an existing role (使用現有角色)。
7. 在 Existing role (現有角色) 下，選擇您在步驟 3 中為您的 Lambda 函數建立的 IAM 角色的名稱 (例如，**`tutorial-lambda-transcode-role`**)。
8. 對於其他設定，請保留預設值。

9. 選擇建立函數。

使用 .zip 檔案封存部署您的 Lambda 函數，並設定 Lambda 函數 (主控台)

1. 在頁面的 Code Source (程式碼來源) 區段中，為您先前建立的 Lambda 函數 (例如，**tutorial-lambda-convert**)，依次選擇 Upload from (上傳來源) 和 .zip 檔案。
2. 選擇 Upload (上傳) 以選取您的本機 `.zip` 檔案。
3. 選擇您之前建立的 `lambda.zip` 檔案，然後選擇 Open (開啟)。
4. 選擇儲存。
5. 在 Runtime settings (執行時間設定) 區段中，選擇 Edit (編輯)。
6. 若要告知 Lambda 執行時間要叫用 Lambda 函數程式碼中的處理常式方法，請在 Handler (處理常式) 欄位中輸入 **convert.handler**。

   當您設定 Python 函式時，處理常式的設定值就是檔案名稱加上處理常式模組名稱，並且以點 (.) 分隔。例如，`convert.handler` 會呼叫 `convert.py` 檔案中定義的 `handler` 方法。
7. 選擇儲存。
8. 在 Lambda 函數頁面上，選擇 Configuration (組態) 標籤。在 Configuration (組態) 索引標籤的左側導覽窗格中，選擇 Environment variables (環境變數)，然後選擇 Edit (編輯)。
9. 選擇 Add environment variable (新增環境變數)。然後，為以下每個環境變數輸入指定的 Key (索引鍵) 和 Value (數值)：

   - Key (索引鍵)：**DestinationBucket** Value (數值)：**tutorial-bucket-2**

     該數值是您在步驟 1 中建立的輸出媒體檔案的 S3 儲存貯體。
   - Key (索引鍵)：**MediaConvertRole** Value (數值)：**arn:aws:iam::*111122223333*:role/tutorial-mediaconvert-role**

     這個值是您在步驟 2 中建立的 IAM 角色的 ARN。 MediaConvert 請務必將此 ARN 取代為 IAM 角色的實際 ARN。
   - Key (索引鍵)：**Application** Value (數值)：**Batch-Transcoding**

     這是應用程式名稱的值。
10. 選擇儲存。
11. (選用) 在 Configuration (組態) 標籤上，在左側導覽窗格的 General configuration (一般組態) 區段中，選擇 Edit (編輯)。在 Timeout (逾時) 欄位中，輸入 **2** 分 **0** 秒。然後選擇 Save (儲存)。

Timeout (逾時) 是 Lambda 在停用函數前允許函數執行叫用的時間。預設為 3 秒。定價是根據設定的記憶體數量和程式碼執行的時間量而定。如需詳細資訊，請參閱 AWS Lambda 定價。

步驟 5：為您的 S3 來源儲存貯體設定 Amazon S3 清查

設定 Lambda 函數轉碼後，您會建立 S3 批次操作任務以轉碼一組影片。首先，您需要一份您希望 S3 批次操作對其執行指定轉碼動作的輸入影片物件清單。若要取得輸入影片物件的清單，您可以為 S3 來源儲存貯體 (例如，**tutorial-bucket-1**) 產生 S3 清查報告。

子步驟

- 為輸入影片的 S3 清查報告建立並設定儲存貯體
- 為 S3 影片來源儲存貯體設定 Amazon S3 清查
- 檢查您的 S3 影片來源儲存貯體的清查報告

為輸入影片的 S3 清查報告建立並設定儲存貯體

若要存放列出 S3 來源儲存貯體物件的 S3 清查報告，請建立 S3 清查目的地儲存貯體，然後為儲存貯體設定儲存貯體政策，以便將清查檔案寫入 S3 來源儲存貯體。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇建立儲存貯體。
4. 針對 Bucket name (儲存貯體名稱)，輸入儲存貯體的名稱 (例如 **tutorial-bucket-3**)。
5. 在中 AWS 區域，選擇您 AWS 區域 要儲存貯體的位置。

   庫存目的地儲存貯體必須與 AWS 區域 您設定 S3 庫存的來源儲存貯體位於相同的位置。清查目的地儲存貯體可位於不同的 AWS 帳戶之中。
6. 在 Block Public Access settings for this bucket (此儲存貯體的封鎖公開存取設定) 中，保留預設設定 (已啟用 Block all public access (封鎖所有公開存取))。
7. 對於其他設定，請保留預設值。
8. 選擇建立儲存貯體。
9. 在 Buckets (儲存貯體) 清單中，選擇您剛剛建立的儲存貯體名稱 (例如，**tutorial-bucket-3**)。

10. 若要授予 Amazon S3 將清查報告的資料寫入 S3 清查目的地儲存貯體的許可，請選擇 Permissions (許可) 標籤。
11. 向下捲動到 Bucket policy (儲存貯體政策) 區段，然後選擇 Edit (編輯)。Bucket policy (儲存貯體政策) 頁面隨即開啟。
12. 若要授予 S3 清查的許可，請在 Policy (政策) 欄位中，貼上以下儲存貯體政策。

    將三個範例值取代為下列值：

    - 您為存放清查報告而建立的儲存貯體名稱 (例如，*tutorial-bucket-3*)。
    - 存放輸入影片的來源儲存貯體名稱 (例如，*tutorial-bucket-1*)。
    - 您用來建立 S3 視訊來源儲存貯體的 AWS 帳戶 ID (例如，*111122223333*.

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

13. 選擇儲存變更。

為 S3 影片來源儲存貯體設定 Amazon S3 清查

若要產生影片物件和中繼資料的一般檔案清單，您需要為 S3 影片來源儲存貯體設定 S3 清查。這些排程清查報告可以包含儲存貯體中的所有物件，或是依共用字首分組的物件。在本教學課程中，S3 清查報告包含 S3 來源儲存貯體中的所有影片物件。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 若要在 S3 來源儲存貯體中設定輸入影片的 S3 清查報告，請在 Buckets (儲存貯體) 清單中，選擇 S3 來源儲存貯體的名稱 (例如，**tutorial-bucket-1**)。
4. 選擇 Management (管理) 標籤，
5. 向下捲動至 Inventory configurations (清查組態) 區段，然後選擇 Create inventory configuration (建立清查組態)。
6. 在 Inventory configuration name (清查組態名稱) 中，輸入一個名稱 (例如，**tutorial-inventory-config**)。
7. 在 Inventory scope (清查範圍) 下，針對 Object versions (物件版本) 選擇 Current version only (僅限目前版本)，並將其他 Inventory scope (清查範圍) 設定為本教學課程保留預設值。
8. 在 Report details (報告詳細資訊) 區段中，針對 Destination bucket (目的地儲存貯體)，選擇 This account (此帳戶)。
9. 針對 Destination (目的地)，選擇 Browse S3 (瀏覽 S3)，然後選擇您先前建立的目的地儲存貯體，以儲存清查報告 (例如，**tutorial-bucket-3**)。然後，選擇 Choose path (選擇路徑)。

   庫存目的地儲存貯體必須與 AWS 區域 您設定 S3 庫存的來源儲存貯體位於相同的位置。清查目的地儲存貯體可位於不同的 AWS 帳戶之中。

   在 Destination bucket (目的地儲存貯體) 欄位下，您會看到 Destination bucket permission (目的地儲存貯體許可)，這會新增至清查目的地儲存貯體政策，以允許 Amazon S3 將資料放入清查目的地儲存貯體中。如需詳細資訊，請參閱 建立目的地儲存貯體政策。
10. 針對 Frequency (頻率)，請選擇 Daily (每日)。
11. 在 Output format (輸出格式) 中，選擇 CSV。
12. 針對 Status (狀態)，請選擇 Enable (啟用)。
13. 在 Server-side encryption (伺服器端加密) 區段中，為此教學課程選擇 Disable (停用)。

    如需詳細資訊，請參閱 使用 S3 主控台設定清查 和 對 Amazon S3 授予許可使用您的客戶受管金鑰進行加密。

14. 在 Additional fields - optional (其他欄位 - 選用) 區段中，選取 Size (大小)、Last modified (上次修改) 及 Storage class (儲存方案)。
15. 選擇建立。

如需詳細資訊，請參閱 使用 S3 主控台設定清查。

檢查您的 S3 影片來源儲存貯體的清查報告

發佈清查報告時，資訊清單檔案會傳送到 S3 清查目的地儲存貯體。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇影片來源儲存貯體的名稱 (例如，**tutorial-bucket-1**)。
4. 選擇 Management (管理)。
5. 若要查看 S3 清查報告是否準備好讓您在步驟 7 中建立 S3 批次操作任務，則在 Inventory configurations (清查組態) 下，檢查是否已啟用 Create job from manifest (從資訊清單建立任務) 按鈕。

   > **Note**
   >
   > 最多可能需要 48 小時的時間來提供首次清查報告。如果停用 Create job from manifest (從資訊清單建立任務) 按鈕，則尚未交付第一份清查報告。等到第一份清查報告交付，並且已啟用 Create job from manifest (從資訊清單建立任務) 按鈕，才能在步驟 7 中建立 S3 批次操作任務。

6. 若要檢查 S3 清查報告 (`manifest.json`)，在 Destination (目的地) 欄位中，選擇您先前為存放清查報告而建立的清查目的地儲存貯體名稱 (例如，**tutorial-bucket-3**)。
7. 在 Objects (物件) 索引標籤上，選擇具有 S3 來源儲存貯體名稱的現有資料夾 (例如，**tutorial-bucket-1**)。然後在 Inventory configuration name (清查組態名稱) 中，選擇您先前建立清查組態時所輸入的名稱 (例如，**tutorial-inventory-config**)。

   您可以看到以報告產生日期為名稱的資料夾清單。
8. 若要檢查某個特定日期的每日 S3 清查報告，請選擇具有對應產生日期名稱的資料夾，然後選擇 `manifest.json`。

9. 若要在特定日期檢查清查報告的詳細資訊，請在 manifest.json 頁面中，選擇 Download (下載) 或 Open (開啟)。

步驟 6：為 S3 批次操作建立 IAM 角色

若要使用 S3 批次操作進行批次轉碼，您必須先建立 IAM 角色，授予 Amazon S3 執行 S3 批次操作的許可。

子步驟

- 為 S3 批次操作建立 IAM 政策
- 建立 S3 批次操作 IAM 角色並連接許可政策

為 S3 批次操作建立 IAM 政策

您必須建立 IAM 政策，授予 S3 批次操作讀取輸入資訊清單、叫用 Lambda 函數及寫入 S3 批次操作任務完成報告的許可。

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中選擇 Policies (政策)。
3. 選擇 Create policy (建立政策)。
4. 請選擇 JSON 標籤。
5. 在 JSON 文字欄位，貼上以下 JSON 政策。

   在 JSON 政策中，將四個範例值取代為下列值：

   - 存放您的輸入影片的來源儲存貯體名稱 (例如，*tutorial-bucket-1*)。
   - 您在步驟 5 中建立的用於存放 `manifest.json` 檔案的清查目的地儲存貯體的名稱 (例如，*tutorial-bucket-3*)。
   - 您在步驟 1 中建立的用於存放輸出媒體檔案的儲存貯體的名稱 (例如，*tutorial-bucket-2*)。在本教學課程中，我們把任務完成報告放在輸出媒體檔案的目的地儲存貯體。
   - 您在步驟 4 中建立的 Lambda 函數的角色 ARN。若要尋找並複製 Lambda 函數的角色 ARN，執行下列動作：
     - 在新的瀏覽器標籤中，在 https://console.aws.amazon.com/lambda/home#/functions 上開啟 Lambda 主控台中的 Functions (函數) 頁面。

- 在 Functions (函數) 清單中，選擇您在步驟 4 中建立的 Lambda 函數名稱 (例如，**tutorial-lambda-convert**)。
- 選擇 Copy ARN (複製 ARN)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Get",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::tutorial-bucket-1/*",
                "arn:aws:s3:::tutorial-bucket-3/*"
            ]
        },
        {
            "Sid": "S3PutJobCompletionReport",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::tutorial-bucket-2/*"
        },
        {
            "Sid": "S3BatchOperationsInvokeLambda",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-west-2:111122223333:function:tutorial-lambda-
convert"
            ]
        }
    ]
}
```

6. 選擇下一步：標籤。
7. 選擇下一步：檢閱。

8. 在 Name (名稱) 欄位中，輸入 **tutorial-s3batch-policy**。
9. 選擇建立政策。

建立 S3 批次操作 IAM 角色並連接許可政策

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 選擇 AWS 服務 角色類型，然後選擇 S3 服務。
4. 在 Select your use case (選取使用案例) 下，選擇 S3 Batch Operations (S3 批次操作)。
5. 選擇下一步：許可。
6. 在 Attach permissions policies (連接許可政策) 下，在搜尋方塊中輸入您先前建立的 IAM 政策名稱 (例如，**tutorial-s3batch-policy**) 來篩選政策清單。選取政策名稱 (例如，**tutorial-s3batch-policy**) 旁的核取方塊。
7. 選擇下一步：標籤。
8. 選擇下一步：檢閱。
9. 在角色名稱中，輸入 **tutorial-s3batch-role**。
10. 選擇建立角色。

    建立 S3 批次操作的 IAM 角色之後，下列信任政策會自動連接到該角色。此信任政策會允許 S3 批次操作服務主體擔任 IAM 角色。

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

步驟 7：建立並執行 S3 批次操作任務

若要建立 S3 批次操作任務以處理 S3 來源儲存貯體中的輸入影片，您必須指定此特定任務的參數。

> **Note**
>
> 在開始建立 S3 批次操作任務之前，請確定 Create job from manifest (從資訊清單建立任務) 按鈕已啟用。如需詳細資訊，請參閱 檢查您的 S3 影片來源儲存貯體的清查報告。如果 Create job from manifest (從資訊清單建立任務) 按鈕已停用，則尚未交付第一份清查報告且您必須等到該按鈕啟用為止。在步驟 5 中為 S3 來源儲存貯體設定 Amazon S3 清查後，交付第一份清查報告最多可能需要 48 小時的時間。

子步驟

- 建立一個 S3 批次操作任務
- 執行 S3 批次操作任務來叫用 Lambda 函數
- (選用) 檢查完成報告
- (選用) 監控 Lambda 主控台中的每個 Lambda 叫用
- (選擇性) 監控主控台中的每個 MediaConvert 視訊轉碼工作 MediaConvert

建立一個 S3 批次操作任務

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Batch Operations (批次操作)。
3. 選擇建立作業。
4. 針對 AWS 區域，選擇要在其中建立任務的區域。

   在本教學課程中，若要使用 S3 批次操作任務來叫用 Lambda 函數，您必須在與資訊清單中參照的物件所在的 S3 影片來源儲存貯體相同的區域中建立任務。
5. 在 Manifest (資訊清單) 區段中，執行下列動作：

   a. 針對 Manifest format (資訊清單格式)，選擇 S3 Inventory report (manifest.json) (S3 庫存報告 (manifest.json))。

   b. 針對 Manifest object (資訊清單物件)，選擇 Browse S3 (瀏覽 S3)，以查找您在步驟 5 中為存放清查報告而建立的儲存貯體 (例如，**`tutorial-bucket-3`**)。在 Manifest object (資訊清單

物件) 頁面上，瀏覽物件名稱，直到找到特定日期的 `manifest.json` 檔案。此檔案會列出您想要批次轉碼的所有影片的相關資訊。當您找到 `manifest.json` 檔案時，請選擇檔案旁邊的選項按鈕。然後，選擇 Choose path (選擇路徑)。

c. (選用) 對於 Manifest object version ID - optional (資訊清單物件版本 ID - 選用)，如果您想要使用最新版本以外的版本，請輸入資訊清單物件的版本 ID。

6. 選擇下一步。
7. 若要使用 Lambda 函數轉碼選定的 `manifest.json` 檔案中列出的所有物件，請在 Operation type (操作類型) 下，選擇 Invoke AWS Lambda function (叫用 AWS Lambda 函數)。
8. 在 Invoke Lambda function (叫用 Lambda 函數) 區段中，執行下列動作：

   a. 選擇 Choose from functions in your account (從帳戶中的函數選擇)。

   b. 針對 Lambda function (Lambda 函數)，選擇您在步驟 4 中建立的 Lambda 函數 (例如，**`tutorial-lambda-convert`**)。

   c. 針對 Lambda function version (Lambda 函數版本)，請保留為預設值 $LATEST。

9. 選擇下一步。Configure additional options (設定其他選項) 頁面隨即開啟。
10. 在 Additional options (其他選項) 區段中，保留預設設定。

    如需關於這些選項的詳細資訊，請參閱 批次操作任務請求元素。

11. 在 Completion report (完成報告) 區段中, 針對 Path to completion report destination (完成報告目的地的路徑)，選擇 Browse S3 (瀏覽 S3)。找出在步驟 1 中您為輸出媒體檔案建立的儲存貯體 (例如，**`tutorial-bucket-2`**)。選擇儲存貯體名稱旁的選項按鈕。然後，選擇 Choose path (選擇路徑)。

    對於剩餘的 Completion report (完成報告) 設定，請保留預設值。如需完成報告設定的詳細資訊，請參閱 批次操作任務請求元素。完成報告會維護任務的詳細資訊和執行的操作的記錄。

12. 在 Permissions (許可) 區段中，選擇 Choose from existing IAM roles (從現有 IAM 角色中選擇)。對於 IAM role (IAM 角色)，為您在步驟 6 中建立的 S3 批次操作任務選擇 IAM 角色 (例如，**`tutorial-s3batch-role`**)。
13. 選擇下一步。
14. 請詳閱 Review (檢閱) 頁面上的設定。然後，選擇 Create Job (建立任務)。

    在 S3 完成讀取 S3 批次操作任務的資訊清單後，會將任務的 Status (狀態) 設定為 Awaiting your confirmation to run (等待確認執行)。若要查看任務狀態的更新，請重新整理頁面。您無法執行任務，直到其狀態為 Awaiting your confirmation to run (等待確認執行)。

執行 S3 批次操作任務來叫用 Lambda 函數

執行批次操作任務以叫用 Lambda 函數進行影片轉碼。如果您的任務失敗，您可以檢查完成報告，以找出原因。

執行 S3 批次操作任務

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Batch Operations (批次操作)。
3. 從 Jobs (任務) 清單中，選擇第一列的任務的 Job ID (任務 ID)，也就是您先前建立的 S3 批次操作任務。
4. 選擇執行工作。
5. 再次檢閱您的任務參數，並確認 Total objects listed in manifest (資訊清單中列出的物件總數) 的數值與資訊清單中的物件數目相同。接著選擇 Run job (執行任務)。

   隨即開啟 S3 批次操作任務頁面。
6. 開始執行任務後，在任務頁面上的 Status (狀態) 下，檢查 S3 批次操作任務的進度，例如 Status (狀態)、% Complete (完成百分比)、Total succeeded (rate) (總成功 (率))、Total failed (rate) (總失敗 (率))、Date terminated (終止日期) 以及 Reason for termination (終止的原因)。

   S3 批次操作任務完成時，請檢視任務頁面上的資料，以確認任務是否如預期完成。

   如果超過 50% 的 S3 批次操作任務的物件操作在嘗試超過 1,000 次操作後失敗，則任務會自動失敗。若要檢查完成報告以識別失敗的原因，請使用下列選用程序。

(選用) 檢查完成報告

您可以使用完成報告來判斷哪些物件失敗以及失敗的原因。

若要檢查完成報告，以取得失敗物件的詳細資訊

1. 在 S3 批次操作任務的頁面上，向下捲動至 Completion report (完成報告) 區段，然後選擇 Completion report destination (完成報告目的地) 下的連結。

   S3 輸出目的地儲存貯體頁面隨即開啟。
2. 在 Objects (物件) 索引標籤上，選擇名稱以您先前建立之 S3 批次操作任務的任務 ID 為結尾的資料夾。

3. 選擇 results/。
4. 選取 `.csv` 檔案旁的核取方塊。
5. 若要檢視任務報告，請選擇 Open (開啟) 或 Download (下載)。

(選用) 監控 Lambda 主控台中的每個 Lambda 叫用

開始執行 S3 批次操作任務之後，任務會叫用每個輸入影片物件的 Lambda 函數。S3 會將每個 Lambda 叫用的記錄寫入 CloudWatch 日誌。您可以使用 Lambda 主控台的監控儀表板來監控您的 Lambda 函數。

1. 請在以下位置開啟 AWS Lambda 主控台。 https://console.aws.amazon.com/lambda/
2. 在左側導覽窗格中，選擇 Functions (函數)。
3. 在 Functions (函數) 清單中，選擇您在步驟 4 中建立的 Lambda 函數名稱 (例如，**`tutorial-lambda-convert`**)。
4. 選擇 監控 索引標籤。
5. 在 Metrics (指標) 下，請參閱 Lambda 函數的執行時間指標。
6. 在「記錄檔」下，透過日誌深入解析檢視每個 Lambda 叫用的 CloudWatch 日誌資料。

> **Note**
>
> 當您搭配 Lambda 函數使用 S3 批次操作時，Lambda 函數會叫用每個物件。如果您的 S3 批次操作任務較大，則可以同時叫用多個 Lambda 函數，會造成 Lambda 並行峰值。每個區域都 AWS 帳戶 有一個 Lambda 並行配額。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 AWS Lambda 函數擴展。搭配 S3 批次操作使用 Lambda 函數的最佳實務是設定 Lambda 函數本身的並行限制。設定並行限制可以避免您的任務耗用大部分 Lambda 並行，並可能調節您帳戶中的其他功能。如需詳細資訊，請參閱中的《AWS Lambda 開發人員指南》中的管理 Lambda 預留並行。

(選擇性) 監控主控台中的每個 MediaConvert 視訊轉碼工作 MediaConvert

工作完成對媒體文件進行轉碼的工作。 MediaConvert 當 S3 Batch 操作任務為每個視訊叫用 Lambda 函數時，每個 Lambda 函數呼叫都會為每個輸入視訊建立 MediaConvert 轉碼任務。

1. 請登入 AWS Management Console 並開啟 MediaConvert 主控台，網址為 https://console.aws.amazon.com/mediaconvert/。

2. 如果出現 MediaConvert 介紹頁面，請選擇 [開始使用]。
3. 從 Jobs (任務) 清單中，檢視每一列，以監控每個輸入影片的轉碼任務。
4. 識別您要檢查的任務列，然後選擇 Job ID (任務 ID) 連結，以開啟任務詳細資訊頁面。
5. 在 Job summary (任務摘要) 頁面，在 Outputs (輸出) 下，根據瀏覽器支援的內容，選擇 HLS、MP4 或縮圖輸出的連結，以前往輸出媒體檔案的 S3 目的地儲存貯體。
6. 在 S3 輸出目的地儲存貯體的對應資料夾 (HLS、MP4 或縮圖) 中，選擇輸出媒體檔案物件的名稱。

   物件詳細資訊頁面隨即開啟。
7. 在物件詳細資訊頁面的 Object overview (物件概觀) 下，選擇 Object URL (物件 URL)，以觀看轉碼後的輸出媒體檔案。

步驟 8：從 S3 目的地儲存貯體檢查輸出媒體檔案

若要從 S3 目的地儲存貯體檢查輸出媒體檔案

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇您在步驟 1 中建立的輸出媒體檔案的 S3 目的地儲存貯體名稱 (例如，**`tutorial-bucket-2`**)。
4. 在 Objects (物件) 標籤上，每個輸入影片都有一個具有輸入影片名稱的資料夾。每個資料夾皆包含輸入影片的轉碼後的輸出媒體檔案。

   若要檢查輸入影片的輸出媒體檔案，請執行下列動作：

   a. 選擇以您要檢查的輸入影片名稱命名的資料夾。
   b. 選擇 Default/ 資料夾。
   c. 選擇轉碼後的格式的資料夾 (本教學課程中為 HLS、MP4 或縮圖)。
   d. 選擇輸出媒體檔案的名稱。
   e. 如要觀看轉碼後的檔案，在物件詳細資訊頁面上，選擇 Object URL (物件 URL) 下的連結。

      HLS 格式的輸出媒體檔案會分割成數個短區段。若要播放這些影片，請在相容的播放器中嵌入 `.m3u8` 檔案。

步驟 9：清除

如果您使用 S3 Batch 操作 Lambda 轉碼影片，並且 MediaConvert 僅作為學習練習，請刪除您分配的 AWS 資源，以免再產生費用。

子步驟

- 刪除 S3 來源儲存貯體的 S3 清查組態
- 刪除 Lambda 函數
- 刪除記 CloudWatch 錄群組
- 刪除 IAM 角色以及 IAM 角色的內嵌政策
- 刪除客戶受管的 IAM 政策
- 清空 S3 儲存貯體
- 刪除 S3 儲存貯體

刪除 S3 來源儲存貯體的 S3 清查組態

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇來源儲存貯體的名稱 (例如，**tutorial-bucket-1**)。
4. 選擇 Management (管理) 標籤，
5. 在 Inventory configurations (清查組態) 中，選擇您在步驟 5 中建立的清查組態 (例如，**tutorial-inventory-config**) 旁的選項按鈕。
6. 選擇 Delete (刪除)，然後選擇 Confirm (確認)。

刪除 Lambda 函數

1. 請在以下位置開啟 AWS Lambda 主控台。 https://console.aws.amazon.com/lambda/
2. 在左側導覽窗格中，選擇 Functions (函數)。
3. 選取您在步驟 4 中建立的函數旁的核取方塊 (例如，**tutorial-lambda-convert**)。
4. 選擇動作，然後選擇刪除。
5. 在 Delete function (刪除函數) 對話方塊中，選擇 Delete (刪除)。

刪除記 CloudWatch 錄群組

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在左側導覽窗格中，選擇 Logs (日誌)，然後選擇 Log groups (日誌群組)。
3. 選取日誌群組旁的核取方塊，該日誌群組的名稱以您在步驟 4 中建立的 Lambda 函數結尾 (例如，**`tutorial-lambda-convert`**)。
4. 選擇 Actions (動作)，然後選擇 Delete log group (刪除日誌群組)。
5. 在 刪除日誌群組 對話方塊中，選擇 刪除 。

刪除 IAM 角色以及 IAM 角色的內嵌政策

若要刪除您在步驟 2、步驟 3 及步驟 6 中建立的 IAM 角色，請執行下列動作：

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中，選擇 Roles (角色)，然後選取您要刪除之角色名稱旁的核取方塊。
3. 在頁面頂端，選擇 Delete (刪除)。
4. 在確認對話方塊中，根據提示在文字輸入欄位中輸入所需的回應，然後選擇 Delete (刪除)。

刪除客戶受管的 IAM 政策

若要刪除您在步驟 6 中建立的客戶受管的 IAM 政策，請執行下列動作：

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中選擇 Policies (政策)。
3. 選擇您在步驟 6 中建立之政策 (例如，**`tutorial-s3batch-policy`**) 旁的選項按鈕。您可以使用搜尋方塊來篩選政策清單。
4. 選擇 動作，然後選擇 刪除。
5. 在文字欄位中，輸入本政策的名稱，以確認您要刪除此政策，然後選擇 Delete (刪除)。

清空 S3 儲存貯體

若要清空您在先決條件、步驟 1 及步驟 5 中建立的 S3 儲存貯體，請執行下列動作：

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Bucket (儲存貯體) 清單中，選擇您要清空之儲存貯體名稱旁的選項按鈕，然後選擇 Empty (清空)。
4. 在 Empty bucket (清空儲存貯體) 頁面上，在文字欄位中輸入 **`permanently delete`** 以確認您要清空儲存貯體，然後選擇 Empty (清空)。

刪除 S3 儲存貯體

若要刪除您在先決條件、步驟 1 及步驟 5 中建立的 S3 儲存貯體，請執行下列動作：

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇您要刪除之儲存貯體名稱旁的選項按鈕。
4. 選擇刪除。
5. 在 Delete bucket (刪除儲存貯體) 頁面上，在文字欄位中輸入儲存貯體名稱以確認您要刪除該儲存貯體，然後選擇 Delete bucket (刪除儲存貯體)。

後續步驟

完成本教學課程後，您可以進一步探索其他相關使用案例：

- 您可以使用 Amazon CloudFront 將已轉碼的媒體檔案串流給全球的觀眾。如需詳細資訊，請參閱 教學課程：使用 Amazon S3、Amazon 和 Amazon Route 53 託管隨選串流視訊 CloudFront。
- 當您將影片上傳到 S3 來源儲存貯體時，您可以對影片進行轉碼。為此，您可以設定 Amazon S3 事件觸發器，該觸發器會自動叫用 Lambda 函數，在 S3 中將新物件轉碼。 MediaConvert如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的教學課程：使用 Amazon S3 觸發器叫用 Lambda 函數。

教學課程：在 Amazon S3 上設定靜態網站

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

您可以配置 Amazon S3 儲存貯體配置成網站一般地運作。此範例會演練如何在 Amazon S3 上託管網站的步驟。

> **Important**
>
> 下列教學課程需要停用「封鎖公開存取」。我們建議將「封鎖公開存取」保持啟用狀態。如果您想要保持啟用所有四個「封鎖公用存取」設定並託管靜態網站，可以使用 Amazon CloudFront 來源存取控制 (OAC)。Amazon CloudFront 提供設定安全靜態網站所需的功能。Amazon S3 靜態網站只支援 HTTP 端點。亞馬遜 CloudFront 使用 Amazon S3 的耐用儲存，同時提供額外的安全標頭，例如 HTTPS。HTTPS 透過加密一般 HTTP 請求並防止常見的網路攻擊來增加安全性。如需詳細資訊，請參閱 Amazon 開 CloudFront 發人員指南中的安全靜態網站入門。

主題

步驟 1：建立儲存貯體

以下指示提供如何為網站託管建立儲存貯體的概觀。如需建立值區的詳細 step-by-step 說明，請參閱建立儲存貯體。

建立儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 選擇 Create bucket (建立儲存貯體)。
3. 輸入 Bucket name (儲存貯體名稱) (例如，**`example.com`**)。
4. 選擇您要在其中建立儲存貯體的區域。

   建議您選擇接近您地理位置的區域以充分降低延遲及成本，或因應法規要求。您選擇的區域會決定您的 Amazon S3 網站端點。如需詳細資訊，請參閱「網站端點」。
5. 若要接受預設設定並建立儲存貯體，請選擇 Create (建立)。

步驟 2：啟用靜態網站託管

建立儲存貯體後，您就可以為儲存貯體啟用靜態網站託管。您可以建立新的儲存貯體，或使用現有的儲存貯體。

啟用靜態網站託管

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體名稱) 清單中，選擇希望為其啟用靜態網站託管的儲存貯體名稱。
3. 選擇 Properties (屬性)。
4. 在 Static website hosting (靜態網站託管) 下，選擇 Edit (編輯)。
5. 選擇 Use this bucket to host a website (使用此儲存貯體來託管網站)。
6. 在 Static website hosting (靜態網站託管) 下，選擇Enable (啟用)。
7. 在索引文件中，輸入索引文件的名稱，通常是 `index.html`。

   索引文件名稱區分大小寫，而且必須完全符合您計畫上傳至 S3 儲存貯體的 HTML 索引文件檔案名稱。當您為網站託管設定儲存貯體時，必須指定索引文件。在對根網域或任何子資料夾提出請求時，Amazon S3 會傳回此索引文件。如需詳細資訊，請參閱「設定索引文件」。

8. 若要為 4XX 類別錯誤提供自己的自訂錯誤文件，請在 Error document (錯誤文件) 中輸入自訂錯誤文件檔案名稱。

   錯誤文件名稱區分大小寫，而且必須完全符合您計畫上傳至 S3 儲存貯體的 HTML 錯誤文件檔案名稱。如果您未指定自訂錯誤文件且發生錯誤，則 Amazon S3 會傳回預設的 HTML 錯誤文件。如需詳細資訊，請參閱 設定自訂錯誤文件。

9. (選用) 如果您要指定進階重新導向規則，請在 Redirection rules (重新導向規則) 中輸入 JSON 來描述規則。

   例如，您可依據要求中特定的物件金鑰名稱或字首，依條件路由要求。如需詳細資訊，請參閱「配置重新引導規則以使用進階條件重新引導」。

10. 選擇 Save changes (儲存變更)。

    Amazon S3 會為您的儲存貯體啟用靜態網站託管。在頁面底部的靜態網站託管下，您會看到儲存貯體的網站端點。

11. 在 Static website hosting 下，請記下 Endpoint (端點)。

    端點是儲存貯體的 Amazon S3 網站端點。將儲存貯體設為靜態網站之後，您可以使用此端點來測試您的網站。

步驟 3：編輯封鎖公開存取設定

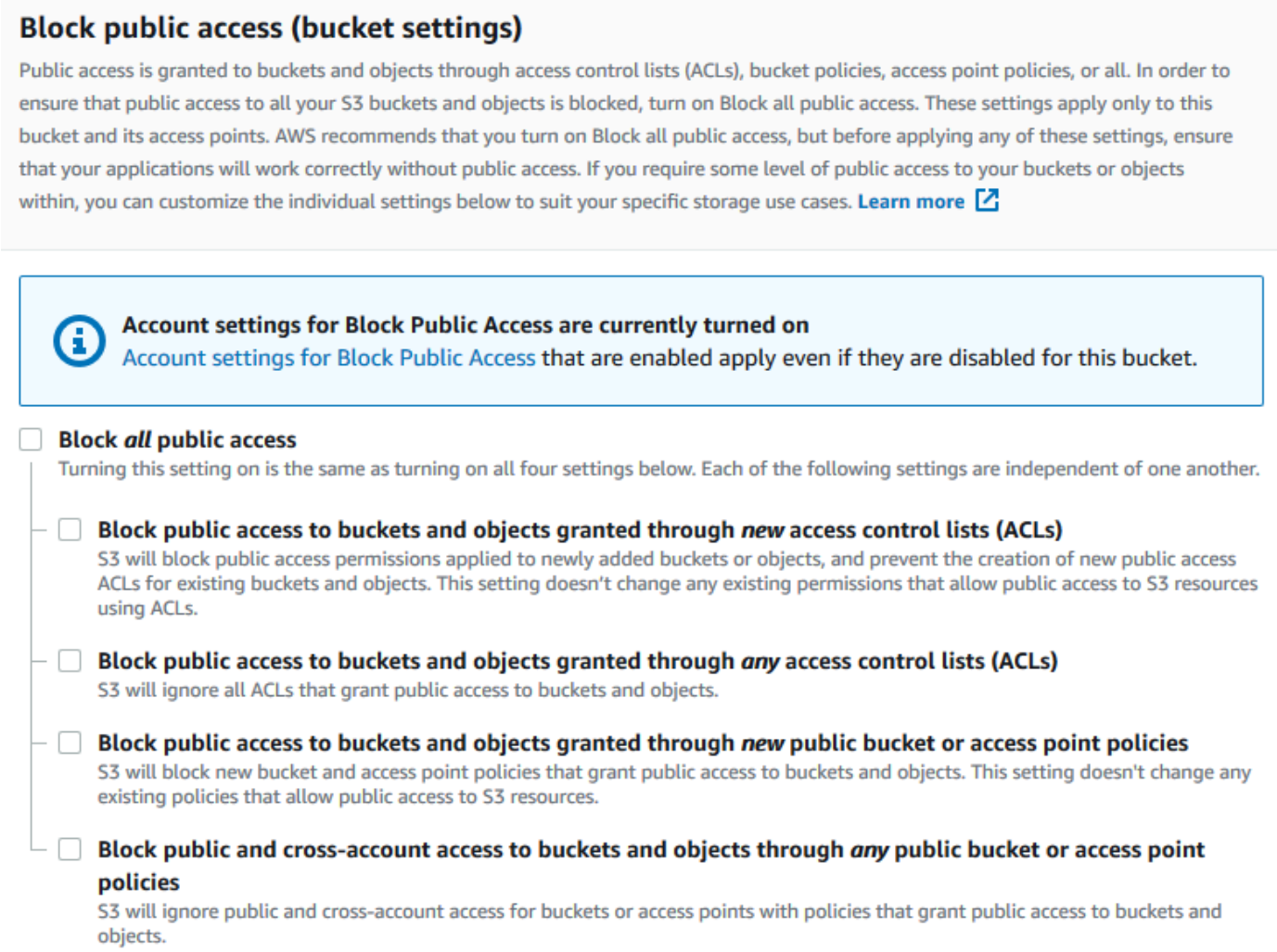
根據預設，Amazon S3 會封鎖對帳戶和儲存貯體的公開存取。如想要使用儲存貯體託管靜態網站，您可使用這些步驟編輯封鎖公有存取設定：

> **Warning**
>
> 在完成此步驟之前，請檢閱 封鎖對 Amazon S3 儲存體的公開存取權 以確保您了解並接受允許公開存取所涉及的風險。當您關閉封鎖公開存取設定以公開儲存貯體時，網際網路上的任何人都可以存取您的儲存貯體。我們建議您封鎖對儲存貯體的所有公用存取權。

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 選擇已設定為靜態網站的儲存貯體名稱。
3. 選擇 Permissions (許可)。
4. 在 Block public access (bucket settings) (封鎖公開存取 (儲存貯體設定)) (封鎖公開存取 (儲存貯體設定)) 下，選擇 Edit (編輯)。

5. 清除 Block all public access (封鎖所有公開存取)，然後選擇 Save changes (儲存變更)。

> **Warning**
>
> 在完成此步驟之前，請檢閱 封鎖對 Amazon S3 儲存體的公開存取權 以確保您了解並接受允許公開存取所涉及的風險。當您關閉封鎖公開存取設定以公開儲存貯體時，網際網路上的任何人都可以存取您的儲存貯體。我們建議您封鎖對儲存貯體的所有公用存取權。



Amazon S3 會關閉儲存貯體的封鎖公開存取設定。若要建立公開的靜態網站，在新增儲存貯體原則之前，可能還需要針對您的帳戶編輯封鎖公開存取設定。如果帳戶的封鎖公開存取設定目前已開啟，您在 封鎖公開存取 (儲存貯體設定) 下會看到附註。

步驟 4：新增儲存貯體政策，將儲存貯體內容設為可供大眾讀取

編輯 S3 封鎖公用存取設定之後，您可以新增儲存貯體政策，以授予儲存貯體的公用讀取權限。當您授予公有讀取權限時，網際網路上的任何人都可以存取您的儲存貯體。

> **Important**
>
> 以下政策僅為範例，允許完整存取您儲存貯體的內容。繼續執行此步驟之前，請檢閱如何保護 Amazon S3 儲存貯體中的檔案？，以確保您瞭解 S3 儲存貯體中檔案保護的最佳實務，以及授予公開存取權所涉及的風險。

1. 在 Buckets(儲存貯體) 下方，選擇儲存貯體的名稱。
2. 選擇 Permissions (許可)。
3. 在 Bucket Policy (儲存貯體政策) 下方，選擇 Edit (編輯)。
4. 若要授予您網站的公開讀取存取權，請複製以下儲存貯體政策，並將它貼上至 Bucket policy editor (儲存貯體政策編輯器)。

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

5. 將 `Resource` 更新為您的儲存貯體名稱。

   在上述範例儲存貯體政策中，*`Bucket-Name`* 是儲存貯體名稱的預留位置。若要使用此儲存貯體策略與您自己的儲存貯體搭配，您必須更新此名稱以符合您的儲存貯體名稱。

6. 選擇 Save changes (儲存變更)。

顯示的訊息指出已成功新增儲存貯體原則。

如果您看到指出 `Policy has invalid resource` 的錯誤，請確認儲存貯體政策中的儲存貯體名稱與您的儲存貯體名稱相符。如需有關新增儲存貯體原則的資訊，請參閱如何新增 S3 儲存貯體原則？

如果您收到錯誤訊息且無法儲存貯體原則，請檢查您的帳戶和儲存貯體的封鎖公開存取設定，以確認您允許公開存取儲存貯體。

步驟 5：設定索引文件

當您為儲存貯體啟用靜態網站託管時，請輸入索引文件的名稱 (例如，**`index.html`**)。為儲存貯體啟用靜態網站託管後，您可以將含有索引文件名稱的 HTML 檔案上傳到儲存貯體。

設定索引文件

1. 建立 `index.html` 檔案。

   如果您還沒有 `index.html` 檔案，您可以使用下列HTML 建立一個檔案。

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

2. 在本機儲存索引檔案。

   索引文件檔案名稱必須完全符合您在 Static website hosting (靜態網站託管) 對話方塊中輸入的索引文件名稱。索引文件名稱有區分大小寫。例如，如果您在 Static website hosting (靜態網站託管) 對話方塊的 Index document (索引文件) 名稱中輸入 `index.html`，您的索引文件檔案名稱也必須是 `index.html` 而非 `Index.html`。

3. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

4. 在 Buckets (儲存貯體) 清單中，選擇您要用於託管靜態網站的儲存貯體名稱。

5. 為您的儲存貯體啟用靜態網站，然後輸入索引文件的確切名稱 (例如，`index.html`)。如需詳細資訊，請參閱「啟用網站託管」。

   啟用靜態網路託管之後，請繼續執行步驟 6。

6. 若要將索引文件上傳至您的儲存貯體，請執行下列其中一項：

   - 將索引檔拖放到主控台儲存貯體清單中。
   - 選擇 Upload (上傳)，然後依照提示選擇並上傳索引檔案。

   如需 step-by-step 指示，請參閱上傳物件。

7. (選用) 將其他網站內容上傳到您的儲存貯體。

步驟 6：設定錯誤文件

當您為儲存貯體啟用靜態網站託管時，請輸入錯誤文件的名稱 (例如 **`404.html`**)。為儲存貯體啟用靜態網站託管後，您可以將含有錯誤引文件名稱的 HTML 檔案上傳到儲存貯體。

設定錯誤文件

1. 建立錯誤文件，例如 `404.html`。
2. 將錯誤文件檔案儲存在本機。

   錯誤文件名稱區分大小寫，且須完全符合您在啟用靜態網站託管時所輸入的名稱。例如，如果您在 Static website hosting (靜態網站託管) 對話方塊的 Error document (錯誤文件) 名稱中輸入 `404.html`，您的錯誤文件檔案名稱也必須是 `404.html`。

3. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
4. 在 Buckets (儲存貯體) 清單中，選擇您要用於託管靜態網站的儲存貯體名稱。
5. 為您的儲存貯體啟用靜態網站託管，並輸入錯誤文件的確切名稱 (例如 `404.html`)。如需詳細資訊，請參閱 啟用網站託管 及 設定自訂錯誤文件。

   啟用靜態網路託管之後，請繼續執行步驟 6。

6. 若要將錯誤文件上傳至您的儲存貯體，請執行下列其中一項：

   - 將錯誤文件檔案拖放至主控台儲存貯體清單中。
   - 選擇 Upload (上傳)，然後依照提示選擇並上傳索引檔案。

如需 step-by-step 指示，請參閱上傳物件。

步驟 7：測試您的網站端點

在設定儲存貯體的靜態網站託管後，您就可以測試網站端點。

> **Note**
>
> Amazon S3 不支援使用 HTTPS 存取網站。如果您想使用 HTTPS，則可以使用 Amazon CloudFront 為 Amazon S3 上託管的靜態網站提供服務。
> 如需詳細資訊，請參閱如 CloudFront 何使用在 Amazon S3 上託管的靜態網站提供服務？ 以及要求 HTTPS 才能在檢視者和 CloudFront.

1. 在 Buckets (儲存貯體) 下方，選擇儲存貯體的名稱。
2. 選擇 Properties (屬性)。
3. 在頁面底部的 Static website hosting (靜態網站託管) 下，選擇您的 Bucket website endpoint (儲存貯體網站端點)。

   您的索引文件會在不同的瀏覽器視窗中開啟。

您已在 Amazon S3 上完成網站託管。此網站已可在 Amazon S3 網站端點上使用。但您可能有像 `example.com` 的網域，想要將其用來提供所建立之網站的內容。您也可能想要使用 Amazon S3 根網域支援來自 `http://www.example.com` 及 `http://example.com` 的請求。這需要額外的步驟。如需範例，請參閱 教學課程：使用向 Route 53 註冊的自訂網域設定靜態網站。

步驟 8：清除

若建立的靜態網站僅供學習練習之用，請先刪除已配置的 AWS 資源，如此即不會再產生費用。刪除 AWS 資源後，您的網站將無法再使用。如需詳細資訊，請參閱 刪除儲存貯體。

教學課程：使用向 Route 53 註冊的自訂網域設定靜態網站

假設您想要在 Amazon S3 上託管靜態網站。您已經在 Amazon Route 53 註冊了一個網域 (例如，`example.com`)，而且您希望從 Amazon S3 內容提供 `http://www.example.com` 和

`http://example.com` 提供請求。您可以使用此演練來了解如何託管靜態網站，以及在 Amazon S3 上為使用 Route 53 註冊之自訂網域名稱的網站建立重新導向。您可以從要在 Amazon S3 上託管的現有網站著手，或使用此演練從頭開始。

完成此逐步解說後，您可以選擇使用 Amazon CloudFront 來改善網站的效能。如需詳細資訊，請參閱加快您的網站與 Amazon CloudFront。

> **Note**
>
> Amazon S3 網站端點不支援 HTTPS 或存取點。如果您想使用 HTTPS，則可以使用 Amazon CloudFront 為 Amazon S3 上託管的靜態網站提供服務。
> 如需如何使用 CloudFront 和 Amazon S3 安全託管內容的教學課程，請參閱教學課程：使用 Amazon S3、Amazon 和 Amazon Route 53 託管隨選串流視訊 CloudFront。如需詳細資訊，請參閱如CloudFront 何使用在 Amazon S3 上託管的靜態網站提供服務？ 以及要求 HTTPS 才能在檢視者和 CloudFront.

使用模板自動化靜態網站設置 AWS CloudFormation

您可以使用 AWS CloudFormation 範本將靜態網站設定自動化。該 AWS CloudFormation 模板設置了託管安全靜態網站所需的組件，以便您可以更專注於網站的內容，而不必擔心配置組件。

該 AWS CloudFormation 模板包括以下組件：

- Amazon S3 – 建立一個 Amazon S3 儲存貯體來託管您的靜態網站。
- CloudFront — 創建一個 CloudFront 分佈，以加快您的靜態網站。
- Lambda@Edge - 使用 Lambda@Edge 將安全標頭新增至每個伺服器回應。安全標頭是 Web 服務器回應中的一組標頭，告訴 Web 瀏覽器採取額外的安全預防措施。如需詳細資訊，請參閱部落格文章使用 Lambda @Edge 和 Amazon 新增 HTTP 安全標頭 CloudFront。

此 AWS CloudFormation 模板可供您下載和使用。如需相關資訊和指示，請參閱 Amazon 開 CloudFront 發人員指南中的安全靜態網站入門。

主題

開始之前

當您依此範例中的步驟進行操作，您將會使用下列服務：

Amazon Route 53 – 您使用 Route 53 註冊網域，以及為您的網域定義網路流量要路由到的位置。範例說明如何建立 Route 53 別名記錄，將網域 (`example.com`) 和子網域 (`www.example.com`) 的流量路由到包含 HTML 檔案的 Amazon S3 儲存貯體。

Amazon S3 – 您使用 Amazon S3 來建立儲存貯體、上傳網站頁面範例、配置許可以讓所有人都能看到內容，然後配置儲存貯體處理網站託管。

步驟 1：使用 Route 53 註冊自訂網域

如果您尚沒有註冊網域名稱，如：`example.com`，請利用 Route 53 註冊。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》中的註冊新網域。註冊您的網域名稱後，您可以建立和配置 Amazon S3 儲存貯體用於網站託管。

步驟 2：建立兩個儲存貯體

為了同時從根網域與子網域支援請求，您會建立兩個儲存貯體。

- 網域儲存貯體 - `example.com`
- 子網域儲存貯體 - `www.example.com`

這些儲存貯體名稱必須完全符合您的網域名稱。在此範例中，網域名稱是 example.com。您將內容託管在根網域儲存貯體外部 (example.com)。您為子網域儲存貯體建立重新導向請求 (www.example.com)。如果有人在其瀏覽器輸入 www.example.com，系統會將他們重新導向至 example.com，然後他們會看到在具有該名稱的 Amazon S3 儲存貯體中託管的內容。

為網站託管建立儲存貯體

以下指示提供如何為網站託管建立儲存貯體的概觀。如需建立值區的詳細 step-by-step 說明，請參閱建立儲存貯體。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 建立您的根網域儲存貯體：
   a. 選擇 Create bucket (建立儲存貯體)。
   b. 輸入 Bucket name (儲存貯體名稱) (例如，**example.com**)。
   c. 選擇您要在其中建立儲存貯體的區域。

      建議您選擇接近您地理位置的區域以充分降低延遲及成本，或因應法規要求。您選擇的區域會決定您的 Amazon S3 網站端點。如需詳細資訊，請參閱「網站端點」。
   d. 若要接受預設設定並建立儲存貯體，請選擇 Create (建立)。
3. 建立您的子網域儲存貯體：
   a. 選擇 Create bucket (建立儲存貯體)。
   b. 輸入 Bucket name (儲存貯體名稱) (例如，**www.example.com**)。
   c. 選擇您要在其中建立儲存貯體的區域。

      建議您選擇接近您地理位置的區域以充分降低延遲及成本，或因應法規要求。您選擇的區域會決定您的 Amazon S3 網站端點。如需詳細資訊，請參閱「網站端點」。
   d. 若要接受預設設定並建立儲存貯體，請選擇 Create (建立)。

在下一個步驟中，您為針對託管設定 example.com 網站。

步驟 3：設定網站託管的根網域儲存貯體

在此步驟中，您會將根網域儲存貯體 (example.com) 設定為網站。此儲存貯體將包含您的網站內容。當您配置用於網站託管的儲存貯體時，您可以使用 網站端點 存取網站。

啟用靜態網站託管

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體名稱) 清單中，選擇希望為其啟用靜態網站託管的儲存貯體名稱。
3. 選擇 Properties (屬性)。
4. 在 Static website hosting (靜態網站託管) 下，選擇 Edit (編輯)。
5. 選擇 Use this bucket to host a website (使用此儲存貯體來託管網站)。
6. 在 Static website hosting (靜態網站託管) 下，選擇Enable (啟用)。
7. 在索引文件中，輸入索引文件的名稱，通常是 `index.html`。

   索引文件名稱區分大小寫，而且必須完全符合您計畫上傳至 S3 儲存貯體的 HTML 索引文件檔案名稱。當您為網站託管設定儲存貯體時，必須指定索引文件。在對根網域或任何子資料夾提出請求時，Amazon S3 會傳回此索引文件。如需詳細資訊，請參閱「設定索引文件」。
8. 若要為 4XX 類別錯誤提供自己的自訂錯誤文件，請在 Error document (錯誤文件) 中輸入自訂錯誤文件檔案名稱。

   錯誤文件名稱區分大小寫，而且必須完全符合您計畫上傳至 S3 儲存貯體的 HTML 錯誤文件檔案名稱。如果您未指定自訂錯誤文件且發生錯誤，則 Amazon S3 會傳回預設的 HTML 錯誤文件。如需詳細資訊，請參閱 設定自訂錯誤文件。
9. (選用) 如果您要指定進階重新導向規則，請在 Redirection rules (重新導向規則) 中輸入 JSON 來描述規則。

   例如，您可依據要求中特定的物件金鑰名稱或字首，依條件路由要求。如需詳細資訊，請參閱「配置重新引導規則以使用進階條件重新引導」。
10. 選擇 Save changes (儲存變更)。

    Amazon S3 會為您的儲存貯體啟用靜態網站託管。在頁面底部的靜態網站託管下，您會看到儲存貯體的網站端點。
11. 在 Static website hosting 下，請記下 Endpoint (端點)。

    端點是儲存貯體的 Amazon S3 網站端點。將儲存貯體設為靜態網站之後，您可以使用此端點來測試您的網站。

編輯封鎖公有存取設定並新增允許公有讀取存取權的儲存貯體政策之後，您可以使用網站端點存取您的網站。

在下一個步驟中，您會設定自己的子網域 (www.example.com) 以將請求重新導向至您的網域 (example.com)。

步驟 4：設定用於網站重新導向的子網域儲存貯體

設定用於網站託管的根網域儲存貯體之後，您可以設定子網域儲存貯體以將所有請求重新導向至網域。在此範例中，www.example.com 的所有請求都會重新導向至 example.com。

設定重新導向要求

1. 在 Amazon S3 主控台中，於儲存貯體清單中選擇子網域儲存貯體 (在本例中為 www.example.com)。
2. 選擇 Properties (屬性)。
3. 在 Static website hosting (靜態網站託管) 下，選擇 Edit (編輯)。
4. 選擇 Redirect requests for an object (重新引導物件請求)。
5. 在 Target bucket (目標儲存貯體) 方塊中，輸入您的根網域，例如 **example.com**。
6. 在 Protocol (通訊協定)中，選擇 http。
7. 選擇 Save changes (儲存變更)。

步驟 5：為網站流量設定記錄

如果您想要追蹤存取您網站的訪客人數，則可以選擇性地為根網域儲存貯體啟用記錄。如需詳細資訊，請參閱 使用伺服器存取記錄記錄要求。如果您打算使用 Amazon CloudFront 加快網站速度，也可以使用 CloudFront 日誌記錄。

為您的根網域儲存貯體啟用伺服器存取記錄

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在建立設為靜態網站之儲存貯體的相同區域中，建立用於記錄的儲存貯體，例如 logs.example.com。
3. 為伺服器存取記錄日誌檔建立資料夾 (例如，logs)。
4. (選擇性) 如果您想 CloudFront 要用來改善網站效能，請為 CloudFront 記錄檔建立資料夾 (例如cdn)。

> **Important**
>
> 當您建立或更新發行版並啟用 CloudFront 記錄時，請 CloudFront 更新值區存取控制清單 (ACL)，以授與`awslogsdelivery`帳戶將記錄寫入值區的`FULL_CONTROL`權限。如需詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的設定標準記錄和存取日誌檔所需的權限。如果儲存記錄的儲存貯體使用 S3 物件擁有權強制執行的儲存貯體擁有者設定來停用 ACL，則CloudFront 無法將記錄寫入儲存貯體。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

5. 在儲存貯體清單中，選擇您的根網域儲存貯體。
6. 選擇 Properties (屬性)。
7. 在 Server access logging (伺服器存取記錄) 下，選擇 Edit (編輯)。
8. 選擇 Enable (啟用)。
9. 在目的地儲存貯體下，選擇伺服器存取日誌的儲存貯體和資料夾目的地：
   - 瀏覽至資料夾和儲存貯體位置：
     1. 選擇 Browse S3 (瀏覽 S3)。
     2. 選擇儲存貯體名稱，然後選擇日誌資料夾。
     3. 選擇 Choose path (選擇路徑)。
   - 輸入 S3 儲存貯體路徑，例如 `s3://logs.example.com/logs/`。
10. 選擇 Save changes (儲存變更)。

    在您的日誌儲存貯體中，您現在可以存取您的日誌。Amazon S3 會每隔 2 小時，將網站存取日誌寫入您的日誌儲存貯體。

步驟 6：上傳索引和網站內容

在此步驟中，您將索引文件和選用的網站內容上傳至根網域儲存貯體。

當您為儲存貯體啟用靜態網站託管時，請輸入索引文件的名稱 (例如，**`index.html`**)。為儲存貯體啟用靜態網站託管後，您可以將含有索引文件名稱的 HTML 檔案上傳到儲存貯體。

設定索引文件

1. 建立 `index.html` 檔案。

如果您還沒有 index.html 檔案，您可以使用下列HTML 建立一個檔案。

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

2. 在本機儲存索引檔案。

   索引文件檔案名稱必須完全符合您在 Static website hosting (靜態網站託管) 對話方塊中輸入的索引文件名稱。索引文件名稱有區分大小寫。例如，如果您在 Static website hosting (靜態網站託管) 對話方塊的 Index document (索引文件) 名稱中輸入 index.html，您的索引文件檔案名稱也必須是 index.html 而非 Index.html。
3. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
4. 在 Buckets (儲存貯體) 清單中，選擇您要用於託管靜態網站的儲存貯體名稱。
5. 為您的儲存貯體啟用靜態網站，然後輸入索引文件的確切名稱 (例如，index.html)。如需詳細資訊，請參閱「啟用網站託管」。

   啟用靜態網路託管之後，請繼續執行步驟 6。
6. 若要將索引文件上傳至您的儲存貯體，請執行下列其中一項：

   - 將索引檔拖放到主控台儲存貯體清單中。
   - 選擇 Upload (上傳)，然後依照提示選擇並上傳索引檔案。

   如需 step-by-step 指示，請參閱上傳物件。
7. (選用) 將其他網站內容上傳到您的儲存貯體。

步驟 7：上傳錯誤文件

當您為儲存貯體啟用靜態網站託管時，請輸入錯誤文件的名稱 (例如 **404.html**)。為儲存貯體啟用靜態網站託管後，您可以將含有錯誤引文件名稱的 HTML 檔案上傳到儲存貯體。

設定錯誤文件

1. 建立錯誤文件，例如 `404.html`。
2. 將錯誤文件檔案儲存在本機。

   錯誤文件名稱區分大小寫，且須完全符合您在啟用靜態網站託管時所輸入的名稱。例如，如果您在 Static website hosting (靜態網站託管) 對話方塊的 Error document (錯誤文件) 名稱中輸入 `404.html`，您的錯誤文件檔案名稱也必須是 `404.html`。
3. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
4. 在 Buckets (儲存貯體) 清單中，選擇您要用於託管靜態網站的儲存貯體名稱。
5. 為您的儲存貯體啟用靜態網站託管，並輸入錯誤文件的確切名稱 (例如 `404.html`)。如需詳細資訊，請參閱 啟用網站託管 及 設定自訂錯誤文件。

   啟用靜態網路託管之後，請繼續執行步驟 6。
6. 若要將錯誤文件上傳至您的儲存貯體，請執行下列其中一項：

   - 將錯誤文件檔案拖放至主控台儲存貯體清單中。
   - 選擇 Upload (上傳)，然後依照提示選擇並上傳索引檔案。

   如需 step-by-step 指示，請參閱上傳物件。

步驟 8：編輯 S3 封鎖公有存取設定

在此範例中，您可以編輯網域儲存貯體 (`example.com`) 的封鎖公有存取設定，以允許公有存取。

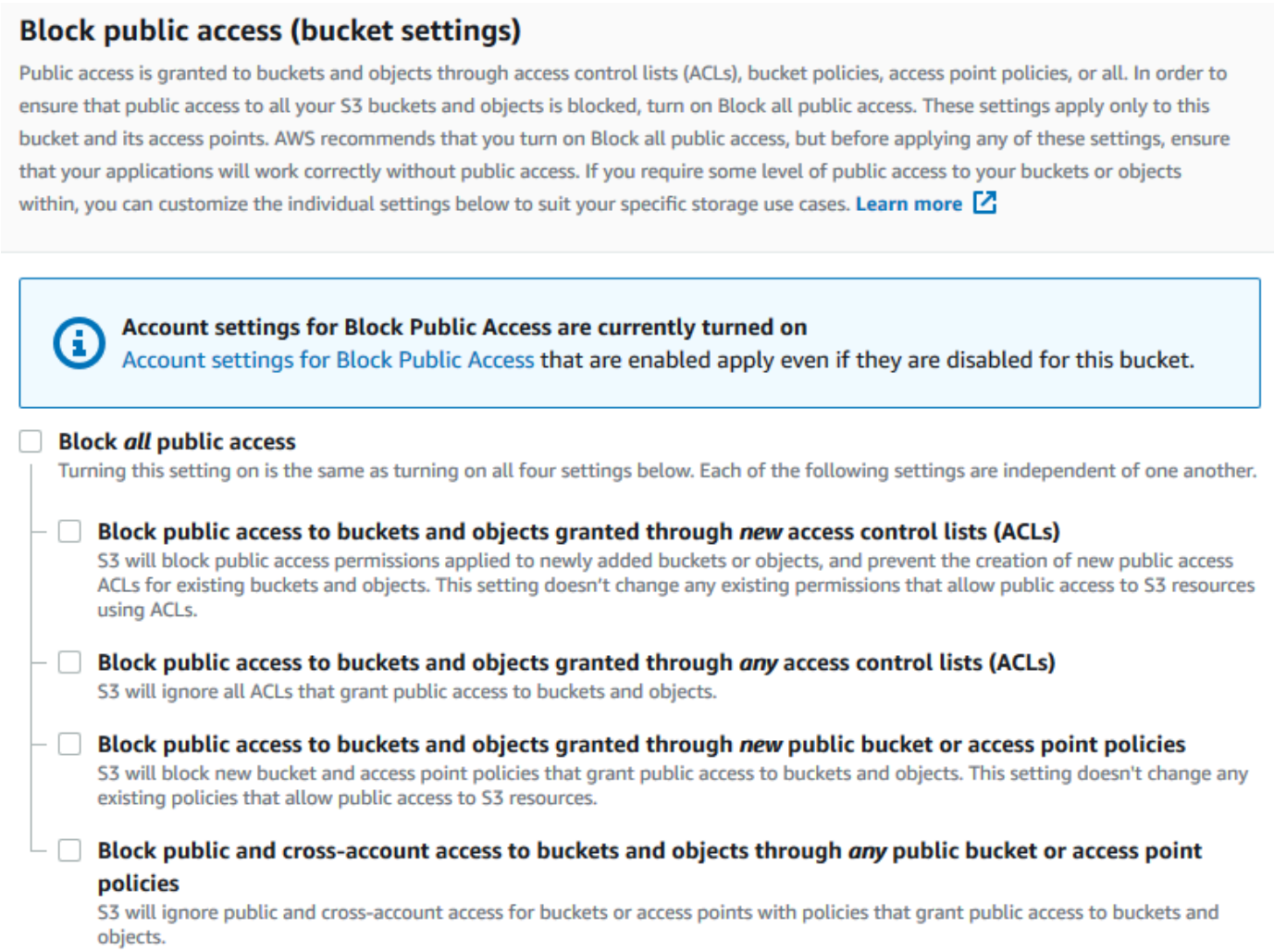
根據預設，Amazon S3 會封鎖對帳戶和儲存貯體的公開存取。如想要使用儲存貯體託管靜態網站，您可使用這些步驟編輯封鎖公有存取設定：

> **Warning**
>
> 在完成此步驟之前，請檢閱 封鎖對 Amazon S3 儲存體的公開存取權 以確保您了解並接受允許公開存取所涉及的風險。當您關閉封鎖公開存取設定以公開儲存貯體時，網際網路上的任何人都可以存取您的儲存貯體。我們建議您封鎖對儲存貯體的所有公用存取權。

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

2. 選擇已設定為靜態網站的儲存貯體名稱。
3. 選擇 Permissions (許可)。
4. 在 Block public access (bucket settings) (封鎖公開存取 (儲存貯體設定)) (封鎖公開存取 (儲存貯體設定)) 下，選擇 Edit (編輯)。
5. 清除 Block all public access (封鎖所有公開存取)，然後選擇 Save changes (儲存變更)。

> **Warning**
>
> 在完成此步驟之前，請檢閱 封鎖對 Amazon S3 儲存體的公開存取權 以確保您了解並接受允許公開存取所涉及的風險。當您關閉封鎖公開存取設定以公開儲存貯體時，網際網路上的任何人都可以存取您的儲存貯體。我們建議您封鎖對儲存貯體的所有公用存取權。



Amazon S3 會關閉儲存貯體的封鎖公開存取設定。若要建立公開的靜態網站，在新增儲存貯體原則之前，可能還需要針對您的帳戶編輯封鎖公開存取設定。如果帳戶的封鎖公開存取設定目前已開啟，您在 封鎖公開存取 (儲存貯體設定) 下會看到附註。

步驟 9：連接儲存貯體政策

在此範例中，您將儲存貯體政策連接至網域儲存貯體 (example.com)，以允許公有讀取存取權。您可以將範例儲存貯體政策中的 *Bucket-Name* 取代為網域儲存貯體的名稱，例如 example.com。

編輯 S3 封鎖公用存取設定之後，您可以新增儲存貯體政策，以授予儲存貯體的公用讀取權限。當您授予公有讀取權限時，網際網路上的任何人都可以存取您的儲存貯體。

> **Important**
>
> 以下政策僅為範例，允許完整存取您儲存貯體的內容。繼續執行此步驟之前，請檢閱如何保護 Amazon S3 儲存貯體中的檔案？，以確保您瞭解 S3 儲存貯體中檔案保護的最佳實務，以及授予公開存取權所涉及的風險。

1. 在 Buckets(儲存貯體) 下方，選擇儲存貯體的名稱。
2. 選擇 Permissions (許可)。
3. 在 Bucket Policy (儲存貯體政策) 下方，選擇 Edit (編輯)。
4. 若要授予您網站的公開讀取存取權，請複製以下儲存貯體政策，並將它貼上至 Bucket policy editor (儲存貯體政策編輯器)。

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

```
        }
    ]
}
```

5. 將 `Resource` 更新為您的儲存貯體名稱。

   在上述範例儲存貯體政策中，*`Bucket-Name`* 是儲存貯體名稱的預留位置。若要使用此儲存貯體策略與您自己的儲存貯體搭配，您必須更新此名稱以符合您的儲存貯體名稱。

6. 選擇 Save changes (儲存變更)。

   顯示的訊息指出已成功新增儲存貯體原則。

   如果您看到指出 `Policy has invalid resource` 的錯誤，請確認儲存貯體政策中的儲存貯體名稱與您的儲存貯體名稱相符。如需有關新增儲存貯體原則的資訊，請參閱如何新增 S3 儲存貯體原則？

   如果您收到錯誤訊息且無法儲存貯體原則，請檢查您的帳戶和儲存貯體的封鎖公開存取設定，以確認您允許公開存取儲存貯體。

在接下來的步驟中，您可以找出自己網站的端點，並測試自己網域的端點。

步驟 10：測試您的網域端點

將網域儲存貯體設定為託管公有網站後，您就可以測試自己的端點。如需詳細資訊，請參閱「網站端點」。您只能測試網域儲存貯體的端點，因為子網域儲存貯體已設置為網站重新導向，且不是靜態網站託管。

> **Note**
>
> Amazon S3 不支援使用 HTTPS 存取網站。如果您想使用 HTTPS，則可以使用 Amazon CloudFront 為 Amazon S3 上託管的靜態網站提供服務。
> 如需詳細資訊，請參閱如 CloudFront 何使用在 Amazon S3 上託管的靜態網站提供服務？ 以及要求 HTTPS 才能在檢視者和 CloudFront.

1. 在 Buckets (儲存貯體) 下方，選擇儲存貯體的名稱。
2. 選擇 Properties (屬性)。
3. 在頁面底部的 Static website hosting (靜態網站託管) 下，選擇您的 Bucket website endpoint (儲存貯體網站端點)。

您的索引文件會在不同的瀏覽器視窗中開啟。

在下一個步驟中，您會使用 Amazon Route 53，讓客戶可以同時使用您的自訂 URL 來導覽至您的網站。

步驟 11：為您的網域和子網域新增別名記錄

在此步驟中，您會建立可新增至網域對應 `example.com` 與 `www.example.com` 的託管區域的別名記錄。別名記錄使用 Amazon S3 網站端點，而不是使用 IP 位址。Amazon Route 53 會在別名記錄與 Amazon S3 儲存貯體所在的 IP 地址之間保持對應。您會建立兩個別名記錄，一個為根網域使用，另一個為子網域使用。

為根網域和子網域新增別名記錄

新增根網域 (**`example.com`**) 的別名記錄

1. 請在 https://console.aws.amazon.com/route53/開啟 Route 53 主控台。

   > **Note**
   >
   > 如果您尚未使用 Route 53，請參閱《Amazon Route 53 開發人員指南》中的步驟 1：註冊網域。完成設定後，您可依說明繼續作業。

2. 選擇 Hosted Zones (託管區域)。
3. 在託管區域的清單中，選擇與您網域名稱相同的託管區域名稱。
4. 選擇建立記錄。
5. 選擇 Switch to wizard (切換至精靈)。

   > **Note**
   >
   > 如果您想要使用快速建立來建立別名記錄，請參閱設定 Route 53 以將流量路由傳送至 S3 儲存貯體。

6. 選擇簡易路由，然後選擇下一步。
7. 選擇定義簡易記錄。
8. 在 Record name (記錄名稱) 中，接受預設值，這是您的託管區域與網域的名稱。

9. 在 Value/Route traffic to (值/路由傳送流量至) 中，選擇 Alias to S3 website endpoint (連至 S3 網站端點的別名)。
10. 選擇 區域。
11. 選擇 S3 儲存貯體。

    儲存貯體名稱應與 Name (名稱) 方塊中顯示的名稱相符。在 Choose S3 bucket (選擇 S3 儲存貯體) 清單中，儲存貯體名稱會與建立儲存貯體之區域的 Amazon S3 網站端點一起顯示，例如 `s3-website-us-west-1.amazonaws.com (example.com)`。

    如果下列情況，則 Choose S3 bucket (選擇 S3 儲存貯體) 會列出儲存貯體：

    - 您將儲存貯體設定為靜態網站。
    - 儲存貯體與您要建立的記錄同名。
    - 目前 AWS 帳戶 建立的值區。

    如果您的儲存貯體未出現在 Choose S3 bucket (選擇 S3 儲存貯體) 清單中，請輸入建立儲存貯體之區域的 Amazon S3 網站端點，例如 **`s3-website-us-west-2.amazonaws.com`**。如需 Amazon S3 網站端點的完整清單，請參閱 Amazon S3 網站端點。如需別名目標的詳細資訊，請參閱《Amazon Route 53 開發人員指南》中的值/將流量路由到。
12. 在 [記錄類型] 中，選擇 [A]-將流量路由至 IPv4 位址和某些 AWS 資源。
13. 在 Evaluate target health (評估目標運作狀態) 中，選擇 No (否)。
14. 選擇定義簡易記錄。

新增子網域 (**`www.example.com`**) 的別名記錄

1. 在 Configure records (設定記錄) 下，選擇 Define simple record (定義簡易記錄)。
2. 在子網域的 Record name (記錄名稱) 中，輸入 `www`。
3. 在 Value/Route traffic to (值/路由傳送流量值) 中，選擇 Alias to S3 website endpoint (連至 S3 網站端點的別名)。
4. 選擇區域。
5. 選擇 S3 儲存貯體，例如 `s3-website-us-west-2.amazonaws.com (www.example.com)`。

    如果您的儲存貯體未出現在 Choose S3 bucket (選擇 S3 儲存貯體) 清單中，請輸入建立儲存貯體之區域的 Amazon S3 網站端點，例如 **`s3-website-us-west-2.amazonaws.com`**。如需

Amazon S3 網站端點的完整清單，請參閱 Amazon S3 網站端點。如需別名目標的詳細資訊，請參閱《Amazon Route 53 開發人員指南》中的值/將流量路由到。

6. 在 [記錄類型] 中，選擇 [A]-將流量路由至 IPv4 位址和某些 AWS 資源。
7. 在 Evaluate target health (評估目標運作狀態) 中，選擇 No (否)。
8. 選擇 Define simple record (定義簡易記錄)。
9. 在 Configure records (設定記錄) 頁面上，選擇 Create records (建立記錄)。

> **Note**
>
> 變更通常會在 60 秒內傳播至所有 Route 53 伺服器。當傳播完成，您就可以使用在此程序中建立的別名記錄名稱，將流量路由到 Amazon S3 儲存貯體。

新增根網域和子網域的別名記錄 (舊版 Route 53 主控台)

新增根網域 (**example.com**) 的別名記錄

Route 53 主控台已重新設計。在 Route 53 主控台中，您可以暫時使用舊版主控台。如果您選擇使用舊版 Route 53 主控台，請使用下列程序。

1. 請在 https://console.aws.amazon.com/route53/開啟 Route 53 主控台。

   > **Note**
   >
   > 如果您尚未使用 Route 53，請參閱《Amazon Route 53 開發人員指南》中的步驟 1：註冊網域。完成設定後，您可依說明繼續作業。

2. 選擇 Hosted Zones (託管區域)。
3. 在託管區域的清單中，選擇與您網域名稱相同的託管區域名稱。
4. 選擇 Create Record Set (建立記錄集)。
5. 指定下列值：

   名稱

   接受預設值，這是您的託管區域與網域的名稱。

   如為根網域，您不需要在 Name (名稱) 欄位中輸入任何額外的資訊。

類型

選擇 A – IPv4 地址。

別名

選擇 Yes (是)。

別名目標

在清單的 S3 website endpoints (S3 網站端點) 區段中，選擇您的儲存貯體名稱。

儲存貯體名稱應與 Name (名稱) 方塊中顯示的名稱相符。例如，在別名目標清單中，儲存貯體名稱後面是建立儲存貯體區域的 Amazon S3 網站端點，例如 `example.com (s3-website-us-west-2.amazonaws.com)`。若為下列情況，Alias Target (別名目標) 會列出儲存貯體：

- 您將儲存貯體設定為靜態網站。
- 儲存貯體與您要建立的記錄同名。
- 目前 AWS 帳戶 建立的值區。

如果您的儲存貯體未出現在別名目標列表中，請輸入建立儲存貯體之區域的 Amazon S3 網站端點，例如 `s3-website-us-west-2`。如需 Amazon S3 網站端點的完整清單，請參閱 Amazon S3 網站端點。如需別名目標的詳細資訊，請參閱《Amazon Route 53 開發人員指南》中的值/將流量路由到。

路由政策

接受 Simple (簡易) 的預設值。

評估目標運作狀態

接受預設值 No (否)。

6. 選擇 Create (建立)。

新增子網域 (**`www.example.com`**) 的別名記錄

1. 在根網域 (`example.com`) 的託管區域裝，選擇 Create Record Set (建立記錄集)。
2. 指定下列值：

名稱

如為子網域，請在方塊中輸入 `www`。

類型

選擇 A – IPv4 地址。

別名

選擇 Yes (是)。

別名目標

在清單的 S3 網站端點部分中，選擇與名稱欄位中顯示的相同儲存貯體名稱 (例如，`www.example.com (s3-website-us-west-2.amazonaws.com)`)。

路由政策

接受 Simple (簡易) 的預設值。

評估目標運作狀態

接受預設值 No (否)。

3. 選擇 Create (建立)。

> **Note**
>
> 變更通常會在 60 秒內傳播至所有 Route 53 伺服器。當傳播完成，您就可以使用在此程序中建立的別名記錄名稱，將流量路由到 Amazon S3 儲存貯體。

步驟 12：測試網站

確認網站及重新導向可正常運作。在瀏覽器中輸入您的 URL。在此範例中，您可以嘗試以下 URL：

- 網域 (`http://example.com`) – 顯示 `example.com` 儲存貯體中的索引文件。
- 子網域 (`http://www.example.com`) – 將您的要求重新導向到 `http://example.com`。您在 `example.com` 儲存貯體中查看索引文件。

如果您的網站或重新導向連結無法運作，您可以嘗試下列方法：

- Clear cache (清除快取) – 清除網頁瀏覽器的快取。
- Check name servers (檢查名稱伺服器) – 如果您的網頁和重新導向連結在清除快取之後無法運作，您可以比較網域的名稱伺服器，以及託管區域的名稱伺服器。如果名稱伺服器不相符，您可能需要更

新網域名稱伺服器，以符合您託管區域下列出的名稱伺服器。如需詳細資訊，請參閱新增或變更網域的名稱伺服器和黏附記錄。

成功測試根網域和子網域之後，您可以設定 Amazon CloudFront 分發來改善網站效能，並提供可用來檢閱網站流量的記錄。如需詳細資訊，請參閱 加快您的網站與 Amazon CloudFront。

加快您的網站與 Amazon CloudFront

您可以使用 Amazon CloudFront 來提高您的 Amazon S3 網站的性能。 CloudFront 讓您的網站檔案 (例如 HTML、影像和視訊) 可從世界各地的資料中心取得 (稱為邊緣位置)。當訪客從您的網站要求檔案時， CloudFront會自動將要求重新導向至最近邊緣位置的檔案副本。比起訪客向更遠處的資料中心要求內容，如此可以縮短下載時間。

CloudFront 在您指定的一段時間內在節點快取內容。如果訪客要求快取時間超過到期日的內容，請 CloudFront 檢查原始伺服器以查看是否有較新版本的內容可用。如果有更新的版本，請將新版本 CloudFront 複製到邊緣位置。您對原始內容所做的變更，會在訪客要求內容時複寫到節點。

CloudFront 不使用 Route 53

此頁面上的教學課程使用 Route 53 來指向您的 CloudFront 分佈。不過，如果您想要在不使用 Route 53 的 CloudFront 情況下使用 Amazon S3 儲存貯體託管的內容，請參閱 Amazon CloudFront 教學：為 Amazon S3 設定動態內容分發。使用提供在 Amazon S3 儲存貯體中託管的內容時 CloudFront，您可以使用任何儲存貯體名稱，同時支援 HTTP 和 HTTPS。

使用範本自動化設定 AWS CloudFormation

如需有關使用 AWS CloudFormation 範本設定安全靜態網站以建立 CloudFront 分發以為您的網站提供服務的詳細資訊，請參閱 Amazon 開 CloudFront 發人員指南中的安全靜態網站入門。

主題

- 步驟 1：建立 CloudFront 分發
- 步驟 2：更新網域與子網域的記錄集
- (選用) 步驟 3：檢查日誌檔案

步驟 1：建立 CloudFront 分發

首先，您要建立 CloudFront 分發。如此一來，即可從全球各地的資料中心取得您的網站。

使用 Amazon S3 來源伺服器建立分發

1. 在開啟 CloudFront 主控台https://console.aws.amazon.com/cloudfront/v4/home。
2. 選擇 Create Distribution (建立分佈)。
3. 在建立分發頁面的來源設定區段中，針對來源伺服器網域名稱，輸入您儲存貯體的 Amazon S3 網站端點，例如 **`example.com.s3-website.us-west-1.amazonaws.com`**。

   CloudFront 為您填入原點 ID。
4. 針對 Default Cache Behavior Settings (預設快取行為設定)，請保留預設值。

   透過檢視器通訊協定政策的預設設定，您可以將 HTTPS 用於您的靜態網站。如需這些組態選項的詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的建立或更新 Web 分發時指定的值。
5. 針對 Distribution Settings (分佈設定)，請執行下列作業：

   a. 將 Price Class (價格方案) 維持在設為 Use All Edge Locations (Best Performance) (使用所有節點 (最佳效能))。

   b. 將替代網域名稱 (CNAME) 設定為根網域和 www 子網域。在本教學課程中，有 `example.com` 和 `www.example.com`。

      > **Important**
      >
      > 在執行此步驟前，請記下使用替代網域名稱的需求，尤其是有效 SSL/TLS 認證的需求。

   c. 對於 SSL 憑證，請選擇自訂 SSL 憑證 (example.com)，然後選擇涵蓋網域和子網域名稱的自訂憑證。

      如需詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的 SSL 憑證。

   d. 在 Default Root Object (預設根物件) 中，輸入索引文件的名稱，例如 `index.html`。

      如果用於存取分發的 URL 不包含檔案名稱，則 CloudFront 分發會傳回索引文件。Default Root Object (預設根物件) 應該與您靜態網站的索引文件名稱完全相符。如需詳細資訊，請參閱 設定索引文件。

   e. 將 Logging (記錄日誌) 設為 On (開啟)。

> **Important**
>
> 當您建立或更新發行版並啟用 CloudFront 記錄時，請 CloudFront 更新值區存取控制清單 (ACL)，以授與`awslogsdelivery`帳戶將記錄寫入值區的`FULL_CONTROL`權限。如需詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的設定標準記錄和存取日誌檔所需的權限。如果儲存記錄的儲存貯體使用 S3 物件擁有權強制執行的儲存貯體擁有者設定來停用 ACL，則CloudFront 無法將記錄寫入儲存貯體。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

f. 針對 Bucket for Logs (日誌儲存貯體)，選擇所建立的日誌記錄儲存貯體。

   如需配置日誌儲存貯體的詳細資訊，請參閱 (選y用) 記錄 Web 流量。

g. 如果您要將流量產生的記錄儲存在資料夾中 CloudFront ，請在記錄前置詞中輸入資料夾名稱。

h. 將其他所有設定保留為其預設值。

6. 選擇 Create Distribution (建立分佈)。
7. 若要查看分佈的狀態，請在主控台中尋找分佈並檢查 Status (狀態) 欄。

   `InProgress` 狀態表示尚未完整部署分散。

   部署發行版後，您可以使用新的 CloudFront網域名稱來參考您的內容。
8. 記錄主 CloudFront 控台中顯示的網域名稱值，例如`dj4p1rv6mvubz.cloudfront.net`。
9. 若要驗證您的 CloudFront 發行版是否正常運作，請在網頁瀏覽器中輸入發行版的網域名稱。

   如果您的網站可見，則 CloudFront 分發可以正常工作。如果您的網站有在 Amazon Route 53 註冊的自訂網域，您將需要網 CloudFront 域名稱來更新下一個步驟中設定的記錄。

步驟 2：更新網域與子網域的記錄集

現在您已成功建立 CloudFront 發行版，請更新 Route 53 中的別名記錄，以指向新的 CloudFront 發行版。

更新別名記錄以指向 CloudFront 分佈的步驟

1. 請在 https://console.aws.amazon.com/route53/開啟 Route 53 主控台。
2. 在左側導覽窗格中，選擇 Hosted zones (託管區域)。

3. 在 Hosted Zones (託管區域) 頁面上，選擇您為子網域所建立的託管區域，例如 www.example.com。
4. 在 Records (記錄) 下，選取您為子網域建立的 A 記錄。
5. 在 Record details (記錄詳細資訊)下，選擇 Edit record (編輯記錄)
6. 在「路由流量至」下，選擇「要 CloudFront 散佈的別名」。
7. 在「選擇分配」下，選擇 CloudFront 分配。
8. 選擇儲存。
9. 若要將根網域的 A 記錄重新導向至 CloudFront發佈，請針對根網域重複此程序，例如example.com。

   對記錄集所做的更新，會在 2–48 小時內生效。
10. 若要查看新的 A 記錄是否已生效，請在網頁瀏覽器中輸入您的子網域 URL，例如 http://www.example.com。

    如果瀏覽器不再將您重新導向至根網域 (例如，http://example.com)，則表示 A 記錄已生效。當新 A 記錄生效時，由新 A 記錄路由至 CloudFront 發佈的流量不會重新導向至根網域。任何透過使用http://example.com或重新導向至最近的 CloudFront 節點來參考網站的訪客，在http://www.example.com這裡他們將受益於更快的下載時間。

> **Tip**
>
> 瀏覽器可快取重新導向設定。若您認為新的 A 記錄設定應該已經生效，但您的瀏覽器仍將 http://www.example.com 重新導向至 http://example.com，請嘗試清除瀏覽器的歷史記錄與快取、關閉瀏覽器應用程式再重新開啟，或使用不同的 Web 瀏覽器。

(選用) 步驟 3：檢查日誌檔案

存取日誌能告知您瀏覽網站的人數。這些日誌也包含重要的商務資料，您可以使用 Amazon EMR 等其他服務進行分析。

CloudFront 記錄會儲存在您建立 CloudFront分發並啟用記錄時所選擇的值區和資料夾中。 CloudFront 在發出對應請求後的 24 小時內將日誌寫入日誌存儲桶。

查看網站的日誌檔

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

2. 選擇適用於您網站的記錄儲存貯體。
3. 選擇記 CloudFront 錄檔資料夾。
4. 在打開文.gzip件 CloudFront 之前下載它們寫入的文件。

若您建立的網站僅供學習練習，則可先刪除原先已配置的資源，如此即不會再增加費用。若要這麼做，請參閱清理範例資源。刪除 AWS 資源之後，您的網站即不再可供使用。

清理範例資源

若建立的靜態網站僅供學習練習之用，您應先刪除已配置的 AWS 資源，如此即不會再產生費用。刪除 AWS 資源之後，您的網站即不再可供使用。

任務

- 步驟 1：刪除 Amazon CloudFront 分佈
- 步驟 2：刪除 Route 53 託管區域
- 步驟 3：停用記錄並刪除 S3 儲存貯體

步驟 1：刪除 Amazon CloudFront 分佈

刪除 Amazon CloudFront 分發之前，必須先將其停用。已停用的分佈如此即不再有用，也不會產生費用。您隨時都可以啟用之前停用的分佈。刪除停用的分佈之後，它即不再可供使用。

若要停用和刪除散 CloudFront 佈

1. 在開啟 CloudFront 主控台https://console.aws.amazon.com/cloudfront/v4/home。
2. 選取您要停用的分佈，然後選擇 Disable (停用)。
3. 出現確認提示時，請選擇 Yes, Disable (是，停用)。
4. 選取已停用的分佈，然後選擇 Delete (刪除)。
5. 出現確認提示時，選擇 Yes, Delete (是，刪除)。

步驟 2：刪除 Route 53 託管區域

刪除託管區域之前，必須先刪除您建立的記錄集。您不必刪除 NS 和 SOA 記錄，它們會在您刪除託管區域時自動刪除。

刪除記錄集

1. 請在 https://console.aws.amazon.com/route53/開啟 Route 53 主控台。
2. 從網域名稱清單中，選取您的網域名稱，然後選擇 Go to Record Sets (前往記錄集)。
3. 在記錄集清單中，選擇您建立的 A 記錄。

   每個記錄集的類型都會列在 Type (類型) 欄中。
4. 選擇 Delete Record Set (刪除記錄集)。
5. 出現確認提示時，選擇 Confirm (確認)。

刪除 Route 53 託管區域

1. 繼續前一程序，選擇 Back to Hosted Zones (返回託管區域)。
2. 選取您的網域名稱，然後選擇 Delete Hosted Zone (刪除託管區域)。
3. 出現確認提示時，選擇 Confirm (確認)。

步驟 3：停用記錄並刪除 S3 儲存貯體

刪除 S3 儲存貯體之前，請務必停用儲存貯體的記錄。否則，當您刪除值區時，會 AWS 繼續將記錄寫入值區。

停用儲存貯體的記錄

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在 Buckets (儲存貯體) 下，選擇儲存貯體名稱，然後選擇 Properties (屬性)。
3. 從 Properties (屬性) 選擇 Logging (記錄日誌)。
4. 清除 Enabled (已啟用) 核取方塊。
5. 選擇 Save (儲存)。

現在即可刪除儲存貯體。如需詳細資訊，請參閱 刪除儲存貯體。

建立、設定和使用 Amazon S3 儲存貯體

在將資料存放在 Amazon S3 中時，您會使用稱為儲存貯體和物件的資源。儲存貯體是物件的容器。物件是一個檔案和任何描述該檔案的中繼資料。

若要將物件存放在 Amazon S3 中，您需要建立儲存貯體，然後將物件上傳到儲存貯體。當物件在儲存貯體中時，您可以開啟、下載和移動它。當您不再需要物件或儲存貯體時，可以清理這些資源。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱什麼是 S3 Express One Zone？ 和 目錄值區。

> **Note**
>
> 使用 Amazon S3，您只需按實際用量付費。如需 Amazon S3 功能和定價的詳細資訊，請參閱 Amazon S3。若您是 Amazon S3 新客戶，可以免費試用 Amazon S3。如需詳細資訊，請參閱 AWS 免費方案。

本節中的主題提供在 Amazon S3 中使用儲存貯體的概觀，包含命名、建立、存取和刪除貯體的相關資訊。如需檢視或列出儲存貯體中物件的詳細資訊，請參閱「整理、列出和使用物件」。

主題

- 儲存貯體概觀
- 儲存貯體命名規則
- 存取及列出 Amazon S3 儲存貯體
- 建立儲存貯體
- 檢視 S3 儲存貯體的屬性
- 清空儲存貯體
- 刪除儲存貯體
- 對 Amazon S3 儲存貯體設定預設伺服器端加密行為
- 使用適用於 Amazon S3 的掛載點
- 使用 Amazon S3 Transfer Acceleration 設定快速安全的檔案傳輸

- 使用儲存體傳輸和用量的申請者付款儲存貯體
- 儲存貯體的法規與限制

儲存貯體概觀

若要將相片、影片、文件等資料上傳至 Amazon S3，您必須先在其中一個 AWS 區域中建立 S3 儲存貯體。

儲存貯體是 Amazon S3 中用於存放物件的容器。您可以在儲存貯體中存放任意數目的物件，並且帳戶中最多可有 100 個儲存貯體。若要請求增加，請造訪 Service Quotas 主控台。

每個物件都包含在儲存貯體中。例如，如果名為 `photos/puppy.jpg` 的物件存放在美國西部 (奧勒岡) 區域的 `DOC-EXAMPLE-BUCKET` 儲存貯體中，則可以使用 URL `https://DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com/photos/puppy.jpg` 定址。如需詳細資訊，請參閱存取儲存貯體。

在實作方面，儲存貯體和物件是 AWS 資源，Amazon S3 提供 API 供您管理這些資源。例如，您可以使用 Amazon S3 API 建立儲存貯體並上傳物件。或者，您也可以使用 Amazon S3 主控台執行這些操作，主控台就會使用 Amazon S3 API 將請求傳送給 Amazon S3。

本節說明如何使用儲存貯體。如需使用物件的資訊，請參閱「Amazon S3 物件概觀」。

Amazon S3 支援全域儲存貯體，這表示每個儲存貯體名稱 AWS 帳戶 在分割區 AWS 區域 內的所有名稱都必須是唯一的。分割區是區域的群組。 AWS 目前有三個分割區：`aws`(標準區域)、`aws-cn`(中國區域) 和 `aws-us-gov` (AWS GovCloud (US))。

建立值區之後，在刪除值區之前，相同分割區 AWS 帳戶 中的另一個儲存貯體無法使用該儲存貯體的名稱。建議您不要為了可用性或安全驗證目的而依賴特定的儲存貯體命名慣例。如需儲存貯體命名準則，請參閱「儲存貯體命名規則」。

Amazon S3 會在您指定的區域建立儲存貯體。若要減少延遲、將成本降至最低或滿足法規要求，請選擇地理位置靠近您的任何 AWS 區域 項目。舉例而言，如果您住在歐洲，則在歐洲 (愛爾蘭) 或歐洲 (法蘭克福) 區域建立儲存貯體可能較有利。如需 Amazon S3 區域的清單，請參閱《AWS 一般參考》中的區域與端點。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱什麼是 S3 Express One Zone？ 和 目錄值區。

> **Note**
>
> 屬於您在特定值區中建立的物件 AWS 區域 永遠不會離開該區域，除非您明確將它們轉移到另一個區域。例如，存放在歐洲 (愛爾蘭) 區域中的物件絕對會保留在此區域。

主題

關於許可

您可以使用 AWS 帳戶根使用者 登入資料建立儲存貯體並執行任何其他 Amazon S3 操作。不過，我們建議您不要使用您的 root 使用者認證 AWS 帳戶 來提出要求，例如建立值區。而是建立 AWS Identity and Access Management (IAM) 使用者，並授予該使用者完整存取權限 (預設情況下，使用者沒有權限)。

這些使用者稱為「管理員」。您可以使用系統管理員使用者認證 (而非帳戶的根使用者認證) 與其互動 AWS 並執行工作，例如建立值區、建立使用者，以及授予他們權限。

如需詳細資訊，請參閱《AWS 一般參考》中的AWS 帳戶根使用者 憑證與 IAM 使用者憑證以及《IAM 使用者指南》中的 IAM 安全最佳實務。

建立 AWS 帳戶 資源的擁有該資源。例如，如果您在中建立 IAM 使用者， AWS 帳戶 並授與使用者建立值區的權限，則該使用者可以建立值區。但是使用者並不擁有該值區； AWS 帳戶 使用者所屬的擁有值區。使用者需要資源擁有者授予其他許可，才能執行任何其他儲存貯體操作。如需管理 Amazon S3 資源許可的詳細資訊，請參閱 適用於 Amazon S3 的 Identity and Access Management。

管理儲存貯體的公開存取

透過儲存貯體政策、存取控制清單 (ACL) 或兩者來授予對儲存貯體和物件的公開存取。為協助您管理對 Amazon S3 資源的公開存取，Amazon S3 提供了封鎖公開存取權限設定。Amazon S3 封鎖公開存取權限設定可以覆寫 ACL 與儲存貯體政策，方便您對這些資源的公開存取強制執行統一限制。您可套用封鎖公開存取設定到您帳戶中的個別儲存貯體或全部儲存貯體。

若要確保封鎖對所有 Amazon S3 儲存貯體和物件的公開存取，在您建立新的儲存貯體時，預設會啟用「封鎖公開存取」的所有四個設定。建議您也為帳戶開啟「封鎖公開存取」的所有四個設定：這些設定會封鎖所有現有和未來儲存貯體的一切公開存取。

套用這些設定前，請確認應用程式無須公用存取權限也可正常運作。如果儲存貯體或物件需要特定層級的公開存取權限 (例如 使用 Amazon S3 託管靜態網站 中所述的託管靜態網站)，您可配合儲存體使用案例自訂個別設定。如需詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

不過，高度建議將「封鎖公開存取」保持啟用狀態。如果您想要保持啟用所有四個「封鎖公用存取」設定並託管靜態網站，可以使用 Amazon CloudFront 來源存取控制 (OAC)。Amazon CloudFront 提供設定安全靜態網站所需的功能。Amazon S3 靜態網站只支援 HTTP 端點。亞馬遜 CloudFront使用 Amazon S3 的耐用儲存，同時提供額外的安全標頭，例如 HTTPS。HTTPS 透過加密一般 HTTP 請求並防止常見的網路攻擊來增加安全性。

如需詳細資訊，請參閱 Amazon 開 CloudFront 發人員指南中的安全靜態網站入門。

> **Note**
>
> 如果您在列出儲存貯體及其公用存取設定時看到 `Error`，則您可能沒有所需的許可。確保您已將下列許可新增至您的使用者或角色政策：
>
> ```
> s3:GetAccountPublicAccessBlock
> s3:GetBucketPublicAccessBlock
> s3:GetBucketPolicyStatus
> s3:GetBucketLocation
> s3:GetBucketAcl
> s3:ListAccessPoints
> s3:ListAllMyBuckets
> ```
>
> 在極少數狀況下，請求也可能因 AWS 區域 中斷而失敗。

儲存貯體組態選項

Amazon S3 支援各種設定儲存貯體的選項。例如，您可以設定儲存貯體處理網站託管、新增組態來管理儲存貯體中的物件生命週期，以及設定儲存貯體記錄所有對儲存貯體的存取。此外，Amazon S3 也支援子資源，讓您可以存放及管理儲存貯體組態資訊。您可以使用 Amazon S3 API 來建立和管理這些子資源。不過，您也可以使用主控台或 AWS SDK。

> **Note**
>
> 也有物件層級的組態。例如，您可以設定該物件的專用存取控制清單 (ACL)，來設定物件層級許可。

這些會當成子資源參考，因為它們存在於特定的儲存貯體或物件內容中。下表列出子資源，讓您管理儲存貯體專用組態。

子資源	描述
cors (跨來源資源共享)	您可以設定儲存貯體允許跨來源要求。 如需詳細資訊，請參閱「使用跨來源資源分享 (CORS)」。
event notification	您可以讓儲存貯體傳送指定儲存貯體事件的通知。 如需詳細資訊，請參閱 Amazon S3 事件通知。
lifecycle	您可以為已妥善定義生命週期的儲存貯體物件，定義生命週期規則。例如，您可以定義一項規則，建立物件後將它封存一年，或建立物件 10 年後將它刪除。 如需詳細資訊，請參閱「管理儲存生命週期」。
位置	建立儲存貯體時，請指 AWS 區域 定 Amazon S3 建立儲存貯體的位置。Amazon S3 會將此資訊存放在「location」子資源中，並提供可擷取此資訊的 API。
logging	記錄日誌能讓您追蹤存取儲存貯體的要求。每筆存取日誌記錄都會提供有關單一存取要求的詳細資訊，例如要求者、儲存貯體名稱、要求時間、要求動作、回應狀態及錯誤代碼 (若出現錯誤)。存取記錄資訊在安全與存取稽核中相當實用。您也可藉由該資訊了解自己的客戶群，並掌握 Amazon S3 帳單相關資料。 如需詳細資訊，請參閱「使用伺服器存取記錄記錄要求」。
物件鎖定	您必須為儲存貯體啟用 S3 物件鎖定，才能使用該功能。您也可以選擇設定預設保留模式和保留期，並套用至位於儲存貯體中的新物件。

子資源	描述
	如需詳細資訊，請參閱 使用 S3 物件鎖定。
policy 與 ACL (存取控制清單)	所有資源 (例如儲存貯體與物件) 預設都是私有的。Amazon S3 支援儲存貯體政策與存取控制清單 (ACL) 選項，可讓您授予及管理儲存貯體層級許可。Amazon S3 會將許可資訊存放在「policy」與「acl」子資源中。 如需詳細資訊，請參閱 適用於 Amazon S3 的 Identity and Access Management。
複寫	複寫是跨不同或相同 AWS 區域的儲存貯體自動非同步的物件複製。如需詳細資訊，請參閱 複製物件概觀。
requestPayment	根據預設 AWS 帳戶 ，建立值區的 (值區擁有者) 會支付值區的下載費用。使用此子資源，儲存貯體擁有者可以指定要求下載的人支付下載的費用。Amazon S3 提供 API，讓您管理此子資源。 如需詳細資訊，請參閱「使用儲存體傳輸和用量的申請者付款儲存貯體」。
tagging	您可以在儲存貯體中新增成本分配標籤，以分類和追蹤 AWS 成本。Amazon S3 提供的「tagging」子資源可讓您存放及管理儲存貯體上的標籤。使用套用至儲存貯體的標籤， AWS 產生成本分配報告，其中包含依標籤彙總的使用情況和成本。 如需詳細資訊，請參閱 Amazon S3 的帳單和用量報告。
transfer acceleration	Transfer Acceleration 可讓用戶端與 S3 儲存貯體間的長距離檔案傳輸作業變得迅速、簡單又安全。傳輸加速充分利用 Amazon 全球分佈的節點 CloudFront。 如需詳細資訊，請參閱 使用 Amazon S3 Transfer Acceleration 設定快速安全的檔案傳輸。
versioning	版本控制能幫助您復原意外的覆寫與刪除。 建議使用版本控制作為最佳實務，以復原錯誤刪除或覆寫的物件。 如需詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。

子資源	描述
website	您可以設定您的儲存貯體處理靜態網站託管。Amazon S3 會透過建立「website」子資源來存放此組態。 如需詳細資訊，請參閱 使用 Amazon S3 託管靜態網站。

儲存貯體命名規則

以下是命名 Amazon S3 中的一般用途儲存貯體和目錄儲存貯體適用的規則：

主題

一般用途儲存貯體命名規則

以下是一般用途儲存貯體適用的命名規則：

- 儲存貯體名稱長度必須介於 3 (最小值) 到 63 (最大值) 個字元之間。
- 儲存貯體名稱只能由小寫字母、數字、句點 (.) 和連字號 (-) 組成。
- 儲存貯體名稱的開頭和結尾必須為字母或數字。
- 儲存貯體名稱不能包含兩個連續句點。
- 儲存貯體名稱絕不能格式化為 IP 位址 (例如，192.168.5.4)。
- 儲存貯體名稱必須以字首 `xn--` 開頭。
- 值區名稱不得以前綴`sthree-`或前綴開頭`sthree-configurator`。
- 儲存貯體名稱不得以尾碼 `-s3alias` 結尾。存取點別名名稱會保留此尾碼。如需詳細資訊，請參閱針對您的 S3 儲存貯體存取點使用儲存貯體樣式別名。
- 儲存貯體名稱不得以尾碼 `--ol-s3` 結尾。Object Lambda 存取點別名名稱會保留此尾碼。如需詳細資訊，請參閱 如何針對您的 S3 儲存貯體 Object Lambda 存取點使用儲存貯體樣式別名。
- 值區名稱 AWS 帳戶 在所有分區中都必須是唯一的 AWS 區域 。分割區是區域的群組。 AWS 目前有三個分割區：`aws`(標準區域)、`aws-cn` (中國地區) 和 `aws-us-gov` (AWS GovCloud (US))。
- AWS 帳戶 在刪除值區之前，儲存貯體名稱不能被同一個分割區中的其他人使用。

- 與 Amazon S3 Transfer Acceleration 搭配使用的儲存貯體名稱中不能有句點 (.)。如需 Transfer Acceleration 的詳細資訊，請參閱 使用 Amazon S3 Transfer Acceleration 設定快速安全的檔案傳輸。

為了獲得最佳相容性，建議您避免在儲存貯體名稱中使用句點 (.)，但僅用於靜態網站託管的儲存貯體除外。如果儲存貯體名稱中包含點，則除非您執行自己的憑證驗證，否則無法透過 HTTPS 使用 virtual-host-style 定址。這是因為用於儲存貯體虛擬託管的安全憑證不適用於名稱中含有句點的儲存貯體。

此限制不會影響用於靜態網站託管的儲存貯體，因為靜態網站託管只能透過 HTTP 使用。如需 virtual-host-style定址的更多資訊，請參閱儲存貯體的虛擬託管。如需靜態網站託管的詳細資訊，請參閱 使用 Amazon S3 託管靜態網站。

> **Note**
>
> 2018 年 3 月 1 日之前，在美國東部 (維吉尼亞北部) 區域中建立的儲存貯體名稱長度最多可為 255 個字元，且包含大寫字母和底線。自 2018 年 3 月 1 日開始，美國東部 (維吉尼亞北部) 區域中的新儲存貯體必須符合套用至所有其他區域的相同規則。

有關物件金鑰名稱資訊，請參閱建立物件金鑰名稱。

一般用途儲存貯體名稱範例

下列範例儲存貯體名稱有效，並遵循建議的一般用途儲存貯體命名指導方針：

- `docexamplebucket1`
- `log-delivery-march-2020`
- `my-hosted-content`

下列範例儲存貯體名稱有效，但不建議用於靜態網站託管以外的用途：

- `docexamplewebsite.com`
- `www.docexamplewebsite.com`
- `my.example.s3.bucket`

下列範例儲存貯體名稱無效：

- `doc_example_bucket` (包含底線)
- `DocExampleBucket` (包含大寫字母)
- `doc-example-bucket-` (結尾為連字號)

目錄儲存貯體命名規則

目錄儲存貯體名稱必須：

- 在所選區域 AWS 區域 和可用區域內是唯一的。
- 名稱長度必須介於 3 (最小) 到 63 個 (最多) 個字元之間，包括後置字元。
- 只能由小寫字母、數字和連字號 (-) 組成。
- 開頭和結尾為字母或數字。
- 必須包含以下後綴：`--`*`azid`*`--x-s3`。

> **Note**
>
> 當您使用主控台建立目錄值區時，字尾會自動新增至您提供的基本名稱。此字尾包含您所選擇可用區域的可用區域 ID。
> 使用 API 建立目錄儲存貯體時，必須在要求中提供完整尾碼，包括可用區域 ID。如需可用區域 ID 的清單，請參閱S3 Express One Zone 可用區域和區域。

存取及列出 Amazon S3 儲存貯體

若要列出及存取 Amazon S3 儲存貯體，您可以使用各種工具。檢閱下列工具，以判斷哪種方法適合您的使用案例：

- Amazon S3 主控台：透過 Amazon S3 主控台，您可以輕鬆存取儲存貯體及修改儲存貯體的屬性。您也可以使用主控台 UI 執行多數的儲存貯體操作，而無須撰寫任何程式碼。
- AWS CLI：如果您需要存取多個值區，可以使用 AWS Command Line Interface (AWS CLI) 自動執行常見和重複性工作，以節省時間。隨著組織的擴展，常見動作能否製成指令碼和可否重複使用，是常見的考量。如需詳細資訊，請參閱 使用 AWS CLI 來透過 Amazon S3 進行開發。
- Amazon S3 REST API：您可以使用 Amazon S3 REST API 撰寫您自己的程式，並以程式設計方式存取儲存貯體。Amazon S3 支援將儲存貯體與物件視為資源的 API 架構，其各自都有可唯一識別資源的資源 URI。如需詳細資訊，請參閱 使用 REST API 與 Amazon S3 進行開發。

根據 Amazon S3 儲存貯體的使用案例，存取儲存貯體中的基礎資料時適用不同的建議方法。下列清單包含存取資料的常見使用案例。

- 靜態網站 – 您可以使用 Amazon S3 託管靜態網站。在此使用案例中，您可以將 S3 儲存貯體設定為像網站一般運作。如需逐步在 Amazon S3 上託管網站的範例，請參閱 教學課程：在 Amazon S3 上設定靜態網站。

  若要託管啟用封鎖公用存取等安全設定的靜態網站，我們建議您使用 CloudFront 具有原始存取控制 (OAC) 的 Amazon，並實作其他安全標頭，例如 HTTPS。如需詳細資訊，請參閱安全靜態網站入門。

  > **Note**
  >
  > Amazon S3 支援將虛擬託管樣式和路徑樣式 URL 用於靜態網站存取。由於儲存貯體可透過路徑型 URL 與虛擬託管型 URL 存取，建議您使用與 DNS 相容的儲存貯體名稱建立儲存貯體。如需詳細資訊，請參閱 儲存貯體的法規與限制。

- 共用資料集 – 當您在 Amazon S3 上進行擴展時，常會採用多租戶模型；在這種模式中，您可以將不同的終端客戶或業務單位指派給共用儲存貯體內的唯一字首。藉由使用 Amazon S3 Access Points，您可以針對需要存取共用資料集的每個應用程式，將一個大型儲存貯體政策劃分為個別、獨立的存取點政策。此方法可讓您更輕鬆地專注於為應用程式建置正確的存取政策，而不會影響到任何其他應用程式在共用資料集內執行的動作。如需詳細資訊，請參閱 使用 Amazon S3 存取點管理資料存取。
- 高輸送量工作負載 – 適用於 Amazon S3 的掛載點是高輸送量的開放原始碼檔案用戶端，可將 Amazon S3 儲存貯體掛載為本機檔案系統。使用掛載點可讓您的應用程式透過像是開啟和讀取等檔案系統操作，存取儲存在 Amazon S3 中的物件。掛載點會自動將這些操作轉換成 S3 物件 API 呼叫，讓您的應用程式透過檔案介面存取 Amazon S3 的彈性儲存和輸送量。如需詳細資訊，請參閱 使用適用於 Amazon S3 的掛載點。
- 多區域應用程式 – Amazon S3 多區域存取點可提供全域端點，其中應用程式可用來滿足來自多個 AWS 區域的 S3 儲存貯體的請求。您可以使用多區域存取點，進而使用與單一區域中使用的相同架構來建置多區域應用程式，然後在全球任何地方執行這些應用程式。多區域存取點不會透過公有網際網路傳送請求，而是提供內建的網路恢復能力，並加速對 Amazon S3 的網際網路請求。如需詳細資訊，請參閱 Amazon S3 中的多區域存取點。
- 建立新的應用程式 — 您可以在使用 Amazon S3 開發應用程式時使用開發 AWS 套件。這些 AWS 開發套件透過包裝基礎的 Amazon S3 REST API 來簡化您的程式設計任務。若要建置連線的行動和 Web 應用程式，您可以使用 AWS 行動 SDK 和程式 AWS Amplify JavaScript 庫。如需詳細資訊，請參閱 使用開發 AWS 套件使用 Amazon S3 進行開發。

- 安全殼層 (SSH) 檔案傳輸通訊協定 (SFTP) — 如果您嘗試透過網際網路安全地傳輸敏感資料，可以在 Amazon S3 儲存貯體使用已啟用 SFTP 的伺服器。 AWS SFTP 是一種支援 SSH 完整安全性和驗證功能的網路通訊協定。透過此通訊協定，您可以更精細地控制使用者身分、許可和金鑰，或是使用 IAM 政策來管理存取權。若要將已啟用 SFTP 的伺服器與 Amazon S3 儲存貯體建立關聯，請務必先建立已啟用 SFTP 的伺服器。然後，請設定使用者帳戶，並將伺服器與 Amazon S3 儲存貯體建立關聯。如需此程序的逐步解說，請參閱AWS Transfer for SFTP AWS 部落格中適用於 Amazon S3 的全受管 SFTP 服務。

列出儲存貯體

若要列出您所有的儲存貯體，您必須具有 `s3:ListAllMyBuckets` 許可。若要存取值區，請務必同時取得必要的 AWS Identity and Access Management (IAM) 許可，以列出指定值區的內容。如需授予 S3 儲存貯體存取權的範例儲存貯體政策，請參閱 允許 IAM 使用者存取其中一個儲存貯體。如果發生 HTTP 存取遭拒 (403 禁止) 錯誤，請參閱 儲存貯體政策與 IAM 政策。

您可以使用 Amazon S3 主控台、或 AWS 開發套件列出儲存貯體。 AWS CLI

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 從「一般用途值區」清單中，選擇您要檢視的值區。

> **Note**
>
> 「一般用途值區」清單包含位於所有值區的值區 AWS 區域。

使用 AWS CLI

若要使用 AWS CLI 存取 S3 儲存貯體或產生 S3 儲存貯體的清單，請使用`ls`命令。列出儲存貯體中的所有物件時請注意，您必須擁有 `s3:ListBucket` 許可。

若要使用此範例命令，請將 *`DOC-EXAMPLE-BUCKET1`* 取代為您的儲存貯體名稱。

```
$ aws s3 ls s3://DOC-EXAMPLE-BUCKET1
```

下列範例命令會列出您的帳戶中所有的 Amazon S3 儲存貯體。

```
$ aws s3 ls
```

如需詳細資訊和範例，請參閱列出儲存貯體和物件。

使用 AWS 軟體開發套件

您也可以使用 ListBuckets API 操作來存取 Amazon S3 儲存貯體。如需如何搭配不同 AWS SDK 使用此作業的範例，請參閱搭ListBuckets配 AWS 開發套件或 CLI 使用。

建立儲存貯體

若要將資料上傳到 Amazon S3，您必須先在其中一個 AWS 區域中建立 Amazon S3 儲存貯體。建立儲存貯體時，必須選擇儲存貯體名稱與區域。您可以選擇性地為儲存貯體選擇其他儲存管理選項。建立儲存貯體後，便無法變更儲存貯體名稱或區域。如需命名儲存貯體的資訊，請參閱「儲存貯體命名規則」。

創 AWS 帳戶 建存儲桶的擁有它。然後可以上傳任意數目的物件到儲存貯體。根據預設，您可以在每個儲存貯體中建立最多 100 個儲存貯體 AWS 帳戶。如需更多的儲存貯體，您可以提交服務限制增加，將您的帳戶儲存貯體限制增加至 1,000 個儲存貯體的上限。若要進一步了解如何提交請求來提高儲存貯體限制，請參閱《AWS 一般參考資料》中的 AWS 服務 配額。您可以在每個儲存貯體內存放任意物件數量。

S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來同時控制上傳至儲存貯體之物件的擁有權，以及停用或啟用存取控制清單 (ACL)。根據預設，物件擁有權設定為「儲存貯體擁有者強制執行」設定，而且所有 ACL 都會停用。停用 ACL 後，儲存貯體擁有者會擁有儲存貯體中的每個物件，並使用政策專門管理對資料的存取。

如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 是 Amazon S3 中每個儲存貯體的基本加密組態層級。上傳到 S3 儲存貯體的所有新物件都會以 SSE-S3 做為基本加密設定層級，來進行自動加密。若您想使用不同類型的預設加密，也可以使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 或客戶提供的金鑰 (SSE-C)，指定伺服器端加密來為您的資料加密。如需詳細資訊，請參閱 對 Amazon S3 儲存貯體設定預設伺服器端加密行為。

您可以使用 Amazon S3 主控台、Amazon S3 API 或 AWS 開發套件建立儲存貯體。 AWS CLI如需建立儲存貯體所需許可的詳細資訊，請參閱 Amazon 簡單儲存服務 API 參考CreateBucket中的。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在頁面頂端的導覽列中，選擇目前顯示的名稱 AWS 區域。接下來，選擇您要在其中建立值區的「區域」。

   > **Note**
   >
   > 請選擇接近您的區域，以充分降低延遲及成本，並因應法規要求。除非您明確地將存放在區域中的物件傳輸到其他區域，否則物件絕對不會離開該區域。如需 Amazon S3 的清單 AWS 區域，AWS 服務 請參閱 Amazon Web Services 一般參考.

3. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
4. 選擇 Create bucket (建立儲存貯體)。

   Create bucket (建立儲存貯體) 頁面隨即開啟。
5. 在 [一般設定] 下方，檢視 AWS 區域 儲存貯體的建立位置。
6. 在「鏟斗類型」下選擇「一般用途」。
7. 針對 Bucket name (儲存貯體名稱)，輸入儲存貯體的名稱。

   儲存貯體名稱必須；

   - 在分割區內不重複。分割區是區域的群組。 AWS 目前有三個分割區：aws(標準區域)、aws-cn(中國區域) 和 aws-us-gov (AWS GovCloud (US) Regions)。
   - 長度必須介於 3 與 63 個字元之間。
   - 只能由小寫字母、數字、句點 (.) 和連字號 (-) 組成。為了獲得最佳相容性，建議您避免在儲存貯體名稱中使用句點 (.)，但僅用於靜態網站託管的儲存貯體除外。
   - 開頭和結尾為字母或數字。

   建立儲存貯體後，便無法變更其名稱。如需儲存貯體命名的詳細資訊，請參閱 儲存貯體命名規則。

> **Important**
>
> 避免在儲存貯體名稱中包含敏感資訊，例如帳戶號碼。在指向儲存貯體中之物件的 URL 中，會顯示儲存貯體名稱。

8. AWS Management Console 可讓您將現有值區的設定複製到新值區。如果您不想複製現有值區的設定，請跳至下一個步驟。

> **Note**
>
> 此選項：
>
> - 無法在中使用， AWS CLI 且只能在主控台中使用
> - 不適用於目錄值區
> - 不會將儲存貯體政策從現有值區複製到新值區

若要複製現有值區的設定，請在 [從現有值區複製設定] 底下，選取 [選擇值區]。「選擇時段」視窗即會開啟。找出包含您要複製之設定的值區，然後選取 [選擇值區]。「選擇時段」視窗即會關閉，且「建立時段」視窗會重新開啟。

在「從現有值區複製設定」 下方，您現在會看到所選值區的名稱。您也會看到 [還原預設值] 選項，可用來移除複製的值區設定。檢閱「建立值區」頁面上的剩餘值區設定。您將看到它們現在與您選擇的存儲桶的設置匹配。您可以跳到最後一步。

9. 在 Object Ownership (物件擁有權) 下，若要停用或啟用 ACL 並控制上傳在儲存貯體中物件的擁有權，請選擇下列其中一個設定：

   已停用 ACL

   - 儲存貯體擁有者強制執行 (預設) - 停用 ACL，儲存貯體擁有者會自動擁有並完全控制儲存貯體中的每個物件。ACL 不再影響 S3 儲存貯體中資料的存取許可。儲存貯體單獨使用政策來定義存取控制。

     根據預設，會停用 ACL。Amazon S3 中的大多數新式使用案例不再需要使用 ACL。建議您將 ACL 保時停用狀態，除非在異常情況下必須個別控制每個物件的存取。如需詳細資訊，請參閱控制物件的擁有權並停用儲存貯體的 ACL。

已啟用 ACL

- 儲存貯體擁有者偏好 – 儲存貯體擁有者擁有並完全控制其他帳戶使用 bucket-owner-full-control 標準 ACL 寫入儲存貯體的新物件。

  如果您套用儲存貯體擁有者偏好設定，以要求所有 Amazon S3 上傳都包含 bucket-owner-full-control 固定 ACL 時，您可以新增儲存貯體政策，只允許使用此 ACL 的物件上傳。
- 物件寫入器 — 上傳物件的物件擁有物件、擁有物件的完全控制權，並且可以透過 ACL 授與其他使用者存取 AWS 帳戶 該物件。

> **Note**
>
> 預設設定為儲存貯體擁有者強制執行。若要套用預設設定並將 ACL 保持停用狀態，只需要 s3:CreateBucket 許可。若要啟用 ACL，您必須具有 s3:PutBucketOwnershipControls 許可。

10. 在封鎖此儲存貯體的公開存取設定之下，選擇要套用至儲存貯體的封鎖公開存取設定。

    根據預設，會啟用全部四個「封鎖公開存取」設定。建議您將所有設定保持啟用狀態，除非您知道需要針對特定使用案例關閉其中一或多個設定。如需封鎖公開存取的詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

    > **Note**
    >
    > 若要啟用所有「封鎖公用存取」設定，只需要 s3:CreateBucket 許可。若要關閉任何「封鎖公開存取」設定，您必須具有 s3:PutBucketPublicAccessBlock 許可。

11. (選用) 在 Bucket Versioning (儲存貯體版本控制) 下，您可以選擇是否要在儲存貯體中保留物件的變體。如需版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。

    若要對儲存貯體停用或啟用版本控制，請選擇 Disable (停用) 或 Enable (啟用)。

12. (選用) 在 Tags (標籤) 下，您可以選擇新增標籤至儲存貯體。標籤是用來分類儲存的鍵值對。

    若要新增儲存貯體標籤，請輸入 Key (金鑰) 並選擇性地輸入 Value (值)，然後選擇 Add tag (新增標籤)。

13. 在 Default encryption (預設加密) 底下，選擇 Edit (編輯)。

14. 若要設定預設加密，請在加密類型下，選擇下列其中一項：

    - Amazon S3 受管金鑰 (SSE-S3)
    - AWS Key Management Service 金鑰 (SSE-公里)

    > **Important**
    >
    > 如果您針對預設加密組態使用 SSE-KMS 選項，則受到 AWS KMS的每秒請求數目 (RPS) 限制。如需有關 AWS KMS 配額以及如何要求提高配額的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的配額。

    儲存貯體和新物件會以 Amazon S3 受管金鑰做為基本加密組態層級，使用伺服器端加密。如需預設加密的詳細資訊，請參閱對 Amazon S3 儲存貯體設定預設伺服器端加密行為。

    如需有關使用 Amazon S3 伺服器端加密來加密資料的詳細資訊，請參閱「使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密」。

15. 若您選擇 AWS Key Management Service 金鑰 (SSE-KMS)，請執行下列操作：

    a. 在 AWS KMS 金鑰下，使用下列其中一種方式指定 KMS 金鑰：

       - 若要從可用 KMS 金鑰清單中選擇，請選擇從您的金鑰中選擇 AWS KMS keys，然後從可用金鑰清單中選擇您的 KMS 金鑰。

         AWS 受管金鑰 (aws/s3) 和您的客戶管理金鑰都會出現在此清單中。如需詳細了解客戶受管金鑰，請參閱《AWS Key Management Service 開發人員指南》中的客戶金鑰和 AWS 金鑰。
       - 若要輸入 KMS 金鑰 ARN，請選擇輸入 AWS KMS key ARN，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。
       - 若要在 AWS KMS 主控台中建立新的客戶管理金鑰，請選擇 [建立 KMS 金鑰]。

         如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。

       > **Important**
       >
       > 您只能使用與值區相同 AWS 區域 的 KMS 金鑰。Amazon S3 主控台僅會列出與儲存貯體位於相同區域的前 100 個 KMS 金鑰。若要使用未列出的 KMS 金鑰，必須輸

入 KMS 金鑰 ARN。若您想要使用其他帳戶的 KMS 金鑰，您必須先具有該金鑰的使用權限，然後輸入 KMS 金鑰 ARN。如需詳細了解 KMS 金鑰跨帳戶權限，請參閱《AWS Key Management Service 開發人員指南》中的建立其他帳戶可使用的 KMS 金鑰。如需 SSE-KMS 的詳細資訊，請參閱 使用 AWS KMS (SSE-KMS) 指定伺服器端加密。

當您在 Amazon S3 中使 AWS KMS key 用伺服器端加密時，您必須選擇對稱加密 KMS 金鑰。Amazon S3 僅支援對稱加密 KMS 金鑰，而不支援非對稱 KMS 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的識別對稱和非對稱 KMS 金鑰。

如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。如需 AWS KMS 搭配 Amazon S3 搭配使用的詳細資訊，請參閱使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)。

b. 當您將儲存貯體設定為使用 SSE-KMS 的預設加密時，您還可以啟用 S3 儲存貯體金鑰。S3 儲存貯體金鑰可將 Amazon S3 的請求流量減少到 AWS KMS。如需詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本。

若要使用 S3 儲存貯體金鑰，在 Bucket Key (儲存貯體金鑰) 下選擇 Enable (啟用)。

16. (選用) 如果您想要啟用 S3 物件鎖定，請執行下列動作：

a. 選擇 Advanced settings (進階設定)。

> **Important**
>
> 啟用物件鎖定也會啟用儲存貯體的版本控制。啟用後，您必須設定「物件鎖定」預設保留和合法保留設定，以防止新物件遭到刪除或覆寫。

b. 如果想要啟用物件鎖定，請選擇 Enable (啟用)、讀取出現的警告並確認。

如需詳細資訊，請參閱 使用 S3 物件鎖定。

> **Note**
>
> 若要建立已啟用物件鎖定的儲存貯體，您必須具備下列許可：`s3:CreateBucket`、`s3:PutBucketVersioning` 和 `s3:PutBucketObjectLockConfiguration`。

17. 選擇建立儲存貯體。

使用 AWS 軟體開發套件

使用 AWS SDK 建立值區時，必須建立用戶端，然後使用用戶端傳送要求以建立值區。最佳實務是，您應該在同一個 AWS 區域中建立用戶端和儲存貯體。如果您在建立用戶端或儲存貯體時未指定區域，則 Amazon S3 會使用預設的美國東部 (維吉尼亞北部) 區域。如果您將儲存貯體建立限於特定 AWS 區域，請使用 `LocationConstraint` 條件金鑰。

若要建立用戶端來存取雙堆疊端點，您必須指定 AWS 區域。如需詳細資訊，請參閱 雙堆疊端點。如需可用的清單 AWS 區域，請參閱中的區域和端點AWS 一般參考。

建立用戶端時，區域會對應至區域特定的端點。用戶端會使用此端點來與 Amazon S3：`s3.region.amazonaws.com` 通訊。如果您的區域在 2019 年 3 月 20 日之後啟動，您的用戶端和儲存貯體必須位於相同區域中。不過，您可以在美國東部 (維吉尼亞北部) 區域中使用用戶端，以便在 2019 年 3 月 20 日之前推出的任何區域中建立儲存貯體。如需詳細資訊，請參閱 舊版端點。

這些 AWS SDK 程式碼範例會執行下列工作：

- 透過明確指定 AWS 區域 來建立用戶端 – 在此範例中，用戶端使用 `s3.us-west-2.amazonaws.com` 端點與 Amazon S3 通訊。您可指定任何 AWS 區域。如需的清單 AWS 區域，請參閱AWS 一般參考中的區域和端點。
- 透過僅指定儲存貯體名稱來傳送建立儲存貯體 要求 — 用戶端會向 Amazon S3 傳送要求，以在您建立用戶端的區域中建立儲存貯體。
- 擷取儲存貯體位置的相關資訊 — Amazon S3 會將儲存貯體位置資訊存放在與儲存貯體相關聯的 location 子資源中。

Java

本範例示範如何使用 AWS SDK for Java建立 Amazon S3 儲存貯體。如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CreateBucketRequest;
import com.amazonaws.services.s3.model.GetBucketLocationRequest;

import java.io.IOException;

public class CreateBucket2 {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            if (!s3Client.doesBucketExistV2(bucketName)) {
                // Because the CreateBucketRequest object doesn't specify a region,
 the
                // bucket is created in the region specified in the client.
                s3Client.createBucket(new CreateBucketRequest(bucketName));

                // Verify that the bucket was created by retrieving it and checking
 its
                // location.
                String bucketLocation = s3Client.getBucketLocation(new
 GetBucketLocationRequest(bucketName));
                System.out.println("Bucket location: " + bucketLocation);
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
```

```
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

如需如何建立和測試工作範例的詳細資訊，請參閱 .NET 第 3 版 API 參考的AWS SDK。

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.S3.Util;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CreateBucketTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CreateBucketAsync().Wait();
        }

        static async Task CreateBucketAsync()
        {
            try
            {
                if (!(await AmazonS3Util.DoesS3BucketExistAsync(s3Client,
 bucketName)))
                {
                    var putBucketRequest = new PutBucketRequest
                    {
```

```
                        BucketName = bucketName,
                        UseClientRegion = true
                    };

                    PutBucketResponse putBucketResponse = await
 s3Client.PutBucketAsync(putBucketRequest);
                }
                // Retrieve the bucket location.
                string bucketLocation = await FindBucketLocationAsync(s3Client);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
        static async Task<string> FindBucketLocationAsync(IAmazonS3 client)
        {
            string bucketLocation;
            var request = new GetBucketLocationRequest()
            {
                BucketName = bucketName
            };
            GetBucketLocationResponse response = await
 client.GetBucketLocationAsync(request);
            bucketLocation = response.Location.ToString();
            return bucketLocation;
        }
    }
}
```

Ruby

如需有關如何建立和測試工作範例的詳細資訊，請參 AWS SDK for Ruby-第 3 版。

Example

```
require "aws-sdk-s3"
```

```
# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                 create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      "None. You must create a bucket before you can get its location!"
    else
      @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("doc-example-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)
```

```
  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

使用 AWS CLI

您也可以使用 AWS Command Line Interface (AWS CLI) 建立 S3 儲存貯體。如需詳細資訊，請參閱《AWS CLI 命令參考》中的 create-bucket。

如需有關的資訊 AWS CLI，請參閱什麼是 AWS Command Line Interface? 在《AWS Command Line Interface 使用者指南》中。

檢視 S3 儲存貯體的屬性

您可以檢視您擁有的任何 Amazon S3 儲存貯體的屬性。這些設定包含下列項目：

- Bucket Versioning (儲存貯體版本控制) – 使用版本控制在一個儲存貯體中保留物件的多個版本。根據預設，新的儲存貯體會停用版本控制。如需啟用版本控制的資訊，請參閱「在儲存貯體上啟用版本控制」。
- 標籤 — 透過 AWS 成本分配，您可以使用儲存貯體標籤來註解帳單，以便使用儲存貯體。標籤為一組金鑰/值對，代表指派給儲存貯體的標籤。如需詳細資訊，請參閱 使用成本分配 S3 儲存貯體標籤。
- 預設加密 – 啟用預設加密可為您提供伺服器端自動加密。Amazon S3 會在將物件儲存到磁碟之前加密，並在下載物件時解密。如需更多詳細資訊，請參閱 對 Amazon S3 儲存貯體設定預設伺服器端加密行為。
- Server access logging (伺服器存取日誌記錄) – 使用伺服器存取日誌記錄，以取得對儲存貯體所提出請求的詳細記錄。Amazon S3 預設不會收集伺服器存取日誌。如需啟用伺服器存取日誌的資訊，請參閱「啟用 Amazon S3 伺服器存取記錄日誌」。
- AWS CloudTrail 資料事件 — 用 CloudTrail 於記錄資料事件。根據預設，線索不會記錄資料事件。資料事件需支付額外的費用。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的日誌記錄追蹤的資料事件。
- Event notifications (事件通知) – 啟用特定的 Amazon S3 儲存貯體事件，在發生事件時將通知訊息傳送至目的地。如需詳細資訊，請參閱 使用 Amazon S3 主控台啟用和設定事件通知。

- Transfer acceleration (傳輸加速) – 可讓用戶端與 S3 儲存貯體間的長距離檔案傳輸變得迅速、簡單又安全。如需如何啟用 Transfer Acceleration 的資訊，請參閱「啟用和使用 S3 Transfer Acceleration」。
- Object Lock (物件鎖定) – 您可以使用 S3 物件鎖定，讓物件在固定期間或無限期免於遭到刪除或覆寫。如需更多詳細資訊，請參閱 使用 S3 物件鎖定。
- Requester Pays (請求者付款) – 您可以啟用請求者付款，讓請求者 (而不是儲存貯體擁有者) 支付請求與資料傳輸的費用。如需更多詳細資訊，請參閱 使用儲存體傳輸和用量的申請者付款儲存貯體。
- Static website hosting (靜態網站代管) – 您可以在 Amazon S3 上託管靜態網站。如需詳細資訊，請參閱 使用 Amazon S3 託管靜態網站。

您可以使用 AWS Management Console、 AWS CLI或 AWS SDK 檢視值區屬性

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇您要檢視屬性的儲存貯體名稱。
3. 選擇屬性索引標籤。
4. 在 [內容] 頁面上，您可以設定值區的上述屬性。

使用 AWS CLI

檢視值區屬性 AWS CLI

下列指令顯示如何使用 AWS CLI 來列出不同的值區屬性。

下列內容會傳回與值區*##-* s3-儲存貯體 1 相關聯的標籤集。如需值區標籤的詳細資訊，請參閱，使用成本分配 S3 儲存貯體標籤。

```
aws s3api get-bucket-tagging --bucket example-s3-bucket1
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 get-bucket-tagging。

以下內容會傳回值區*##-* s3-儲存區 1 的版本控制狀態。如需值區版本控制的相關資訊，請參閱在 S3 儲存貯體中使用版本控制。

```
aws s3api get-bucket-versioning --bucket example-s3-bucket1
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 get-bucket-versioning。

以下內容會傳回值區*##-* s3-bucket1 的預設加密組態。根據預設，所有儲存貯體都有預設加密組態，該組態使用伺服器端加密搭配 Amazon S3 受管金鑰 (SSE-S3)。如需值區預設加密的詳細資訊，請參閱對 Amazon S3 儲存貯體設定預設伺服器端加密行為。

```
aws s3api get-bucket-encryption --bucket example-s3-bucket1
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 get-bucket-encryption。

以下內容會傳回值區*##-* s3-儲存區 1 的通知組態。如需值區事件通知的相關資訊，請參閱Amazon S3 事件通知。

```
aws s3api get-bucket-notification-configuration --bucket example-s3-bucket1
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 get-bucket-notification-configuration。

以下內容會傳回值區*##-* s3-bucket1 的記錄狀態。如需值區記錄的相關資訊，請參閱使用伺服器存取記錄記錄要求。

```
aws s3api get-bucket-logging --bucket example-s3-bucket1
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 get-bucket-logging。

使用 AWS 軟體開發套件

有關如何使用 AWS SDK 返回值區屬性的示例，例如版本控制，標籤等，請參閱使用 AWS 開發套件為 Amazon S3 執行動作。

如需使用不同 AWS SDK 的一般資訊，請參閱使用開發 AWS 套件使用 Amazon S3 進行開發。

清空儲存貯體

您可以使用 Amazon S3 主控台、 AWS 開發套件或 AWS Command Line Interface (AWS CLI) 清空儲存貯體的內容。當清空一個儲存貯體時，裡面的所有物件將會刪除，但儲存貯體本身將會保留。清空儲存貯體後，就無法復原。清空儲存貯體動作正在進行時在儲存貯體中新增的物件可能會遭到刪除。在刪除儲存貯體之前，必須先刪除儲存貯體中的所有物件 (包括所有物件版本和刪除標記)。

當您清空已啟用或暫停 S3 版本控制的儲存貯體時，會刪除儲存貯體中所有物件的所有版本。如需詳細資訊，請參閱 使用已啟用版本控制之儲存貯體中的物件。

您也可以指定儲存貯體上的生命週期組態使物件過期，讓 Amazon S3 可以刪除這些物件。如需詳細資訊，請參閱 在值區上設定生命週期組態。若要清空大型儲存貯體，建議您使用 S3 生命週期組態規則。生命週期到期是一種非同步程序，因此規則可能需要幾天的時間才會執行，然後再將儲存貯體清空。Amazon S3 第一次執行規則後，符合到期資格的所有物件都會標記為要刪除。您不需再為標記為要刪除的物件付費。如需詳細資訊，請參閱如何使用生命週期組態規則清空 Amazon S3 儲存貯體？。

使用 S3 主控台

您可以使用 Amazon S3 主控台清空儲存貯體，這會刪除儲存貯體中的所有物件，而不需要刪除儲存貯體。

清空 S3 儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Bucket name (儲存貯體名稱) 清單中，選取要清空之儲存貯體名稱旁的選項，然後選擇 Empty (清空)。
3. 在 Empty bucket (清空儲存貯體) 頁面上，在文字欄位中輸入儲存貯體名稱以確認您要清空的儲存貯體，然後選擇 Empty (清空)。
4. 在 Empty bucket: Status (清空儲存貯體：狀態) 頁面上監控儲存貯體清空的進度。

使用 AWS CLI

僅當值區未啟用「值區啟用版本控制」時，您 AWS CLI 才可以使用來清空值區。如果未啟用版本控制，您可以使用 `rm` (remove) AWS CLI 指令搭配`--recursive`參數來清空值區 (或移除具有特定索引鍵名稱前置詞的物件子集)。

下列 `rm` 命令會移除具有索引鍵名稱字首 `doc` (例如 `doc/doc1` 與 `doc/doc2`) 的物件。

```
$ aws s3 rm s3://bucket-name/doc --recursive
```

使用下列命令會移除所有未指定字首的物件。

```
$ aws s3 rm s3://bucket-name --recursive
```

如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的搭配 AWS CLI使用高階 S3 命令。

> **Note**
>
> 您無法從已啟用版本控制的儲存貯體中移除物件。當您刪除物件時，Amazon S3 會新增刪除標記，也就是這個命令會執行的動作。如需 S3 儲存貯體版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。

使用 AWS 軟體開發套件

您可以使用 AWS SDK 清空值區，或移除具有特定金鑰名稱前置詞的物件子集。

如需如何使用清空值區的範例 AWS SDK for Java，請參閱刪除儲存貯體。此程式碼會刪除所有物件，而不論儲存貯體是否已啟用版本控制，然後它會刪除儲存貯體。若只要清空儲存貯體，請務必移除刪除儲存貯體的陳述式。

如需有關使用其他 AWS 開發套件的詳細資訊，請參閱 Amazon Web Services 的工具。

使用生命週期組態

若要清空大型儲存貯體，建議您使用 S3 生命週期組態規則。生命週期到期是一種非同步程序，因此規則可能需要幾天的時間才會執行，然後再將儲存貯體清空。Amazon S3 第一次執行規則後，符合到期資格的所有物件都會標記為要刪除。您不需再為標記為要刪除的物件付費。如需詳細資訊，請參閱如何使用生命週期組態規則清空 Amazon S3 儲存貯體？。

如果使用生命週期組態清空儲存貯體，該組態應包含目前版本、非目前版本、刪除標記和不完整的分段上傳。

您可以新增生命週期組態規則，使所有物件或具有特定索引鍵前綴的物件子集過期。例如，若要移除儲存貯體中的所有物件，您可以設定生命週期規則，使物件在建立一天之後過期。

Amazon S3 支援儲存貯體生命週期規則，您可以使用此規則來停止在啟動後指定天數內未完成的分段上傳。建議您設定此生命週期規則，將儲存成本降至最低。如需詳細資訊，請參閱「設定儲存貯體生命週期組態，以刪除不完整的分段上傳」。

如需有關使用生命週期組態來清空儲存貯體的詳細資訊，請參閱 在值區上設定生命週期組態 和 即將到期的物件。

清空已配置的存儲桶 AWS CloudTrail

AWS CloudTrail 追蹤 Amazon S3 儲存貯體中的物件層級資料事件，例如刪除物件。如果您使用值區做為記錄 CloudTrail 事件的目的地，並且要從同一個值區刪除物件，您可能會在清空值區時建立新物

件。為了防止這種情況，請停止您的 AWS CloudTrail 步道。如需有關停止追 CloudTrail 蹤記錄事件的詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的關閉追蹤記錄。

停止 CloudTrail 追蹤新增至值區的另一種替代方法是在值區政策中新增拒絕`s3:PutObject`陳述式。如果您想稍後將新物件儲存在儲存貯體中，則需要移除此拒絕 `s3:PutObject` 陳述式。如需詳細資訊，請參閱 IAM 使用者指南中的 物件操作 和 IAM JSON 政策元素：效果。

刪除儲存貯體

您可以刪除空的 Amazon S3 儲存貯體。刪除儲存貯體前，請考量下列事項：

- 儲存貯體名稱是唯一的。如果您刪除值區，則其他使用 AWS 者可以使用該名稱。
- 如果儲存貯體託管靜態網站，並且您依照「教學課程：使用向 Route 53 註冊的自訂網域設定靜態網站」中所述建立並設定 Amazon Route 53 託管區域，則必須清除與該儲存貯體相關的 Route 53 託管區域設定。如需詳細資訊，請參閱「步驟 2：刪除 Route 53 託管區域」。
- 如果儲存貯體是從 Elastic Load Balancing (ELB) 接收日誌資料：建議您先停止將 ELB 日誌傳送到儲存貯體，再予以刪除。如果另一位使用者建立與您刪除的儲存貯體相同的名稱，您的日誌資料可能會傳到該儲存貯體裡。如需 ELB 存取日誌的相關資訊，請參閱《Classic Load Balancer 使用者指南》中的存取日誌和 《Application Load Balancer 使用者指南》中的存取日誌。

故障診斷

如果您無法刪除 Amazon S3 儲存貯體，請考慮下列事項：

- 確保儲存貯體為空 – 您只能刪除其中未包含任何物件的儲存貯體。確保儲存貯體為空。
- 確保未附加任何存取點 - 您只能刪除其未附加任何存取點的儲存貯體。刪除儲存貯體之前，請先刪除任何附加至儲存貯體的存取點。
- AWS Organizations 服務控制策略（SCP）— 服務控制策略可以拒絕值區的刪除權限。如需 SCP 的詳細資訊，請參閱 《AWS Organizations 使用者指南》中的服務控制政策。
- s3：DeleteBucket 許可 — 如果您無法刪除存儲桶，請與 IAM 管理員合作以確認您具有`s3:DeleteBucket`許可。如需有關如何檢視或更新 IAM 許可的資訊，請參閱《IAM 使用者指南》中的變更 IAM 使用者的許可。
- s3：DeleteBucket 拒絕陳述式 — 如果您在 IAM 政策中擁有`s3:DeleteBucket`許可，且無法刪除儲存貯體，則儲存貯體政策可能會包含的拒絕陳述式`s3:DeleteBucket`。依預設，由建立的值區 ElasticBeanstalk 具有包含此陳述式的原則。您必須先刪除此陳述式或儲存貯體政策，才能刪除儲存貯體。

> **Important**
>
> 儲存貯體名稱是唯一的。如果您刪除值區，則其他使用 AWS 者可以使用該名稱。若希望繼續使用相同的儲存貯體名稱，請勿刪除該儲存貯體。建議您清空儲存貯體，並保留它。

使用 S3 主控台

刪除 S3 儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選取要刪除之儲存貯體名稱旁的選項，然後選擇頁面頂端的 Delete (刪除)。
3. 在 Delete bucket (刪除儲存貯體) 頁面上，在文字欄位中輸入儲存貯體名稱以確認您要刪除該儲存貯體，然後選擇 Delete bucket (刪除儲存貯體)。

   > **Note**
   >
   > 如果儲存貯體包含任何物件，請在刪除之前清空儲存貯體，方法是選取 This bucket is not empty (此儲存貯體不是空的) 錯誤警示中的清空儲存貯體組態連結，並遵循 Empty bucket (清空儲存貯體) 頁面上的指示。然後返回 Delete bucket (刪除儲存貯體) 頁面並刪除儲存貯體。

4. 若要確認您是否已刪除儲存貯體，請打開 Buckets (儲存貯體) 列表，然後輸入您已刪除的儲存貯體名稱。如果找不到儲存貯體，就表示已成功刪除。

使用適用於 Java 的 AWS 開發套件

下列範例說明如何使用 AWS SDK for Java 刪除值區。首先，此程式碼會刪除儲存貯體中的物件，然後它會刪除儲存貯體。如需其他 AWS SDK 的相關資訊，請參閱適用於 Amazon Web Services 的工具。

Java

下列 Java 範例示範刪除含有物件的儲存貯體。此範例示範刪除所有物件後，也會刪除儲存貯體。此範例也適用於已啟用版本控制或未啟用版本控制的儲存貯體。

> **Note**
>
> 對於未啟用版本控制的儲存貯體，您可以直接刪除所有物件，然後刪除儲存貯體。對於已啟用版本控制的儲存貯體，您一定需先刪除所有物件版本，然後才能刪除儲存貯體。

如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.Iterator;

public class DeleteBucket2 {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Delete all objects from the bucket. This is sufficient
            // for unversioned buckets. For versioned buckets, when you attempt to
 delete
            // objects, Amazon S3 inserts
            // delete markers for all objects, but doesn't delete the object
 versions.
            // To delete objects from versioned buckets, delete all of the object
 versions
            // before deleting
            // the bucket (see below for an example).
            ObjectListing objectListing = s3Client.listObjects(bucketName);
            while (true) {
```

```
                Iterator<S3ObjectSummary> objIter =
objectListing.getObjectSummaries().iterator();
                while (objIter.hasNext()) {
                    s3Client.deleteObject(bucketName, objIter.next().getKey());
                }

                // If the bucket contains many objects, the listObjects() call
                // might not return all of the objects in the first listing. Check
to
                // see whether the listing was truncated. If so, retrieve the next
page of
                // objects
                // and delete them.
                if (objectListing.isTruncated()) {
                    objectListing = s3Client.listNextBatchOfObjects(objectListing);
                } else {
                    break;
                }
            }

            // Delete all object versions (required for versioned buckets).
            VersionListing versionList = s3Client.listVersions(new
ListVersionsRequest().withBucketName(bucketName));
            while (true) {
                Iterator<S3VersionSummary> versionIter =
versionList.getVersionSummaries().iterator();
                while (versionIter.hasNext()) {
                    S3VersionSummary vs = versionIter.next();
                    s3Client.deleteVersion(bucketName, vs.getKey(),
vs.getVersionId());
                }

                if (versionList.isTruncated()) {
                    versionList = s3Client.listNextBatchOfVersions(versionList);
                } else {
                    break;
                }
            }

            // After all objects and object versions are deleted, delete the bucket.
            s3Client.deleteBucket(bucketName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
```

```
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
 couldn't
            // parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用 AWS CLI

AWS CLI 如果未啟用版本控制，您可以刪除包含物件的值區。當您刪除內有物件的儲存貯體時，系統也會永久刪除該儲存貯體中的所有物件，包含轉換到 S3 Glacier 儲存體方案的物件。

如果您的值區未啟用版本控制，您可以使用 rb (remove bucket) AWS CLI 指令搭配--force參數來刪除值區及其中的所有物件。此命令會先刪除所有物件，再刪除儲存貯體。

如果啟用了版本控制，則不會在此過程中刪除版本控制物件，這將導致儲存貯體刪除失敗，因為儲存貯體不會為空。如需刪除使用版本控制物件的資訊，請參閱刪除物件版本。

```
$ aws s3 rb s3://bucket-name --force
```

如需詳細資訊，請參閱使用 AWS Command Line Interface 者指南 AWS Command Line Interface中的搭配使用高階 S3 命令。

對 Amazon S3 儲存貯體設定預設伺服器端加密行為

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

根據預設，所有 Amazon S3 儲存貯體皆已設定加密，且使用伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 自動加密物件。此加密設定適用於 Amazon S3 儲存貯體中的所有物件。

如果您需要對金鑰進行更多控制，例如管理金鑰輪替和存取原則授與，您可以選擇使用伺服器端加密 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS)，或使用金鑰 (DSSE-KMS) 來使用 AWS KMS 雙層伺服器端加密。如需詳細了解編輯 KMS 金鑰，請參閱《AWS Key Management Service 開發人員指南》中的編輯金鑰。

> **Note**
>
> 我們已變更儲存貯體，以自動加密新物件上傳。若您之前建立的儲存貯體沒有預設加密，根據預設，Amazon S3 會使用 SSE-S3 為儲存貯體啟用加密。對於已設定 SSE-S3 或 SSE-KMS 的現有儲存貯體，預設加密設定不會有任何變更。若您想要使用 SSE-KMS 加密物件，則必須變更儲存貯體設定中的加密類型。如需詳細資訊，請參閱 使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)。

當您將儲存貯體設定為搭配 SSE-KMS 使用預設加密時，也可以啟用 S3 儲存貯體金鑰，將 Amazon S3 的請求流量減少到 AWS KMS 並降低加密成本。如需詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本。

若要識別已啟用 SSE-KMS 進行預設加密的儲存貯體，您可以使用 Amazon S3 Storage Lens 指標。S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。如需詳細資訊，請參閱使用 S3 Storage Lens 保護資料。

當您使用伺服器端加密時，Amazon S3 會在將物件儲存到磁碟之前加密，並在下載物件時解密。如需使用伺服器端加密與加密金鑰管理來保護資料的詳細資訊，請參閱 使用伺服器端加密保護資料。

如需預設加密所需許可的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 PutBucketEncryption。

您可以使用 Amazon S3 主控台、 AWS 開發套件、Amazon S3 REST API 和 AWS 命令列界面 (AWS CLI)，為 S3 儲存貯體設定 Amazon S3 預設加密行為。

加密現有物件

若要加密現有的未加密 Amazon S3 物件，您可以使用 Amazon S3 批次操作。請提供 S3 批次作業可操作的物件清單，批次作業就會呼叫個別 API 來執行指定的操作。您可以使用批次操作複製操作來複製現有的未加密物件，並將其作為加密物件寫入相同的儲存貯體中。單一批次操作任務可在數十億個物

件上執行指定的操作。如需詳細資訊，請參閱 在 Amazon S3 物件上執行大規模批次操作 和 AWS 儲存部落格文章《使用 Amazon S3 批次操作加密物件》。

您也可以使用 CopyObject API 作業或copy-object AWS CLI 指令來加密現有物件。如需詳細資訊，請參閱 AWS 儲存部落格文章《使用 AWS CLI加密現有的 Amazon S3 物件》。

> **Note**
>
> 如果 Amazon S3 儲存貯體的預設儲存貯體加密為 SSE-KMS，則不能作為 the section called "記錄伺服器存取" 的目的地儲存貯體。伺服器存取日誌目標儲存貯體僅支援 SSE-S3 預設加密。

針對跨帳戶操作使用 SSE-KMS 加密

對跨帳戶操作使用加密時，請注意下列事項：

- 如果在請求時或透過儲存貯體的預設加密組態未提供 AWS KMS key Amazon 資源名稱 (ARN) 或別名，則會使用 AWS 受管金鑰 (aws/s3)。
- 如果您使用與 KMS 金鑰相同 AWS 帳戶 的 AWS Identity and Access Management (IAM) 主體上傳或存取 S3 物件，則可以使用 AWS 受管金鑰 (aws/s3)。
- 如果您想要授予 S3 物件跨帳戶存取權，請使用客戶受管金鑰。您可以設定客戶受管金鑰的政策，以允許從另一個帳戶存取的權限。
- 如果您要指定客戶受管 KMS 金鑰，建議您使用完整的 KMS 金鑰 ARN。如果您改用 KMS 金鑰別名，請 AWS KMS 解析要求者帳戶內的金鑰。此行為可能會導致資料使用屬於申請者 (而不是儲存貯體擁有者) 的 KMS 金鑰來加密。
- 您必須指定已授予您 (要求者) 已獲授予 Encrypt 許可的金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的允許金鑰使用者在密碼編譯操作中使用 KMS 金鑰。

如需何時使用客戶受管金鑰和受 AWS 管 KMS 金鑰的詳細資訊，請參閱我應該使用 AWS 受管金鑰 還是客戶受管金鑰來加密 Amazon S3 中的物件？

搭配使用預設加密與複寫

在您啟用域複寫目的地儲存貯體的預設加密之後，適用下列加密行為：

- 如果未加密來源儲存貯體中的物件，則會使用目的地儲存貯體的預設加密設定來加密目的地儲存貯體中的複本物件。因此，來源物件的實體標籤 (ETag) 與複本物件的 ETag 不同。如果您有使用 ETag 的應用程式，則必須更新這些應用程式以解決此差異。
- 如果來源儲存貯體中的物件使用伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3)、使用 () 金鑰 (SSE-KMS AWS KMS) 進行伺服器端加密，或使用金鑰的雙層伺服器端加密 (DSSE-KMS) 加密，目的地儲存貯體中的複本物件會使用與 AWS KMS 來源物件相同的加密類型。 AWS Key Management Service 不會使用目的地儲存貯體的預設加密設定。

如需搭配 SSE-KMS 使用預設加密的詳細資訊，請參閱複寫加密的物件。

搭配使用 Amazon S3 儲存貯體金鑰和預設加密

當您將儲存貯體設定為在新物件上使用 SSE-KMS 做為預設加密行為時，您還可以設定 S3 儲存貯體金鑰。S3 儲存貯體金鑰可減少 Amazon S3 的交易數量， AWS KMS 以降低 SSE-KMS 的成本。

當您將儲存貯體設定為在新物件上使用 S3 儲存貯體金鑰進行 SSE-KMS 時， AWS KMS 會產生值區層級金鑰，用來為儲存貯體中的物件建立唯一的資料金鑰。此 S3 儲存貯體金鑰在 Amazon S3 中使用一段時間限制，因此可減少 Amazon S3 提出要求以完成 AWS KMS 加密操作的需求。

如需使用 S3 儲存貯體金鑰的詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰。

設定預設加密

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

根據預設，Amazon S3 儲存貯體皆已啟動儲存貯體加密，且使用伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 自動加密新物件。此加密免費適用於 Amazon S3 儲存貯體中的所有新物件。

如果您需要對加密金鑰進行更多控制，例如管理金鑰輪替和存取原則授與，您可以選擇使用伺服器端加密 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS)，或使用金鑰 (DSSE-KMS) 來使

用 AWS KMS 雙層伺服器端加密。如需 SSE-KMS 的詳細資訊，請參閱「使用 AWS KMS (SSE-KMS) 指定伺服器端加密」。如需 DSSE-KMS 的詳細資訊，請參閱 the section called “雙層伺服器端加密 (DSSE-KMS)”。

若您想要使用其他帳戶的 KMS 金鑰，您必須具有該金鑰的使用權限。如需詳細了解 KMS 金鑰跨帳戶權限，請參閱《AWS Key Management Service 開發人員指南》中的建立其他帳戶可使用的 KMS 金鑰。

將預設儲存貯體加密設定為 SSE-KMS 時，您也可以設定 S3 儲存貯體金鑰以降低 AWS KMS 請求成本。如需詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本。

> **Note**
>
> 如果您使用PutBucketEncryption將預設值區加密設定為 SSE-KMS，則應確認 KMS 金鑰識別碼正確無誤。Amazon S3 不會驗證 PutBucketEncryption 請求中提供的 KMS 金鑰識別碼。

使用 S3 儲存貯體的預設加密不需要額外收費。請求設定預設加密行為會產生標準 Amazon S3 請求費用。如需定價的資訊，請參閱 Amazon S3 定價。對於 SSE-KMS 和 DSSE-KMS，會 AWS KMS 收取費用，並按價格列出。AWS KMS

不支援使用客戶所提供金鑰 (SSE-C) 的伺服器端加密做為預設加密。

您可以使用 Amazon S3 主控台、 AWS 開發套件、Amazon S3 REST API 和 AWS Command Line Interface (AWS CLI)，為 S3 儲存貯體設定 Amazon S3 預設加密。

啟用預設加密之前，應注意的變更

在您啟用儲存貯體的預設加密之後，適用下列加密行為：

- 在啟用預設加密之前，不會變更儲存貯體中現有物件的加密。
- 在啟用預設加密之後上傳物件時：
  - 如果您的 PUT 請求標題未包含加密資訊，則 Amazon S3 會使用儲存貯體的預設加密設定來加密物件。
  - 如果您的 PUT 請求標題包含加密資訊，則 Amazon S3 會先使用 PUT 請求中的加密資訊來加密物件，再將物件存放至 Amazon S3。
- 如果您針對預設加密組態使用 SSE-KMS 或 DSSE-KMS 選項，則受到 AWS KMS的每秒請求數目 (RPS) 配額限制。如需 AWS KMS 配額以及如何請求提高配額的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的配額。

> **Note**
>
> 在啟用預設加密之前上傳的物件不會受到加密。如需詳細了解加密現有物件，請參閱 the section called “設定預設儲存貯體加密”。

使用 S3 主控台

設定 Amazon S3 儲存貯體的預設加密

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇您所需的儲存貯體名稱。
4. 選擇屬性索引標籤。
5. 在 Default encryption (預設加密) 底下，選擇 Edit (編輯)。
6. 若要設定加密，請在加密類型下，選擇下列其中一項：
   - 使用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密
   - 使用 AWS Key Management Service 金鑰 (SSE-KMS) 進行伺服器端加密
   - 使用 AWS Key Management Service 金鑰 (DSSE-KMS) 進行雙層伺服器端加密

   > **Important**
   >
   > 如果您針對預設加密組態使用 SSE-KMS 或 DSSE-KMS 選項，則受到 AWS KMS的每秒請求數目 (RPS) 配額限制。如需有關 AWS KMS 配額以及如何要求提高配額的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的配額。

   除非您為儲存貯體指定其他類型的預設加密，否則根據預設，儲存貯體和新物件皆會使用 SSE-S3 加密。如需預設加密的詳細資訊，請參閱對 Amazon S3 儲存貯體設定預設伺服器端加密行為。

   如需有關使用 Amazon S3 伺服器端加密來加密資料的詳細資訊，請參閱「使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密」。
7. 如果您選擇使用金 AWS Key Management Service 鑰進行伺服器端加密 (SSE-KMS) 或使用金 AWS Key Management Service 鑰的雙層伺服器端加密 (DSSE-KMS)，請執行下列動作：

a. 在 AWS KMS 金鑰下，使用下列其中一種方式指定 KMS 金鑰：

- 若要從可用 KMS 金鑰清單中選擇，請選擇從您的金鑰中選擇 AWS KMS keys，然後從可用金鑰清單中選擇您的 KMS 金鑰。

  AWS 受管金鑰 (aws/s3) 和您的客戶管理金鑰都會出現在此清單中。如需有關客戶受管金鑰的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的客戶 AWS 金鑰和金鑰。
- 若要輸入 KMS 金鑰 ARN，請選擇輸入 AWS KMS key ARN，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。
- 若要在 AWS KMS 主控台中建立新的客戶管理金鑰，請選擇 [建立 KMS 金鑰]。

  如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。

> **Important**
>
> 您只能使用與儲存貯體相 AWS 區域 同的 KMS 金鑰。如果選擇的是 Choose from your KMS keys (從您的 KMS 金鑰選擇)，則 S3 主控台只會列出每個區域的其中 100 個 KMS 金鑰。如果您在相同區域中有超過 100 個 KMS 金鑰，您只能看到 S3 主控台中的前 100 個 KMS 金鑰。若要使用主控台中未列出的 KMS 金鑰，請選擇輸入 AWS KMS key ARN，然後輸入 KMS 金鑰 ARN。
> 當您在 Amazon S3 中使 AWS KMS key 用伺服器端加密時，您必須選擇對稱加密 KMS 金鑰。Amazon WorkMail 只支援對稱加密 KMS 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的對稱加密 KMS 金鑰。

如需有關搭配 Amazon S3 使用 SSE-KMS 的詳細資訊，請參閱 使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)。如需 DSSE-KMS 的使用詳細資訊，請參閱 the section called “雙層伺服器端加密 (DSSE-KMS)”。

b. 當您將儲存貯體設定為使用 SSE-KMS 的預設加密時，您還可以啟用 S3 儲存貯體金鑰。S3 儲存貯體金鑰可將 Amazon S3 的請求流量減少到 AWS KMS。如需詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本。

   若要使用 S3 儲存貯體金鑰，在 Bucket Key (儲存貯體金鑰) 下選擇 Enable (啟用)。

> **Note**
>
> DSSE-KMS 不支援 S3 儲存貯體金鑰。

8. 選擇儲存變更。

使用 AWS CLI

這些範例說明如何使用 SSE-S3 或具有 S3 儲存貯體金鑰的 SSE-KMS 來設定預設加密。

如需預設加密的詳細資訊，請參閱對 Amazon S3 儲存貯體設定預設伺服器端加密行為。如需使用配置預設加密 AWS CLI 的詳細資訊，請參閱put-bucket-encryption。

Example – 使用 SSE-S3 預設加密

此範例會使用 Amazon S3 受管金鑰設定預設儲存貯體加密。

```
aws s3api put-bucket-encryption --bucket example-s3-bucket --server-side-encryption-
configuration '{
    "Rules": [
        {
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }
    ]
}'
```

Example – 使用 S3 儲存貯體金鑰以 SSE-KMS 預設加密

此範例會使用 S3 儲存貯體金鑰，以 SSE-KMS 設定預設儲存貯體加密。

```
aws s3api put-bucket-encryption --bucket example-s3-bucket --server-side-encryption-
configuration '{
    "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
```

```
                "BucketKeyEnabled": true
            }
        ]
    }'
```

使用 REST API

使用 REST API PutBucketEncryption 操作來啟用預設加密，以及設定要使用的伺服器端加密類型 (SSE-S3、SSE-KMS 或 DSSE-KMS)。

如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 PutBucketEncryption。

使用 AWS CloudTrail 和 Amazon 監控默認加密 EventBridge

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

您可以使用 AWS CloudTrail 事件追蹤 Amazon S3 儲存貯體的預設加密組態請求。 CloudTrail 記錄檔中會使用下列 API 事件名稱：

- PutBucketEncryption
- GetBucketEncryption
- DeleteBucketEncryption

您也可以建立 EventBridge 規則以符合這些 API 呼叫的 CloudTrail 事件。如需事件的詳細資 CloudTrail 訊，請參閱使用主控台啟用儲存貯體中物件的記錄。如需有關 EventBridge 事件的詳細資訊，請參閱來自的事件 AWS 服務。

您可以將 CloudTrail 日誌用於物件層級 Amazon S3 動作，以追蹤PUT和向 Amazon S3 發出POST請求。當傳入 PUT 要求沒有加密標頭時，您可以使用這些動作來驗證是否使用預設加密來加密物件。

當 Amazon S3 使用預設加密設定來加密物件時，日誌會包含以下其中一個名稱-值對的欄位："SSEApplied":"Default_SSE_S3"、"SSEApplied":"Default_SSE_KMS" 或 "SSEApplied":"Default_DSSE_KMS"。

當 Amazon S3 使用 PUT 加密標頭來加密物件時，日誌會包含以下名稱-值對的欄位之一："SSEApplied":"SSE_S3"、"SSEApplied":"SSE_KMS"、"SSEApplied":"DSSE_KMS" 或 "SSEApplied":"SSE_C"。

針對分段上傳，此資訊會包含在 InitiateMultipartUpload API 操作要求中。若要取得有關使用 CloudTrail 和的更多資訊 CloudWatch，請參閱監控 Amazon S3。

使用適用於 Amazon S3 的掛載點

適用於 Amazon S3 的掛載點是高輸送量的開放原始碼檔案用戶端，可將 Amazon S3 儲存貯體掛載為本機檔案系統。使用掛載點可讓您的應用程式透過像是開啟和讀取等檔案系統操作，存取儲存在 Amazon S3 中的物件。掛載點會自動將這些操作轉換成 S3 物件 API 呼叫，讓您的應用程式透過檔案介面存取 Amazon S3 的彈性儲存和輸送量。

適用於 Amazon S3 的掛載點已全面推出，可供大量讀取應用程式的生產環境使用，像是：資料湖、機器學習訓練、影像轉譯、自動駕駛車輛模擬、擷取、轉換和載入 (ETL) 等。

掛載點可支援基本的檔案系統操作，並且可讀取最大 5 TB 的檔案。它可以列出和讀取現有檔案，也可以建立新檔案。它無法修改現有檔案或刪除目錄，也不支援符號連結或檔案鎖定。掛載點非常適合不需要共用檔案系統和 POSIX 樣式許可，但需要 Amazon S3 的彈性輸送量來讀取和寫入大型 S3 資料集的應用程式。如需詳細資訊，請參閱 GitHub 上的掛載點檔案系統行為。對於需要完整 POSIX 支援的工作負載，我們建議採用 Amazon FSx for Lustre 及其連結 S3 儲存貯體的支援。

適用於 Amazon S3 的掛載點僅適用於 Linux 作業系統。您可以使用掛載點存取所有儲存類別中的 S3 物件，但 S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive、S3 Intelligent-Tiering Archive Access Tier 和 S3 Intelligent-Tiering Deep Archive Access Tier 除外。

主題

安裝掛載點

您可以使用命令列下載並安裝適用於 Amazon S3 的掛載點的預先建置套件。下載和安裝掛載點的說明會根據您使用的 Linux 作業系統而有所不同。

主題

RPM 型發行版本 (Amazon Linux、Fedora、CentOS、RHEL)

1. 為您的架構複製下列下載 URL。

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.rpm
   ```

2. 下載適用於 Amazon S3 的掛載點套件。將 *download-link* 取代為前一步驟中適當的下載 URL。

   ```
   wget download-link
   ```

3. (選用) 驗證所下載檔案的真實性和完整性。首先，為您的架構複製適當的簽章 URL。

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm.asc
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.rpm.asc
   ```

   接著請參閱驗證適用於 Amazon S3 的掛載點套件的簽章。

4. 使用以下命令安裝套件：

   ```
   sudo yum install ./mount-s3.rpm
   ```

5. 輸入下列命令，以驗證掛載點安裝成功：

```
mount-s3 --version
```

您應該會看到類似下列的輸出：

```
mount-s3 1.3.1
```

DEB 型發行版本 (Debian、Ubuntu)

1. 為您的架構複製下載 URL。

x86_64:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.deb
```

ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.deb
```

2. 下載適用於 Amazon S3 的掛載點套件。將 *download-link* 取代為前一步驟中適當的下載 URL。

```
wget download-link
```

3. (選用) 驗證所下載檔案的真實性和完整性。首先，為您的架構複製簽章 URL。

x86_64:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.deb.asc
```

ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.deb.asc
```

接著請參閱驗證適用於 Amazon S3 的掛載點套件的簽章。

4. 使用以下命令安裝套件：

```
sudo apt-get install ./mount-s3.deb
```

5. 執行下列命令，以驗證適用於 Amazon S3 的掛載點安裝成功：

```
mount-s3 --version
```

您應該會看到類似下列的輸出：

```
mount-s3 1.3.1
```

其他 Linux 發行版本

1. 請參閱您的作業系統文件以安裝 FUSE 和 `libfuse2` 套件，這是必要項。
2. 為您的架構複製下載 URL。

x86_64:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.tar.gz
```

ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.tar.gz
```

3. 下載適用於 Amazon S3 的掛載點套件。將 *download-link* 取代為前一步驟中適當的下載 URL。

```
wget download-link
```

4. (選用) 驗證所下載檔案的真實性和完整性。首先，為您的架構複製簽章 URL。

x86_64:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.tar.gz.asc
```

ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.tar.gz.asc
```

接著請參閱驗證適用於 Amazon S3 的掛載點套件的簽章。

5. 使用以下命令安裝套件：

```
sudo mkdir -p /opt/aws/mountpoint-s3 && sudo tar -C /opt/aws/mountpoint-s3 -xzf ./
mount-s3.tar.gz
```

6. 將 mount-s3 二進位檔案新增至 PATH 環境變數。在您的 $HOME/.profile 檔案中，附加下面這行：

```
export PATH=$PATH:/opt/aws/mountpoint-s3/bin
```

儲存 .profile 檔案，並執行下列命令：

```
source $HOME/.profile
```

7. 執行下列命令，以驗證適用於 Amazon S3 的掛載點安裝成功：

```
mount-s3 --version
```

您應該會看到類似下列的輸出：

```
mount-s3 1.3.1
```

驗證適用於 Amazon S3 的掛載點套件的簽章

1. 安裝 GnuPG (gpg 命令)。您必須驗證所下載適用於 Amazon S3 的掛載點套件的真實性和完整性。GnuPG 會預設安裝在 Amazon Linux Amazon Machine Image (AMI) 上。安裝 GnuPG 之後，繼續進行步驟 2。
2. 執行下列命令以下載掛載點公有金鑰：

```
wget https://s3.amazonaws.com/mountpoint-s3-release/public_keys/KEYS
```

3. 執行下列命令，將掛載點公有金鑰匯入至您的鑰匙圈：

```
gpg --import KEYS
```

4. 執行下列命令以驗證掛載點公有金鑰的指紋：

```
gpg --fingerprint mountpoint-s3@amazon.com
```

確認顯示的指紋字串符合下列內容：

```
673F E406 1506 BB46 9A0E  F857 BE39 7A52 B086 DA5A
```

如果指紋字串不相符，請不要完成安裝掛載點，並聯絡 AWS Support。

5. 下載套件簽章檔案。將 *signature-link* 取代為先前章節中適當的簽章連結。

```
wget signature-link
```

6. 執行下列命令以驗證所下載套件的簽章。將 *signature-filename* 取代為前一步驟中的檔案名稱。

```
gpg --verify signature-filename
```

例如，在 RPM 型發行版本上 (包括 Amazon Linux) 輸入下列命令：

```
gpg --verify mount-s3.rpm.asc
```

7. 輸出應該會包含 `Good signature` 這個片語。如果輸出包括 `BAD signature` 這個片語，請重新下載掛載點套件檔案並重複這些步驟。如果問題仍然存在，請不要完成安裝掛載點，並聯絡 AWS Support。

   輸出可能包含有關信任簽章的警告。這並不表示有問題。這只表示，您尚未獨立驗證掛載點公有金鑰。

配置和使用掛載點

若要使用 Amazon S3 的掛載點，您的主機需要有效的 AWS 登入資料，才能存取您要掛載的儲存貯體或儲存貯體。如需不同的身分驗證方式，請參閱 GitHub 上的 AWS 憑證。

例如，您可以為此目的建立新的 AWS Identity and Access Management (IAM) 使用者和角色。請確定此角色可存取您要掛載的一或多個儲存貯體。您可以使用執行個體設定檔將 IAM 角色傳遞至您的 Amazon EC2 執行個體。

使用適用於 Amazon S3 的掛載點

使用適用於 Amazon S3 的掛載點執行下列操作：

1. 使用 `mount-s3` 命令掛載儲存貯體。

   在下列範例中，將 *DOC-EXAMPLE-BUCKET* 取代為您的 S3 儲存貯體名稱，並將 *~/mnt* 取代為您主機上要掛載 S3 儲存貯體的目錄。

   ```
   mkdir ~/mnt
   mount-s3 DOC-EXAMPLE-BUCKET ~/mnt
   ```

   由於掛載點用戶端預設會在背景執行，因此您現在可透過 *~/mnt* 目錄存取 S3 儲存貯體中的物件。

2. 透過掛載點存取儲存貯體中的物件。

   在本機上掛載儲存貯體後，您可以使用常見的 Linux 命令 (例如 `cat` 或 `ls`) 來使用您的 S3 物件。適用於 Amazon S3 的掛載點會將您 S3 儲存貯體中的金鑰解譯為檔案系統路徑，方法是使用正斜線 (/) 字元分割金鑰。例如，如果您的儲存貯體中有物件金鑰 `Data/2023-01-01.csv`，那麼您的掛載點檔案系統中將會有名為 `Data` 的目錄，且當中會有名為 `2023-01-01.csv` 的檔案。

   適用於 Amazon S3 的掛載點會刻意不對檔案系統實作完整的 POSIX 標準規格。掛載點會針對需要透過檔案系統介面對儲存在 Amazon S3 中的資料進行高輸送量讀取和寫入存取權的工作負載最佳化，否則不會依賴檔案系統功能。如需詳細資訊，請參閱 GitHub 上的適用於 Amazon S3 的掛載點檔案系統行為。需要更豐富檔案系統語意的客戶應考慮使用其他 AWS 檔案服務，例如 Amazon Elastic File System (Amazon EFS) 或 Amazon FS x。

3. 使用 `umount` 命令卸載儲存貯體。此命令會卸載 S3 儲存貯體並結束掛載點。

   若要使用下列範例命令，請將 *~/mnt* 取代為您的 S3 儲存貯體掛載所在主機上的目錄。

   ```
   umount ~/mnt
   ```

   > **Note**
   >
   > 若要取得此命令的選項清單，請執行 `umount --help`。

如需其他掛載點組態的詳細資訊，請參閱 GitHub 上的 S3 儲存貯體組態及檔案系統組態。

在掛載點中設定快取

當您使用適用於 Amazon S3 的掛載點時，可以將其設定為從 Amazon EC2 執行個體儲存或連接的 Amazon EBS 磁碟區上的 S3 儲存貯體快取最近存取的資料。快取此資料有助於提高效能並降低重複存取資料的成本。在掛載點中快取非常適合在多次讀取過程中，重複讀取未變更的相同資料的使用案例。例如，您可以使用快取搭配需要多次讀取訓練資料集的機器學習訓練任務，以提高模型準確度。

掛載 S3 儲存貯體時，可以選擇透過旗標啟用快取。您可以設定資料快取的位置和大小，以及中繼資料保留在快取中的時間長度。當您掛載儲存貯體且快取為啟用狀態時，掛載點會在設定的快取位置建立一個空的子目錄 (如果該子目錄尚未存在)。當您第一次掛載儲存貯體，以及當您卸載時，掛載點會刪除快取位置的內容。如需有關在掛載點中設定和使用快取的詳細資訊，請參閱上的 Amazon S3 快取組態的掛載點。 GitHub

掛載 S3 儲存貯體時，可以使用 `--cache` *`CACHE_PATH`* 旗標啟用快取。在下列範例中，請將 *`CACHE_PATH`* 取代為要將資料快取至其中的目錄檔案路徑。將 *`DOC-EXAMPLE-BUCKET`* 取代為您的 S3 儲存貯體名稱，並將 *`~/mnt`* 取代為您主機上要掛載 S3 儲存貯體的目錄。

```
mkdir ~/mnt
mount-s3 --cache CACHE_PATH DOC-EXAMPLE-BUCKET ~/mnt
```

> **Important**
>
> 如果您啟用快取，掛載點會將來自您 S3 儲存貯體的未加密物件內容保留在掛載時設定的快取位置。為了保護您的資料，建議您限制資料快取位置的存取權。

疑難排解掛載點

Amazon S3 的掛載點由. AWS Support如果您需要協助，請聯絡 AWS Support 中心。

您也可以在 GitHub 上檢閱並提交掛載點問題。

如果您在此專案中發現潛在的安全問題，請透過我們的漏洞報告頁面通知 AWS Security。請勿建立公有 GitHub 問題。

如果您的應用程式在掛載點上的行為異常，您可以查看日誌資訊以診斷問題。

日誌

根據預設，掛載點會將高嚴重性日誌資訊發出至 syslog。

若要在大多數現代 Linux 發行版本 (包括 Amazon Linux) 上檢視日誌，請執行下列 `journald` 命令：

```
journalctl -e SYSLOG_IDENTIFIER=mount-s3
```

在其他 Linux 系統上，`syslog` 項目可能會寫入檔案中，如 `/var/log/syslog`。

您可以使用這些日誌對應用程式進行疑難排解。例如，如果您的應用程式嘗試覆寫現有的檔案，則操作會失敗，且您會在日誌中看到類似下面這行內容：

```
[WARN] open{req=12 ino=2}: mountpoint_s3::fuse: open failed: inode error: inode 2 (full
 key "README.md") is not writable
```

如需詳細資訊，請參閱 GitHub 上的適用於 Amazon S3 的掛載點日誌。

使用 Amazon S3 Transfer Acceleration 設定快速安全的檔案傳輸

Amazon S3 Transfer Acceleration 是一項儲存貯體層級的功能，可讓用戶端與 S3 儲存貯體間的長距離檔案傳輸變得迅速、簡單又安全。Transfer Acceleration 旨在最佳化從世界各地傳輸到 S3 儲存貯體的速度。傳輸加速充分利用 Amazon 中全球分佈的節點 CloudFront。當資料到達節點時，資料會經由最佳化的網路路徑而路由至 Amazon S3。

使用 Transfer Acceleration 時，可能需要收取額外的資料傳輸費用。如需定價的詳細資訊，請參閱 Amazon S3 定價。

為什麼要使用 Transfer Acceleration？

您可能會基於各種原因而想要在儲存貯體上使用 Transfer Acceleration：

- 您的客戶從世界各地上傳資料至集中型儲存貯體 。
- 您會定期跨洲傳送數 GB 到 TB 的資料。
- 上傳到 Amazon S3 時，您無法使用網際網路上可用的所有頻寬。

如需何時使用 Transfer Acceleration 的詳細資訊，請參閱 Amazon S3 常見問答集。

使用 Transfer Acceleration 的要求

在 S3 儲存貯體上使用 Transfer Acceleration 需要滿足下列要求：

- 只有虛擬裝載樣式要求才支援 Transfer Acceleration。如需虛擬託管樣式請求的詳細資訊，請參閱使用 REST API 提出要求。
- 用於 Transfer Acceleration 的儲存貯體名稱必須與 DNS 相容，而且不得包含句點 (".")。
- 必須在儲存貯體上啟用 Transfer Acceleration。如需詳細資訊，請參閱 啟用和使用 S3 Transfer Acceleration。

  在儲存貯體上啟用 Transfer Acceleration 之後，最多可能需要 20 分鐘，儲存貯體的資料傳送速度才會增加。

  > **Note**
  >
  > Transfer Acceleration 目前支援位於下列區域的儲存貯體：
  >
  > - 亞太區域 (東京) (ap-northeast-1)
  > - 亞太區域 (首爾) (ap-northeast-2)
  > - 亞太區域 (孟買) (ap-south-1)
  > - 亞太區域 (新加坡) (ap-southeast-1)
  > - 亞太區域 (雪梨) (ap-southeast-2)
  > - 加拿大 (中部) (ca-central-1)
  > - 歐洲 (法蘭克福) (eu-central-1)
  > - 歐洲 (愛爾蘭) (eu-west-1)
  > - 歐洲 (倫敦) (eu-west-2)
  > - 歐洲 (巴黎) (eu-west-3)
  > - 南美洲 (聖保羅) (sa-east-1)
  > - 美國東部 (維吉尼亞北部) (us-east-1)
  > - 美國東部 (俄亥俄) (us-east-2)
  > - 美國西部 (加利佛尼亞北部) (us-west-1)
  > - 美國西部 (奧勒岡) (us-west-2)

- 若要存取啟用 Transfer Acceleration 的儲存貯體，您必須使用端點 *bucketname*.s3-accelerate.amazonaws.com，或使用雙堆疊端點 *bucketname*.s3-accelerate.dualstack.amazonaws.com，透過 IPv6 連接至已啟用的儲存貯體。您可以繼續使用一般端點進行標準資料傳輸。
- 您必須是儲存貯體擁有者，才能設定傳輸加速狀態。儲存貯體擁有者可以將許可指派給其他使用者，讓他們可以在儲存貯體上設定加速狀態。s3:PutAccelerateConfiguration 許可允許使用者在

儲存貯體上啟用或停用 Transfer Acceleration。該`s3:GetAccelerateConfiguration`權限允許用戶返回值區的傳輸加速狀態，該狀態為`Enabled`或 `Suspended`.

下列章節說明如何開始使用 Amazon S3 Transfer Acceleration 傳輸資料。

主題

- Amazon S3 Transfer Acceleration 入門
- 啟用和使用 S3 Transfer Acceleration
- 使用 Amazon S3 Transfer Acceleration 速度比較工具

Amazon S3 Transfer Acceleration 入門

您可以使用 Amazon S3 Transfer Acceleration 讓用戶端與 S3 儲存貯體間的長距離檔案傳輸變得迅速、簡單又安全。傳輸加速使用 Amazon 中遍佈全球的節點 CloudFront。當資料到達節點時，資料會經由最佳化的網路路徑而路由至 Amazon S3。

若要開始使用 Amazon S3 Transfer Acceleration，請執行下列步驟：

1. 在儲存貯體上啟用 Transfer Acceleration

   您可以透過下列任何方法，在儲存貯體上啟用 Transfer Acceleration：

   - 使用 Amazon S3 主控台。
   - 使用 REST API PUT 儲存貯體加速操作。
   - 使用和軟體 AWS CLI 開 AWS 發套件。如需詳細資訊，請參閱 使用開發 AWS 套件使用 Amazon S3 進行開發。

   如需詳細資訊，請參閱 啟用和使用 S3 Transfer Acceleration。

   > **Note**
   >
   > 若要讓儲存貯體使用 Transfer Acceleration，儲存貯體名稱必須符合 DNS 命名需求，而且不得包含句點 (".")。

2. 與啟用加速功能的儲存貯體進行資料傳輸

   使用下列其中一個 s3-accelerate 端點網域名稱：

- 若要存取啟用加速功能的儲存貯體，請使用 *bucketname*.s3-accelerate.amazonaws.com。
- 若要透過 IPv6 存取啟用加速功能的儲存貯體，請使用 *bucketname*.s3-accelerate.dualstack.amazonaws.com。

  Amazon S3 雙堆疊端點支援透過 IPv6 與 IPv4 的 S3 儲存貯體要求。Transfer Acceleration 雙堆疊端點只會使用虛擬託管樣式類型的端點名稱。如需詳細資訊，請參閱 透過 IPv6 提出請求的入門 及 使用 Amazon S3 雙堆疊端點。

> **Note**
>
> 您的資料傳輸應用程式必須使用下列兩種類型的端點之一來存取儲存貯體，以加快資料傳輸速度：.s3-accelerate.amazonaws.com，或是用於雙堆疊端點的 .s3-accelerate.dualstack.amazonaws.com。如果您想要使用標準資料傳輸，可以繼續使用一般端點。

在您啟用 Transfer Acceleration 之後，可以將 Amazon S3 PUT 物件與 GET 物件要求指向 s3-accelerate 端點網域名稱。例如，假設您目前有使用 PUT 物件的 REST API 應用程式，而此物件在 PUT 請求中使用主機名稱 mybucket.s3.us-east-1.amazonaws.com。為了加速 PUT，您可以將要求中的主機名稱更改為 mybucket.s3-accelerate.amazonaws.com。若要回復為使用標準上傳速度，只需要將名稱變更回 mybucket.s3.us-east-1.amazonaws.com。

啟用 Transfer Acceleration 之後，最多需要 20 分鐘即讓您可以實現效能利益。不過，只要您啟用 Transfer Acceleration，就可以使用加速端點。

您可以在 AWS CLI、 AWS 開發套件和其他工具中使用加速端點，將資料傳入和傳出 Amazon S3。如果您使用的是 AWS SDK，某些支援的語言會使用加速端點用戶端組態旗標，因此您不需要明確設定傳輸加速到*bucketname*.s3-accelerate.amazonaws.com的端點。如需如何使用加速端點用戶端組態旗標的範例，請參閱「啟用和使用 S3 Transfer Acceleration」。

您可以透過 Transfer Acceleration 端點使用所有 Amazon S3 操作，但下列項目除外：

- GET 服務 (列出所有儲存貯體)
- PUT 儲存貯體 (建立儲存貯體)
- DELETE 儲存貯體

此外，Amazon S3 Transfer Acceleration 不支援使用 PUT 物件 - 複製進行跨區域複製。

啟用和使用 S3 Transfer Acceleration

您可以使用 Amazon S3 Transfer Acceleration，在用戶端與 S3 儲存貯體之間長距離快速安全地傳輸檔案。您可以使用 S3 主控台、 AWS Command Line Interface (AWS CLI)、API 或 AWS SDK 啟用傳輸加速。

本節示範如何在儲存貯體上啟用 Amazon S3 Transfer Acceleration，以及使用已啟用儲存貯體的加速端點。

如需 Transfer Acceleratio 要求的更多資訊，請參閱「使用 Amazon S3 Transfer Acceleration 設定快速安全的檔案傳輸」。

使用 S3 主控台

> **Note**
>
> 如果您想要比較加速和非加速的上傳速度，請開啟 Amazon S3 Transfer Acceleration 速度比較工具。
> 速度比較工具使用分段上傳功能，將檔案從瀏覽器傳輸到不使 AWS 區域 用 Amazon S3 傳輸加速的各種檔案。您可以依區域比較直接上傳和傳輸加速上傳的上傳速度。

啟用 S3 儲存貯體的 Transfer Acceleration

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Bucket (儲存貯體) 清單中，選擇您要啟用 Transfer Acceleration 的儲存貯體名稱。
3. 選擇 Properties (屬性)。
4. 在轉換加速下，選擇編輯。
5. 依序選擇 Enable (啟用) 和 Save changes (儲存變更)。

實現資料傳輸加速

1. 在 Amazon S3 為儲存貯體啟用 Transfer Acceleration 後，請檢視儲存貯體的 Properties (屬性) 索引標籤。

2. 在傳輸加速下，加速端點會顯示儲存貯體的傳輸加速端點。使用此端點以實現進出儲存貯體的資料傳輸加速。

   若暫停傳輸加速，則加速端點將不會再運作。

使用 AWS CLI

以下是用於傳輸加速的 AWS CLI 指令範例。如需有關設定的指示 AWS CLI，請參閱使用 AWS CLI 來透過 Amazon S3 進行開發。

在儲存貯體上啟用 Transfer Acceleration

使用指 AWS CLI put-bucket-accelerate-configuration令可在值區上啟用或暫停傳輸加速。

下列範例設定 `Status=Enabled`，在儲存貯體上啟用 Transfer Acceleration。您可以使用 `Status=Suspended` 暫停 Transfer Acceleration。

Example

```
$ aws s3api put-bucket-accelerate-configuration --bucket bucketname --accelerate-
configuration Status=Enabled
```

使用 Transfer Acceleration

您可以將 s3 和 s3api AWS CLI 命令發出的所有 Amazon S3 請求導向到加速端點：。`s3-accelerate.amazonaws.com`若要這麼做，請`true`在 AWS Config 檔案的設定檔中`use_accelerate_endpoint`將組態值設定為。必須在儲存貯體上啟用 Transfer Acceleration，才能使用加速端點。

所有請求都是使用虛擬樣式的儲存貯體定址所傳送：`my-bucket.s3-accelerate.amazonaws.com`。不會將任何 `ListBuckets`、`CreateBucket` 和 `DeleteBucket` 請求傳送至加速端點，因為該端點不支援這些操作。

如需有關 `use_accelerate_endpoint` 的詳細資訊，請參閱 AWS CLI 命令參考中的 AWS CLI S3 組態。

下列範例會將預設描述檔中的 `use_accelerate_endpoint` 設為 `true`。

Example

```
$ aws configure set default.s3.use_accelerate_endpoint true
```

如果您想要將加速端點用於某些 AWS CLI 命令，但不想使用其他命令，則可以使用下列兩種方法之一：

- 透過將 `--endpoint-url` 參數設定為 `https://s3-accelerate.amazonaws.com`，使用任何 s3 或 s3api 命令加速端點。
- 在您的 Config 文件中設置單獨的 AWS 配置文件。例如，您可以建立一個設定檔，將 `use_accelerate_endpoint` 設為 `true`，再建立另一個設定檔不設定 `use_accelerate_endpoint`。當您執行命令時，根據是否要使用加速端點來指定要使用的描述檔。

將物件上傳至已啟用 Transfer Acceleration 的儲存貯體

下列範例使用已設定成使用加速端點的預設描述檔，以將檔案上傳至已啟用 Transfer Acceleration 的儲存貯體。

Example

```
$ aws s3 cp file.txt s3://bucketname/keyname --region region
```

下列範例使用 `--endpoint-url` 參數指定加速端點，以將檔案上傳至已啟用 Transfer Acceleration 的儲存貯體。

Example

```
$ aws configure set s3.addressing_style virtual
$ aws s3 cp file.txt s3://bucketname/keyname --region region --endpoint-url https://s3-accelerate.amazonaws.com
```

使用 AWS 軟體開發套件

以下是使用「傳輸加速」使用 AWS 開發套件將物件上傳到 Amazon S3 的範例。某些 AWS SDK 支援的語言 (例如 Java 和 .NET) 會使用加速端點用戶端設定旗標，因此您不需要將傳輸加速的端點明確設定為儲存格名稱 .s3 *-## .amazonaws.com#*

Java

Example

下列範例示範如何使用加速端點，上傳物件至 Amazon S3。此範例執行下列操作：

- 使用加速端點，建立已經過設定的 AmazonS3Client。所有客戶存取的儲存貯體，一定要啟用 Transfer Acceleration。
- 在特定儲存貯體上，啟用 Transfer Acceleration。此步驟僅在您所指定的儲存貯體並沒有啟用 Transfer Acceleration 時為必需。
- 驗證特定儲存貯體上，是否啟用 Transfer Acceleration。
- 使用儲存貯體的加速端點，上傳新物件至指定的儲存貯體中。

如需有關使用 Transfer Acceleration 的詳細資訊，請參閱 Amazon S3 Transfer Acceleration 入門。如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketAccelerateConfiguration;
import com.amazonaws.services.s3.model.BucketAccelerateStatus;
import com.amazonaws.services.s3.model.GetBucketAccelerateConfigurationRequest;
import com.amazonaws.services.s3.model.SetBucketAccelerateConfigurationRequest;

public class TransferAcceleration {
    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";

        try {
            // Create an Amazon S3 client that is configured to use the accelerate
 endpoint.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .enableAccelerateMode()
                    .build();

            // Enable Transfer Acceleration for the specified bucket.
            s3Client.setBucketAccelerateConfiguration(
                    new SetBucketAccelerateConfigurationRequest(bucketName,
                            new BucketAccelerateConfiguration(
```

```
                                    BucketAccelerateStatus.Enabled()));

            // Verify that transfer acceleration is enabled for the bucket.
            String accelerateStatus = s3Client.getBucketAccelerateConfiguration(
                    new GetBucketAccelerateConfigurationRequest(bucketName))
                    .getStatus();
            System.out.println("Bucket accelerate status: " + accelerateStatus);

            // Upload a new object using the accelerate endpoint.
            s3Client.putObject(bucketName, keyName, "Test object for transfer
 acceleration");
            System.out.println("Object \"" + keyName + "\" uploaded with transfer
 acceleration.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

下列範例顯示如何使用在 AWS SDK for .NET 值區上啟用傳輸加速功能。如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TransferAccelerationTest
    {
        private const string bucketName = "*** bucket name ***";
```

```
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            EnableAccelerationAsync().Wait();
        }

        static async Task EnableAccelerationAsync()
        {
                try
                {
                    var putRequest = new PutBucketAccelerateConfigurationRequest
                    {
                        BucketName = bucketName,
                        AccelerateConfiguration = new AccelerateConfiguration
                        {
                            Status = BucketAccelerateStatus.Enabled
                        }
                    };
                    await
s3Client.PutBucketAccelerateConfigurationAsync(putRequest);

                    var getRequest = new GetBucketAccelerateConfigurationRequest
                    {
                        BucketName = bucketName
                    };
                    var response = await
s3Client.GetBucketAccelerateConfigurationAsync(getRequest);

                    Console.WriteLine("Acceleration state = '{0}' ",
response.Status);
                }
                catch (AmazonS3Exception amazonS3Exception)
                {
                    Console.WriteLine(
                        "Error occurred. Message:'{0}' when setting transfer
acceleration",
                        amazonS3Exception.Message);
                }
        }
    }
```

```
}
```

將物件上傳至已啟用 Transfer Acceleration 的儲存貯體時，您可以在建立用戶端時指定使用加速端點：

```
var client = new AmazonS3Client(new AmazonS3Config
            {
                RegionEndpoint = TestRegionEndpoint,
                UseAccelerateEndpoint = true
            }
```

Javascript

如需使用 SDK 啟用傳輸加速的範例 JavaScript，請參閱在 AWS SDK 中呼叫 putBucketAccelerate 設定作業以取得 JavaScript API 參考。AWS

Python (Boto)

如需有關使用適用於 Python 的開發套件啟用 Transfer Acceleration 的範例，請參閱 AWS SDK for Python (Boto3) API 參考中的 put_bucket_accelerate_configuration。

Other

如需使用其他 AWS SDK 的相關資訊，請參閱範例程式碼和程式庫。

使用 REST API

使用 REST API `PutBucketAccelerateConfiguration` 操作，以在現有儲存貯體上啟用加速設定。

如需詳細資訊，請參閱 Amazon 簡易儲存服務 API 參考PutBucketAccelerateConfiguration中的。

使用 Amazon S3 Transfer Acceleration 速度比較工具

您可以使用 Amazon S3 Transfer Acceleration 速度比較工具，比較各 Amazon S3 區域的加速和非加速上傳速度。速度比較工具在使用和不使用 Transfer Acceleration 的情況下，透過分段上傳，將檔案從您的瀏覽器傳送至各個 Amazon S3 區域。

您可以使用下列任一種方法來存取速度比較工具：

- 將下列 URL 複製到瀏覽器視窗中，以您 AWS 區域 正在使用的*## (###us-west-2) # #yourBucketName#############*：

```
https://s3-accelerate-speedtest.s3-accelerate.amazonaws.com/en/
accelerate-speed-comparsion.html?
region=region&origBucketName=yourBucketName
```

如需 Amazon S3 支援的區域清單，請參閱《AWS 一般參考》中的 Amazon S3 端點和配額。

- 使用 Amazon S3 主控台。

使用儲存體傳輸和用量的申請者付款儲存貯體

所有與儲存貯體相關聯的 Amazon S3 儲存與資料傳輸費用通常是由儲存貯體擁有者支付。但是，您可以將儲存貯體設定成申請者付款的儲存貯體。使用申請者付款儲存貯體，由申請者而非儲存貯體擁有者支付要求與從儲存貯體下載資料的費用。存放資料的費用一律由儲存貯體擁有者支付。

一般是在要分享資料，但不負擔其他資料存取相關費用時，將儲存貯體設定成申請者付款的儲存貯體。例如，您可能在建立可用大型資料集時使用申請者付款儲存貯體，例如郵遞區號目錄、參考資料、地理空間資訊或網路抓取資料。

> **Important**
>
> 如果對儲存貯體啟用申請者付款，則不允許匿名存取該儲存貯體。

您必須驗證所有與申請者付款儲存貯體有關的要求。請求身分驗證能讓 Amazon S3 識別申請者，並向他們索取使用申請者付款儲存貯體的費用。

如果請求者在提出請求之前擔任 AWS Identity and Access Management (IAM) 角色，則會針對該請求收取該角色所屬帳戶的費用。如需 IAM 角色的相關資訊，請參閱《IAM 使用者指南》中的 IAM 角色。

將值區設定為 Requester Pays 值區後，請求者必須告知他們瞭解要求和資料下載將收取費用。要顯示他們接受費用，請求者必須在其 API 請求中包含`x-amz-request-payer`為 DELETE，GET，HEAD，POST 和 PUT 請求的標題，或者在其 REST 請求中添加`RequestPayer`參數。對於 CLI 請求，請求者可以使用參數`--request-payer`。

Example — 刪除物件時使用請求者付費

若要使用下列 DeleteObjectVersionAPI 範例，請以您自己*user input placeholders*的資訊取代。

```
DELETE /Key+?versionId=VersionId HTTP/1.1
Host: Bucket.s3.amazonaws.com
x-amz-mfa: MFA
x-amz-request-payer: RequestPayer
x-amz-bypass-governance-retention: BypassGovernanceRetention
x-amz-expected-bucket-owner: ExpectedBucketOwner
```

如果請求者使用 RestoreObjectAPI 還原物件，只要x-amz-request-payer標頭或RequestPayer參數位於請求中，就會支援「要求者付費」；不過，請求者只會支付要求的費用。值區擁有者支付擷取費用。

申請者付款儲存貯體不支援下列作業：

- 匿名要求
- SOAP 要求
- 將申請者支付儲存貯體做為最終使用者日誌記錄的目標儲存貯體使用，反之亦然。但是，您可以開啟申請者付款儲存貯體的最終使用者日誌記錄，其中目標儲存貯體不是申請者支付儲存貯體。

申請者如何支付工作的費用

成功申請者付款要求的收費很直接：申請者支付資料傳輸與要求的費用，且儲存貯體擁有者支付資料儲存的費用。不過，下列情況會向儲存貯體擁有者收取要求的費用：

- 要求會傳回 AccessDenied (HTTP403 Forbidden) 錯誤，並在值區擁有者的個別 AWS 帳戶或 AWS 組織內啟動要求。
- 要求是 SOAP 要求。

如需「申請者付款」的詳細資訊，請參閱下列主題。

主題

- 設定儲存貯體上的申請者付款
- 使用 REST API 擷取 requestPayment 組態
- 從請求者付費值區下載物件

設定儲存貯體上的申請者付款

您可以將 Amazon S3 儲存貯體設定成申請者付款儲存貯體，因此是由申請者而非儲存貯體擁有者支付要求和資料下載成本費用。

本節提供如何使用主控台和 REST API 設定 Amazon S3 儲存貯體上申請者付款的範例。

使用 S3 主控台

為 S3 儲存貯體啟用申請者付款

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，按一下要啟用申請者付款的儲存貯體名稱。
3. 選擇 Properties (屬性)。
4. 在 Requester pays (申請者付款) 底下，選擇 Edit (編輯)。
5. 依序選擇 Enable (啟用) 和 Save changes (儲存變更)。

   Amazon S3 隨即會為儲存貯體啟用申請者付款，然後顯示 Bucket overview (儲存貯體概觀)。您會在 Requester pays (申請者付款) 底下看到 Enabled (已啟用)。

使用 REST API

只有儲存貯體擁有者可將儲存貯體的 `RequestPaymentConfiguration.payer` 組態值設成 `BucketOwner` (預設值) 或 `Requester`。`requestPayment` 資源為選擇性設定。儲存貯體預設不是申請者付款的儲存貯體。

若要將申請者付款的儲存貯體還原為一般的儲存貯體，請使用 `BucketOwner` 值。您一般會在將資料上傳至 Amazon S3 儲存貯體時使用 `BucketOwner`，然後將值設成 `Requester`，再於儲存貯體中發佈物件。

設定 requestPayment

- 使用 PUT 要求，在指定的儲存貯體上將 `Payer` 值設成 `Requester`。

```
PUT ?requestPayment HTTP/1.1
Host: [BucketName].s3.amazonaws.com
Content-Length: 173
```

```
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]

<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

如果請求成功，Amazon S3 會傳回類似如下的回應。

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
x-amz-request-charged:requester
```

您只可以設定儲存貯體層級的申請者付款。您無法設定儲存貯體內特定物件的申請者付款。

您可以隨時將儲存貯體設定成 `BucketOwner` 或 `Requester`。但需要經過幾分鐘才會讓新的組態生效。

> **Note**
>
> 分發預先簽章 URL 的儲存貯體擁有者，在將儲存貯體設定成申請者付款前應該慎思，特別是如果 URL 有很長的存活時間時。每次申請者使用利用儲存貯體擁有者登入資料的預先簽章 URL 時，儲存貯體擁有者都需要支付費用。

使用 REST API 擷取 requestPayment 組態

您可以要求資源 `Payer`，決定在儲存貯體上設定的 `requestPayment` 值。

傳回 requestPayment 資源

- 使用 GET 要求以取得 `requestPayment` 資源，如下列要求所示。

```
GET ?requestPayment HTTP/1.1
```

```
Host: [BucketName].s3.amazonaws.com
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

如果請求成功，Amazon S3 會傳回類似如下的回應。

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Type: [type]
Content-Length: [length]
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

此回應顯示 payer 值設成 Requester。

從請求者付費值區下載物件

因為會向從申請者付款儲存貯體下載資料的申請者收費，所以要求必須包含特殊的參數 x-amz-request-payer，它會確認申請者知道下載要收取費用。為存取申請者付款儲存貯體中的物件，要求必須包含下列項目之一。

- 在 DELETE、GET、HEAD、POST 和 PUT 請求的標題中要包含 x-amz-request-payer : requester
- 在已簽章的 URL 要求中要包含 x-amz-request-payer=requester

如果要求成功且向申請者收費，回應要包含標頭 x-amz-request-charged:requester。如果請求中沒有 x-amz-request-payer，Amazon S3 會傳回 403 錯誤並向儲存貯體擁有者收取請求的費用。

> **Note**
>
> 儲存貯體擁有者不需要在其要求中新增 x-amz-request-payer。

確保已在簽章運算中包含 x-amz-request-payer 及其值。如需詳細資訊，請參閱建構 CanonicalizedAmzHeaders 元素。

使用 REST API

從申請者付款儲存貯體下載物件

- 使用 GET 要求從申請者付款儲存貯體下載物件，如下列要求所示。

```
GET / [destinationObject] HTTP/1.1
Host: [BucketName].s3.amazonaws.com
x-amz-request-payer : requester
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

如果 GET 要求成功且向申請者收費，回應要包含 x-amz-request-charged:requester。

Amazon S3 會針對嘗試從申請者付款儲存貯體取得物件的請求傳回 Access Denied 錯誤。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的錯誤回應。

使用 AWS CLI

若要使用「請求者付款」值區下載物件 AWS CLI，請指定--request-payer requester為get-object請求的一部分。如需詳細資訊，請參閱《AWS CLI CLI 參考》中的取得 get-object。

儲存貯體的法規與限制

Amazon S3 存儲桶由創建它 AWS 帳戶 的人擁有。儲存貯體所有權不可轉移至另一個帳戶。

建立值區時，您可 AWS 區域 以選擇其名稱，並在中建立值區。建立儲存貯體後，您無法變更其名稱或區域。

命名儲存貯體時，請選擇與您或您的企業相關的名稱。避免使用與其他人相關聯的名稱。例如，您應該避免在您的儲存貯體名稱中使用 AWS 或 Amazon。

根據預設，您可以在每個儲存貯體中建立最多 100 個儲存貯體 AWS 帳戶。如果您需要更多的儲存貯體，可以提交配額增加請求，以便將帳戶儲存貯體配額增加至上限 1,000 個儲存貯體。無論您使用多個儲存貯體還是少數儲存貯體，效能都沒有差別。

> **Note**
>
> 您不需要為每個提交多個配額增加請求 AWS 區域。您的儲存貯體配額會套用至您的 AWS 帳戶。

如需如何增加儲存貯體配額的資訊，請參閱《AWS 一般參考》中的 AWS Service Quotas。

重複使用儲存貯體名稱

如果儲存貯體為空白，您可以將其刪除。刪除某個儲存貯體後，該名稱就可以重複使用。不過，刪除儲存貯體之後，您可能因為各種原因無法重複使用該名稱。

例如，當您刪除儲存貯體且名稱可以重複使用時，另一個 AWS 帳戶 可能會建立具有該名稱的儲存貯體。此外，在您可以重複使用已刪除儲存貯體的名稱之前，可能需要一段時間。如果想要使用相同的儲存貯體名稱，建議您不要刪除儲存貯體。

如需儲存貯體名稱的詳細資訊，請參閱儲存貯體命名規則。

物件和儲存貯體限制

儲存貯體大小沒有上限，也就是說您可以在儲存貯體中儲存的物件數量沒有限制。您可以將所有物件存放在單一儲存貯體，或者可以整理到多個不同的儲存貯體中。但是，您無法從另一個儲存貯體內建立儲存貯體。

儲存貯體操作

Amazon S3 的高可用性工程設計重點在於取得、放置、列出與刪除等操作。因為儲存貯體操作是針對集中化的全域資源空間運作，所以不建議在應用程式的高可用性程式碼路徑中建立、刪除或設定儲存貯體。最好是在不常執行的其他初始化或安裝例行作業中建立、刪除或設定儲存貯體。

儲存貯體命名和自動建立的儲存貯體

如果您的應用程式自動建立儲存貯體，請選擇不可能引起命名衝突的儲存貯體命名配置。如已採用某個儲存貯體名稱，請確保您的應用程式邏輯會選擇不同的儲存貯體名稱。

如需儲存貯體命名的詳細資訊，請參閱 儲存貯體命名規則。

在 Amazon S3 中上傳、下載和使用物件

在將資料存放在 Amazon S3 中時，您會使用稱為儲存貯體和物件的資源。儲存貯體是物件的容器。物件是一個檔案和任何描述該檔案的中繼資料。

若要將物件存放在 Amazon S3 中，您需要建立儲存貯體，然後將物件上傳到儲存貯體。當物件在儲存貯體中時，您可以開啟、下載和複製物件。當您不再需要物件或儲存貯體時，可以清理這些資源。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱什麼是 S3 Express One Zone？ 和 目錄值區。

> **Important**
>
> 在 Amazon S3 主控台中，當您針對物件選擇 Open (開啟) 或 Download As (下載為) 時，這些操作會建立預先簽章的 URL。在五分鐘內，任何有權存取這些預先簽章 URL 的人都可以存取您的物件。如需預先簽章 URL 的詳細資訊，請參閱使用預先簽章 URL。

使用 Amazon S3，您只需按實際用量付費。如需 Amazon S3 功能和定價的詳細資訊，請參閱 Amazon S3。若您是 Amazon S3 新客戶，可以免費試用 Amazon S3。如需詳細資訊，請參閱 AWS 免費方案。

主題

- 整理、列出和使用物件
- 使用預先簽章的 URL
- 使用 S3 Object Lambda 轉換物件

Amazon S3 物件概觀

Amazon S3 是一個物件存放區，它使用唯一金鑰值來儲存您要的任意物件數量。您可以將這些物件存放在一或多個儲存貯體中，而且每個物件的大小最多可達 5 TB。物件由下列各項所組成：

Key

您指派給物件的名稱。您可以使用物件金鑰來擷取該物件。如需詳細資訊，請參閱使用物件中繼資料。

版本 ID

在儲存貯體中，鍵加上版本 ID 即可識別唯一的物件。版本 ID 是將物件新增至儲存貯體時，由 Amazon S3 所產生的字串。如需詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。

Value

要存放的內容。

物件值可以是任意位元組的序列。物件的大小範圍可從零到 5 TB。如需詳細資訊，請參閱「上傳物件」。

中繼資料

一組名稱值對，可用來存放物件的相關資訊。您可以將中繼資料 (稱為使用者定義的中繼資料) 指派給 Amazon S3 中的物件。Amazon S3 也會將系統中繼資料指派給這些物件，用於管理物件。如需詳細資訊，請參閱使用物件中繼資料。

子資源

Amazon S3 使用子資源機制來存放物件專屬的其他資訊。因為子資源隸屬於物件，因此一律會與物件或儲存貯體等其他實體相關聯。如需詳細資訊，請參閱「物件子資源」。

存取控制資訊

您可以控制對存儲在 Amazon S3 中的物件的存取。Amazon S3 支援資源類型存取控制 (例如存取控制清單 (ACL) 和儲存貯體政策) 以及使用者類型存取控制。如需存取控制的詳細資訊，請參閱以下內容：

- 存取管理

- 適用於 Amazon S3 的 Identity and Access Management
- 設定 ACL

您的 Amazon S3 資源 (例如，儲存貯體與物件) 預設皆為私有。您必須明確授與許可，其他人才可存取這些資源。如需分享物件的詳細資訊，請參閱「使用預先簽章的 URL 來共用物件」。

標籤

您可以用標籤來分類儲存的物件、存取控制或成本分配。如需詳細資訊，請參閱 使用標籤分類儲存空間。

物件子資源

Amazon S3 定義一組與儲存貯體及物件相關聯的子資源。子資源從屬於物件。這意味著子資源本身不存在。它們一律會與物件或儲存貯體等其他實體相關聯。

下表列出與 Amazon S3 物件相關聯的子資源。

子資源	描述
acl	包含授予清單，其中指出被授予者及獲授予的許可。當您建立物件時，acl 會認定物件擁有者具有該物件的完整控制權。您可以擷取物件 ACL，或以更新過的授與清單取代。ACL 的任何更新都需要您取代現有的 ACL。如需 ACL 的詳細資訊，請參閱「存取控制清單 (ACL) 概觀」。

建立物件索引鍵名稱

物件索引鍵 (或索引鍵名稱) 可在 Amazon S3 儲存貯體中找出獨一的物件。物件中繼資料是一組名稱/值對。如需物件中繼資料的詳細資訊，請參閱「使用物件中繼資料」。

建立物件時，您可以指定能在儲存貯體中找出獨一物件的金鑰名稱。例如，在 Amazon S3 主控台上，當您反白某個儲存貯體時，即會顯示該儲存貯體中的物件清單。這些名稱即為物件金鑰。物件金鑰名稱是 Unicode 字元的序列，其 UTF-8 編碼長度最多為 1,024 個位元組。物件金鑰名稱區分大小寫。

> **Note**
>
> virtual-hosted-style 請求不支援值為「soap」的物件索引鍵名稱。對於使用「soap」的物件金鑰名稱值，必須改用路徑樣式 URL。

Amazon S3 資料模型是單層式結構：您建立儲存貯體，儲存貯體存放物件。子儲存貯體或子資料夾沒有階層。但您可以仿照 Amazon S3 主控台的做法，使用金鑰名稱字首以及分隔符號來推斷邏輯階層。Amazon S3 主控台支援資料夾的概念。如需如何從 Amazon S3 主控台編輯中繼資料的詳細資訊，請參閱「在 Amazon S3 主控台中編輯物件中繼資料」。

假設您的儲存貯體 (admin-created) 具有四個物件，其物件金鑰如下：

```
Development/Projects.xls

Finance/statement1.pdf

Private/taxdocument.pdf

s3-dg.pdf
```

主控台會使用索引鍵名稱字首 (Development/、Finance/ 和 Private/) 以及分隔符號 ('/')，來呈現資料夾結構。s3-dg.pdf 金鑰沒有字首，因此會直接在儲存貯體的根層級中顯示其物件。如果開啟 Development/ 資料夾，會看到其中內含 Projects.xlsx 物件。

- Amazon S3 支援儲存貯體與物件，且沒有任何階層。但是，透過在物件金鑰名稱中使用首碼和分隔符號，Amazon S3 主控台和 AWS SDK 可以推斷階層並引入資料夾的概念。
- Amazon S3 主控台會透過資料夾前綴和分隔符號值作為金鑰建立零位元組物件，以實作資料夾物件的建立。這些資料夾物件不會出現在主控台中，否則，它們的行為與任何其他對象一樣，可以通過 REST API， AWS CLI 和 AWS SDK 進行查看和操作。

物件索引鍵命名準則

您可以在物件索引鍵名稱中使用任何 UTF-8 字元。但是，在索引鍵名稱中使用特定字元可能對某些應用程式和通訊協定造成問題。下列準則可協助您提高與 DNS、網頁適用字元、XML 剖析器及其他 API 的合規。

安全字元

下列字元集通常可安心用於索引鍵名稱中。

英數字元	• 0-9 • a-z • A-Z
特殊字元	• 驚嘆號 (!)

- 連字號 (-)
- 底線 (_)
- 句號 (.)
- 星號 (*)
- 單引號 (')
- 左括號 (()
- 右括號 ())

有效的物件金鑰名稱範例如下：

- `4my-organization`
- `my.great_photos-2014/jan/myvacation.jpg`
- `videos/2014/birthday/video1.wmv`

> **Note**
>
> 使用 Amazon S3 主控台下載的金鑰名稱以句點「.」結尾的物件，該句點「.」會從下載物件的金鑰名稱中移除。若要下載保留在下載物件中的金鑰名稱以句點「.」結尾的物件，您必須使用 AWS Command Line Interface (AWS CLI)、 AWS SDK 或 REST API。
> 此外，也請注意以下字首限制：
>
> - 前置詞為「./」的物件必須使用 AWS Command Line Interface (AWS CLI)、 AWS SDK 或 REST API 上傳或下載。您無法使用 Amazon S3 主控台。
> - 字首為 "../" 的物件無法使用 AWS Command Line Interface (AWS CLI) 或 Amazon S3 主控台上傳。

可能需要特殊處理的字元

索引鍵名稱中的下列字元可能需要額外的程式碼處理，且有可能需要編碼為 URL 或是以十六進位參考。其中一部分是您的瀏覽器可能無法處理的不可列印字元，這些字元也需要特殊處理：

- & 符號
- 貨幣符號 ("$")
- ASCII 字元範圍：00 - 1F 十六進位 (0 - 31 十進位) 與 7F (127 十進位)

- @ 符號
- 等號 ("=")
- 分號 (";")
- 正斜線 ("/")
- 冒號 (":")
- 加號 ("+")
- 空格 - 在某些情況下，可能會遺失大量連續空格 (特別是多個空格)
- 逗號 (",")
- 問號 ("?")

需要避免的字元

我們建議您不要在金鑰名稱中使用下列字元，因為重要的特殊字元處理方式在所有應用程式中都不一致。

- 反斜線 ("\")
- 左大括弧 ("{")
- 無法列印的 ASCII 字元 (128 - 255 十進位字元)
- 插入號 ("^")
- 右大括弧 ("}")
- 百分比字元 ("%")
- 重音符號/反引號 ("`")
- 右方括號 ("]")
- 問號
- 「大於」符號 (">")
- 左方括號 ("[")
- 波狀符號 ("~")
- 「小於」符號 ("<")
- 井字號 ("#")
- 垂直分隔號/縱線字元 ("|")

XML 相關物件金鑰限制條件

正如 XML 標準 end-of-line 處理時所指定的，所有 XML 文字都會歸一化，使得單一歸位字元 (ASCII 碼 13) 和回車後跟一個換行符 (ASCII 碼 10) 都會被單一換行字元取代。為了確保在 XML 請求中正確剖析物件索引鍵，歸位字元和其他特殊字元插入 XML 標籤時，必須用其對等的 XML 實體程式碼取代。以下是此類特殊字元及其對等實體程式碼的清單：

- ' 作為 '
- " 作為 "
- & 作為 &
- < 作為 <
- > 作為 >
- \r 作為
 或
- \n 作為
 或

Example

下列範例說明了如何使用 XML 實體程式碼作為歸位字元的替代。此 DeleteObjects 請求刪除具有 key 參數的物件：/some/prefix/objectwith\rcarriagereturn (其中 \r 是歸位字元)。

```
<Delete xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Object>
    <Key>/some/prefix/objectwith&#13;carriagereturn</Key>
  </Object>
</Delete>
```

使用物件中繼資料

您可以在上傳物件到 Amazon S3 時設定物件中繼資料。物件中繼資料是一組名稱/值對。上傳物件之後，即無法修改物件中繼資料。修改物件中繼資料唯一的方式是製作物件的複本，再設定中繼資料。

建立物件時，您還可以指定能在儲存貯體中找出獨一物件的金鑰名稱。物件索引鍵 (或索引鍵名稱) 可在 Amazon S3 儲存貯體中找出獨一的物件。如需詳細資訊，請參閱「建立物件索引鍵名稱」。

Amazon S3 中的中繼資料有兩種：系統定義中繼資料和使用者定義中繼資料。下列章節提供有關系統定義和使用者定義中繼資料的詳細資訊。如需使用 Amazon S3 主控台編輯中繼資料的詳細資訊，請參閱「在 Amazon S3 主控台中編輯物件中繼資料」。

系統定義的物件中繼資料

若是存放在儲存貯體中的每個物件，Amazon S3 會保留一組系統中繼資料。Amazon S3 會視需要處理此系統中繼資料。例如，Amazon S3 會保留物件建立日期與大小中繼資料，而且在物件管理過程中會使用這項資訊。

系統中繼資料可分為兩類：

- 系統控制 – 物件建立日期之類的中繼資料由系統控制，只有 Amazon S3 才可修改此值。
- 使用者控制 – 另一種系統中繼資料的範例包括為物件所設定的儲存體方案，以及物件是否已啟用伺服器端加密，您可以控制這類系統中繼資料的值。若您的儲存貯體設定為網站，可能有時會希望將頁面要求重新導向至其他頁面或外部的 URL。在此情況下，網頁即為您儲存貯體中的物件。Amazon S3 會將頁面重新導向值另存為您可以控制其值的系統中繼資料。

  當您建立物件時，可以設定這些系統中繼資料項目的值，或視需要更新這些值。如需儲存體方案的詳細資訊，請參閱「使用 Amazon S3 儲存體方案」。

  Amazon S3 使用 AWS KMS 金鑰加密您的 Amazon S3 物件。 AWS KMS 僅加密物件資料。檢查總和以及指定的演算法會儲存為物件中繼資料的一部分。若物件要求伺服器端加密，檢查總和會以加密形式儲存。如需伺服器端加密的詳細資訊，請參閱「使用加密來保護資料」。

> **Note**
>
> PUT 要求標頭的大小限制為 8 KB。在 PUT 要求標頭中，系統定義中繼資料的大小限制為 2 KB。測量系統定義中繼資料大小的方式，是透過計算每個索引鍵和值之 US-ASCII 編碼中的位元組數目加總。

下表提供系統定義的中繼資料的清單，並指出您是否可更新這些中繼資料。

名稱	描述	使用者是否可以修改值？
`Date`	目前的日期與時間。	否
`Cache-Control`	用於指定快取政策的一般標頭欄位。	是

名稱	描述	使用者是否可以修改值？
Content-Disposition	物件表示資訊。	是
Content-Length	物件大小 (位元組)。	否
Content-Type	物件的類型。	是
Last-Modified	物件建立日期或上次修改日期，以最近者為準。對於分段上傳，物件建立日期是指啟動分段上傳的日期。	否
ETag	表示物件特定版本的實體標籤 (ETag)。對於未做為分段上傳上傳且未加密或透過使用 Amazon S3 受管金鑰 (SSE-S3) 之伺服器端加密來進行加密的物件，ETag 是資料的 MD5 Digest。	否
x-amz-server-side-encryption	表示是否已為物件啟用伺服器端加密的標頭，以及該加密是使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 還是使用 Amazon S3 受管加密金鑰 (SSE-S3)。如需詳細資訊，請參閱「使用伺服器端加密保護資料」。	是
x-amz-checksum-crc32 ，x-amz-checksum-crc32c ，x-amz-checksum-sha1 ，x-amz-checksum-sha256	包含物件檢查總和或摘要的標題。根據您指示 Amazon S3 使用的檢查總和演算法，最多可以一次設定其中一個標頭。如需選擇檢查總和演算法的詳細資訊，請參閱「檢查物件完整性」。	否
x-amz-version-id	物件版本控制。當您啟用儲存貯體的版本控制時，Amazon S3 會將版本 ID 指派給已新增至儲存貯體的物件。如需詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。	否

名稱	描述	使用者是否可以修改值？
x-amz-delete-marker	指出物件是否為刪除標記的布林值標記。此標記僅用於已啟用版本控制的版本控制，	否
x-amz-storage-class	用於存放物件的儲存體方案。如需詳細資訊，請參閱「使用 Amazon S3 儲存體方案」。	是
x-amz-website-redirect-location	將關聯物件重新導向至相同儲存貯體中的其他物件或外部 URL 的標題。如需詳細資訊，請參閱「(選用) 配置網頁重新導向」。	是
x-amz-server-side-encryption-aws-kms-key-id	標頭，指出用來加密物件之 AWS KMS 對稱加密 KMS 金鑰的識別碼。只有當 x-amz-server-side-encryption 標題存在且具有 aws:kms 值的時候，才會使用此標題。	是
x-amz-server-side-encryption-customer-algorithm	指出是否已啟用由客戶提供加密金鑰的伺服器端加密 (SSE-C) 的標題。如需詳細資訊，請參閱「搭配客戶提供的金鑰 (SSE-C) 使用伺服器端加密」。	是
x-amz-tagging	物件的標籤集。標籤集必須編碼為 URL 查詢參數。	是

使用者定義的物件中繼資料

您也可以在上傳物件時，將中繼資料指派給物件。您可以在傳送 PUT 或 POST 要求來建立物件時，以名稱/值 (金鑰/值) 對的形式提供這項選用資訊。當您使用 REST API 上傳物件時，選用的使用者定義中繼資料名稱必須以 x-amz-meta- 開頭，以便與其他 HTTP 標頭有所區別。當您使用 REST API 擷取物件時，會傳回此字首。當您使用 SOAP API 上傳物件時，不需要此字首。當您使用 SOAP API 擷取物件時，不論用於上傳物件的 API 為何，都會移除此字首。

> **Note**
>
> HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。我們建議您使用 REST API 或 AWS 開發套件，而不是使用 SOAP。

透過 REST API 擷取中繼資料時，Amazon S3 會將具有相同名稱 (忽略大小寫) 的標頭合併成一份以逗號分隔的清單。如果某些中繼資料包含無法列印的字元，就不會傳回。反之，傳回的會是 `x-amz-missing-meta` 標頭，其值為無法列印的中繼資料項目數。該 `HeadObject` 動作從一個物件擷取中繼資料，而不傳回物件本身。如果您只對物件的中繼資料感興趣，此操作非常有用。若要使用 `HEAD`，您必須具有物件的 `READ` 存取權。如需詳細資訊，請參閱 Amazon 簡易儲存服務 API 參考HeadObject中的。

使用者定義的中繼資料是一組金鑰/值對。Amazon S3 會以小寫存放使用者定義的中繼資料金鑰。

Amazon S3 會允許在您的中繼資料值中有任意 Unicode 字元。

為了避免有關呈現這些中繼資料值的問題，使用 REST 時應該遵從使用 US-ASCII 字元，以及使用 SOAP 或透過 `POST` 的瀏覽器式上傳時應該遵從使用 UTF-8。

在中繼資料值中使用非 US-ASCII 字元時，會檢查提供的 Unicode 字串是否有非 US-ASCII 字元。這些標題的值是在儲存之前依據 RFC 2047 僅需解碼，然後依據 RFC 2047 進行編碼，以便在傳回之前使其能夠安全地以電子郵件寄送。如果字串只包含 US-ASCII 字元，則會依原樣呈現。

以下是範例。

```
PUT /Key HTTP/1.1
Host: DOC-EXAMPLE-BUCKET1.s3.amazonaws.com
x-amz-meta-nonascii: ÄMÄZÕÑ S3

HEAD /Key HTTP/1.1
Host: DOC-EXAMPLE-BUCKET1.s3.amazonaws.com
x-amz-meta-nonascii: =?UTF-8?B?w4PChE3Dg8KEWsODwpXDg8KRIFMz?=

PUT /Key HTTP/1.1
Host: DOC-EXAMPLE-BUCKET1.s3.amazonaws.com
x-amz-meta-ascii: AMAZONS3

HEAD /Key HTTP/1.1
Host: DOC-EXAMPLE-BUCKET1.s3.amazonaws.com
```

```
x-amz-meta-ascii: AMAZONS3
```

> **Note**
>
> PUT 要求標頭的大小限制為 8 KB。在 PUT 要求標頭中，使用者定義中繼資料的大小限制為 2 KB。測量使用者定義中繼資料大小的方式，是透過計算每個金鑰和值之 UTF-8 編碼中的位元組數目加總。

如需透過建立物件複本、修改它以及取代舊物件或建立新版本來變更上傳之後物件的中繼資料的相關資訊，請參閱 在 Amazon S3 主控台中編輯物件中繼資料。

在 Amazon S3 主控台中編輯物件中繼資料

您可以使用 Amazon S3 主控台編輯現有 S3 物件的中繼資料。上傳物件時，Amazon S3 會設定某些中繼資料。例如，`Content-Length` 和 `Last-Modified` 是使用者無法修改的系統定義物件中繼資料欄位。

您也可以在上傳物件時設定一些中繼資料，稍後在需求變更時進行編輯。例如，您可能有一組最初儲存在 `STANDARD` 儲存類別中的物件。隨著時間的推移，您可能不再需要讓這些資料高度可用。因此，您可透過將 `GLACIER` 索引鍵的值從 `x-amz-storage-class` 編輯為 `STANDARD`，從而將存儲類更改為 `GLACIER`。

> **Note**
>
> 在 Amazon S3 中編輯物件中繼資料時，請考慮下列問題：
>
> - 此動作會建立具有更新設定和上次修改日期的物件複本。如果啟用 S3 版本控制，則系統會建立物件的新版本，且現有物件會變成較舊的版本。如果未啟用 S3 版本控制，則物件的新複本會取代原始物件。 AWS 帳戶 與變更屬性的 IAM 角色相關聯的也會成為新物件或 (物件版本) 的擁有者。
> - 若要使用 Amazon S3 主控台編輯具有使用者定義標籤之物件的中繼資料，您還必須擁有該`s3:GetObjectTagging`權限。如果您使用 Amazon S3 主控台編輯沒有使用者定義標籤但大小超過 16 MB 的物件的中繼資料，則還必須擁有該`s3:GetObjectTagging`權限。
>
>   如果目的地儲存貯體原則拒絕該`s3:GetObjectTagging`動作，則會更新物件的中繼資料，但使用者定義的標籤會從物件中移除，而且您會收到錯誤訊息。
> - 編輯中繼資料會更新現有金鑰名稱的值。

- 使用客戶提供的加密金鑰 (SSE-C) 加密的物件無法經由主控台複製。您必須使用 AWS CLI、 AWS 開發套件或 Amazon S3 REST API。

> **Warning**
>
> 編輯資料夾的中繼資料時，請等待 `Edit metadata` 操作完成，然後再將新物件新增至資料夾。否則，新物件可能也會被編輯。

下列主題說明如何使用 Amazon S3 主控台編輯物件的中繼資料。

編輯系統定義中繼資料

您可以配置 S3 物件的一些 (非全部) 系統中繼資料。如需系統定義的中繼資料清單以及您是否可以修改其值，請參閱「系統定義的物件中繼資料」。

編輯物件的系統定義中繼資料

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 導覽至 Amazon S3 儲存貯體或資料夾，然後選取含有您要編輯之中繼資料物件名稱左側的核取方塊。
3. 在 Actions (動作) 功能表上，選擇 Edit actions (編輯動作)，然後選擇 Edit metadata (編輯中繼資料)。
4. 檢閱列出的物件，然後選擇 Add metadata (新增中繼資料)。
5. 對於中繼資料類型，請選取系統定義。
6. 指定唯一的金鑰和中繼資料值。
7. 若要編輯其他中繼資料，請選擇新增中繼資料 您也可以選擇「移除」來移除一組 type-key-values。
8. 完成後，選擇 Edit metadata (編輯中繼資料)，Amazon S3 會編輯指定物件的中繼資料。

編輯使用者定義中繼資料

您可以結合中繼資料前綴和您選擇建立自訂金鑰的名稱 `x-amz-meta-`，來編輯物件的使用者定義中繼資料。例如，如果您新增自訂名稱 `alt-name`，則中繼資料金鑰會是 `x-amz-meta-alt-name`。

使用者定義的中繼資料最大可達 2 KB。若要計算使用者定義中繼資料的總大小，須加總 UTF-8 編碼的每個索引鍵和值的位元組數。金鑰與其值都必須符合 US-ASCII 標準。如需詳細資訊，請參閱「使用者定義的物件中繼資料」。

編輯物件的使用者定義中繼資料

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇您要向其新增中繼資料之儲存貯體的名稱。

   您也可以選擇性地導覽至資料夾。
3. 在 Objects (物件) 清單中，選取要新增中繼資料之物件名稱旁的核取方塊。
4. 在 Actions (動作) 功能表上，選擇 Edit metadata (編輯中繼資料)。
5. 檢閱列出的物件，然後選擇 Add metadata (新增中繼資料)。
6. 對於中繼資料 Type (類型)，選擇 User-defined (使用者定義)。
7. 在 `x-amz-meta-` 後輸入唯一的自訂索引鍵。同時輸入中繼資料值。
8. 若要新增其他中繼資料，請選擇 Add metadata (新增中繼資料)。您也可以選擇「移除」來移除一組 type-key-values。
9. 選擇 Edit metadata (編輯中繼資料)。

   Amazon S3 會編輯指定物件的中繼資料。

上傳物件

當您將檔案上傳至 Amazon S3 時，該檔案會另存為 S3 物件。物件是由檔案資料與說明物件的中繼資料所組成。儲存貯體中可以有不限數目的物件。您需要儲存貯體的寫入許可，才能將檔案上傳至 Amazon S3 儲存貯體。如需存取許可的詳細資訊，請參閱「適用於 Amazon S3 的 Identity and Access Management」。

您可以將任何檔案類型 (影像、備份、資料、影片等) 上傳至 S3 儲存貯體。使用 Amazon S3 主控台可上傳的檔案大小上限為 160 GB。若要上傳大於 160 GB 的檔案，請使用 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 Amazon S3 REST API。

如果上傳物件的金鑰名稱已存在於啟用版本控制的儲存貯體中，Amazon S3 會建立另一個物件版本，而不是取代現有的物件。如需版本控制的詳細資訊，請參閱「使用 S3 主控台」。

視上傳資料大小之不同，Amazon S3 提供下列選項：

- 使用 AWS SDK、REST API 在單一作業中上傳物件，或 AWS CLI— 透過單一PUT作業，您可以上傳大小最多 5 GB 的單一物件。
- 使用 Amazon S3 主控台上傳單一物件– – 使用 Amazon S3 主控台，您可以上傳大小最多 160 GB 的單一物件。
- 使用 AWS SDK、REST API 分部分上傳物件，或 AWS CLI — 使用多部分上傳 API 作業，您可以上傳單一大型物件，大小最多可達 5 TB。

  分段上傳 API 操作是專為改善較大型物件上傳體驗所設計。您可以上傳零件中的物件。這些物件部分可個別、依任何順序以及同時上傳。您可以為大小介於 5 MB 到 5 TB 之間的物件使用分段上傳。如需詳細資訊，請參閱「使用分段上傳來上傳和複製物件」。

上傳物件時，根據預設，物件使用伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 進行自動加密。當您下載物件時，該物件會解密。如需詳細資訊，請參閱 對 Amazon S3 儲存貯體設定預設伺服器端加密行為 及 使用加密來保護資料。

上傳物件時，若您想要使用不同類型的預設加密，也可以在 S3 PUT 請求中使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 指定伺服器端加密，或在目標儲存貯體中設定預設加密組態，以使用 SSE-KMS 加密資料。如需 SSE-KMS 的詳細資訊，請參閱「使用 AWS KMS (SSE-KMS) 指定伺服器端加密」。若您想要使用其他帳戶的 KMS 金鑰，您必須具有該金鑰的使用權限。如需詳細了解 KMS 金鑰跨帳戶權限，請參閱《AWS Key Management Service 開發人員指南》中的建立其他帳戶可使用的 KMS 金鑰。

如果您在 Amazon S3 中遇到「拒絕存取」(403 禁止) 錯誤，請參閱針對 Amazon S3 中的拒絕存取 (403 禁止) 錯誤進行疑難排解以進一步了解其常見原因。

使用 S3 主控台

此程序說明如何使用主控台，將物件和資料夾上傳至 Amazon S3 儲存貯體。

上傳物件時，物件索引鍵名稱為檔案名稱加上任何選擇性的字首。在 Amazon S3 主控台中，您可以建立資料夾來整理物件。在 Amazon S3 中，資料夾表示為物件索引鍵名稱中的字首。如果您在 Amazon S3 主控台中將個別物件上傳到資料夾，則物件索引鍵名稱中會包含該資料夾名稱。

例如，如果您將名為 `sample1.jpg` 的物件上傳到名為 `backup` 的資料夾，則索引鍵名稱為 `backup/sample1.jpg`。不過，此物件在主控台中會顯示為 `sample1.jpg` 資料夾中的 `backup`。如需金鑰名稱的詳細資訊，請參閱「使用物件中繼資料」。

> **Note**
>
> 如果您在 Amazon S3 主控台中重新命名物件或變更物件的任何屬性 (例如儲存類別、加密或中繼資料)，則會建立新物件來取代舊的物件。如果啟用 S3 版本控制，則系統會建立物件的新版本，且現有物件會變成較舊的版本。變更屬性的角色也會成為新物件 (或物件版本) 的擁有者。

當您上傳資料夾時，Amazon S3 會將指定資料夾中的所有檔案與子資料夾上傳至您的儲存貯體。然後會指派一個由已上傳檔案名稱與資料夾名稱所組成的物件金鑰名稱。例如，如果您上傳一個名為 `/images` 的資料夾，其中包含兩個檔案 `sample1.jpg` 與 `sample2.jpg`，Amazon S3 會上傳檔案，然後指派對應的索引鍵名稱 `images/sample1.jpg` 與 `images/sample2.jpg`。金鑰名稱包含資料夾名稱作為字首。Amazon S3 主控台只會顯示最後一個 `/` 後面的金鑰名稱部分。例如，在 `images` 資料夾中，`images/sample1.jpg` 與 `images/sample2.jpg` 物件會顯示為 `sample1.jpg` 與 `sample2.jpg`。

將資料夾和檔案上傳至 S3 儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇您要上傳資料夾或檔案的目標儲存貯體名稱。
4. 選擇 Upload (上傳)。
5. 在 Upload (上傳) 視窗中，執行下列其中一個操作：
   - 將檔案和資料夾拖放至 Upload (上傳) 視窗。
   - 選擇新增檔案或新增資料夾，選擇要上傳的檔案或資料夾，然後選擇開啟。
6. 若要啟用多版本設定，請在 Destination (目的地) 下選擇 Enable Bucket Versioning (啟用儲存貯體版本控制)。
7. 若要上傳列出的檔案和資料夾，而不設定其他上傳選項，請選擇頁面底部的 Upload (上傳)。

   Amazon S3 會上傳您的物件和資料夾。上傳完成後，您可以在上傳：狀態頁面上看到成功訊息。

設定其他物件屬性

1. 若要變更存取控制清單許可，請選擇Permissions (許可)。
2. 在存取控制清單 (ACL)中，編輯許可。

如需物件存取許可的資訊，請參閱「使用 S3 主控台來設定物件的 ACL 許可」。您可以針對您上傳的所有檔案，將物件的讀取存取許可授予大眾 (全世界的所有人)。但是，建議不要變更公有讀取存取的預設設定。授予公有讀取存取適用於一小部分的使用情況，例如當儲存貯體用於網站時。您一律可以在上傳物件之後變更物件許可。

3. 若要設定其他附加屬性，請選擇Properties (屬性)。
4. 在儲存類別下，選擇您要上傳的檔案的儲存類別。

   如需儲存體方案的詳細資訊，請參閱「使用 Amazon S3 儲存體方案」。
5. 若要更新物件的加密設定，請在 Server-side encryption settings (伺服器端加密設定) 下，執行下列操作。

   a. 選擇 Specify an encryption key (指定加密金鑰)。

   b. 在加密設定底下，選擇使用預設加密的儲存貯體設定或覆寫預設加密的儲存貯體設定。

   c. 若您選擇覆寫預設加密的儲存貯體設定，則您必須設定下列加密設定。

      - 若要使用 Amazon S3 管理的金鑰加密上傳的檔案，請選擇 Amazon S3 受管金鑰 (SSE-S3)。

        如需詳細資訊，請參閱 使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密。
      - 若要使用 AWS Key Management Service () 中儲存的金鑰加密上傳的檔案，請選擇金AWS Key Management Service 鑰 (SSE-KMS AWS KMS)。然後針對 AWS KMS 金鑰，選擇下列其中一個選項：
        - 若要從可用的 KMS 金鑰清單中選擇，請選擇從 AWS KMS keys中選擇，然後從可用金鑰清單中選擇您的 KMS 金鑰。

          AWS 受管金鑰 (aws/s3) 和您的客戶管理金鑰都會出現在此清單中。如需詳細了解客戶受管金鑰，請參閱《AWS Key Management Service 開發人員指南》中的客戶金鑰和 AWS 金鑰。
        - 若要輸入 KMS 金鑰 ARN，請選擇 [輸入 AWS KMS key ARN]，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。
        - 若要在 AWS KMS 主控台中建立新的客戶管理金鑰，請選擇 [建立 KMS 金鑰]。

          如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。

> **Important**
>
> 您只能使用與值區相同 AWS 區域 的 KMS 金鑰。Amazon S3 主控台僅會列出與儲存貯體位於相同區域的前 100 個 KMS 金鑰。若要使用未列出的 KMS 金鑰，必須輸入 KMS 金鑰 ARN。若您想要使用其他帳戶的 KMS 金鑰，您必須先具有該金鑰的使用權限，然後輸入 KMS 金鑰 ARN。
>
> Amazon S3 僅支援對稱加密 KMS 金鑰，而不支援非對稱 KMS 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的識別對稱和非對稱 KMS 金鑰。

6. 若要使用額外的檢查總和，請選擇 On (開啟)。然後為 檢查總和函數，選擇您要使用的函數。Amazon S3 在接收完整物件後計算並存儲檢查總和的值。您可以使用 預先計算的值 方塊以提供預先計算的值。如果您這樣做，Amazon S3 會將您提供的值與其計算的值進行比較。如果兩個值不匹配，Amazon S3 將產生錯誤。

   通過額外的檢查總和，您可以指定要用於驗證資料的檢查總合演算法。如需額外的檢查總和詳細資訊，請參閱「檢查物件完整性」。

7. 若要將標籤新增至您要上傳的所有物件，請選擇 Add tag (新增標籤)。在金鑰欄位中輸入標籤名稱。輸入標籤值。

   物件標記提供您一個分類儲存的方法。每個標籤都是金鑰值對。金鑰與標記值皆區分大小寫。每物件最多可有 10 個標籤。標籤金鑰最長可包含 128 個 Unicode 字元；標籤值最長可包含 255 個 Unicode 字元。如需物件標籤的詳細資訊，請參閱 使用標籤分類儲存空間。

8. 若要新增中繼資料，請選擇 Add metadata (新增中繼資料)。

   a. 在 Type (類型) 下，選擇 System defined (系統定義) 或 User defined (使用者定義)。

      對於系統定義的中繼資料，您可以選取常見的 HTTP 標頭，例如 Content-Type 和 Content-Disposition。如需系統定義的中繼資料清單及是否可新增值的資訊，請參閱「系統定義的物件中繼資料」。以字首 `x-amz-meta-` 開頭的所有中繼資料都會視為使用者定義的中繼資料。使用者定義的中繼資料會隨著物件一起存放，並在下載物件時傳回。金鑰及其值都必須符合 US-ASCII 標準。使用者定義的中繼資料最大可達 2 KB。如需系統定義與使用者定義中繼資料的詳細資訊，請參閱 「使用物件中繼資料」。

   b. 針對 Key (金鑰) 中，選擇一個金鑰。

   c. 輸入金鑰的值。

9. 若要上傳物件，請選擇 Upload (上傳)。

Amazon S3 會上傳您的物件。上傳完成後，您可以在 Upload: status (上傳：狀態) 頁面上看到成功訊息。

10. 選擇 Exit (退出)。

使用 AWS 軟體開發套件

您可以使用 AWS 開發套件在 Amazon S3 上傳物件。SDK 提供包裝函式程式庫，供您輕鬆地上傳資料。如需詳細資訊，請參閱支援的 SDK 清單。

以下是幾個選取 SDK 的範例：

.NET

下列 C# 程式碼範例會使用兩個 PutObjectRequest 請求建立兩個物件：

- 第一個 PutObjectRequest 請求會將文字字串儲存為範例物件資料。它也會指定儲存貯體和物件金鑰名稱。
- 第二個 PutObjectRequest 請求會指定檔案名稱來上傳檔案。此請求也會指定 ContentType 標頭以及選用的物件中繼資料 (標題)。

如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadObjectTest
    {
        private const string bucketName = "*** bucket name ***";
        // For simplicity the example creates two objects from the same file.
        // You specify key names for these objects.
        private const string keyName1 = "*** key name for first object created ***";
        private const string keyName2 = "*** key name for second object created
 ***";
        private const string filePath = @"*** file path ***";
```

```
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.EUWest1;

        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            WritingAnObjectAsync().Wait();
        }

        static async Task WritingAnObjectAsync()
        {
            try
            {
                // 1. Put object-specify only key name for the new object.
                var putRequest1 = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName1,
                    ContentBody = "sample text"
                };

                PutObjectResponse response1 = await
client.PutObjectAsync(putRequest1);

                // 2. Put the object-set ContentType and add metadata.
                var putRequest2 = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName2,
                    FilePath = filePath,
                    ContentType = "text/plain"
                };

                putRequest2.Metadata.Add("x-amz-meta-title", "someTitle");
                PutObjectResponse response2 = await
client.PutObjectAsync(putRequest2);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an
object"
```

```
                        , e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Unknown encountered on server. Message:'{0}' when writing an
 object"
                    , e.Message);
            }
        }
    }
}
```

Java

以下範例將建立兩個物件。第一個物件擁有文字字串形式的資料，而第二個物件是一個檔案。該範例會直接在對 AmazonS3Client.putObject() 的呼叫中指定儲存貯體名稱、物件金鑰和文字資料，來建立第一個物件。該範例會使用 PutObjectRequest 指定儲存貯體名稱、物件金鑰和檔案路徑，來建立第二個物件。PutObjectRequest 也會指定 ContentType 標頭和標題中繼資料。

如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;

import java.io.File;
import java.io.IOException;

public class UploadObject {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String stringObjKeyName = "*** String object key name ***";
        String fileObjKeyName = "*** File object key name ***";
        String fileName = "*** Path to file to upload ***";
```

```
        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .build();

            // Upload a text string as a new object.
            s3Client.putObject(bucketName, stringObjKeyName, "Uploaded String
 Object");

            // Upload a file as a new object with ContentType and title specified.
            PutObjectRequest request = new PutObjectRequest(bucketName,
 fileObjKeyName, new File(fileName));
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentType("plain/text");
            metadata.addUserMetadata("title", "someTitle");
            request.setMetadata(metadata);
            s3Client.putObject(request);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

JavaScript

下列範例會將現有檔案上傳至特定區域的 Amazon S3 儲存貯體。

```
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new PutObjectCommand({
    Bucket: "test-bucket",
```

```
    Key: "hello-s3.txt",
    Body: "Hello S3!",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

PHP

此範例會引導您使用來自的類別， AWS SDK for PHP 以上傳大小最大為 5 GB 的物件。若是較大的檔案，則必須使用分段上傳 API 操作。如需詳細資訊，請參閱 使用分段上傳來上傳和複製物件。

有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

Example - 上傳資料以在 Amazon S3 儲存貯體中建立物件

下列 PHP 範例會使用 `putObject()` 方法上傳資料，以在指定的儲存貯體中建立物件。

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

try {
    // Upload data.
    $result = $s3->putObject([
        'Bucket' => $bucket,
        'Key'    => $keyname,
        'Body'   => 'Hello, world!',
        'ACL'    => 'public-read'
    ]);
```

```
    // Print the URL to the object.
    echo $result['ObjectURL'] . PHP_EOL;
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

AWS SDK for Ruby -版本 3 有兩種將對象上傳到 Amazon S3 的方法。第一個方法使用受管檔案上傳工具，可讓您更輕鬆地從磁碟上傳任意大小的檔案。若要使用受管理的檔案上傳工具方法：

1. 建立 `Aws::S3::Resource` 類別的執行個體。
2. 依儲存貯體名稱與金鑰參考目標物件。物件存在於儲存貯體中，並具有可識別每個物件且不重複的金鑰。
3. 在物件上呼叫 `#upload_file`。

Example

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
```

```
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

第二種方式 AWS SDK for Ruby -第 3 版可以上傳物件使用的#put方法Aws::S3::Object。如果物件是字串或非磁碟上檔案的 I/O 物件，即可使用此方法。若要使用此方法：

1. 建立 Aws::S3::Resource 類別的執行個體。
2. 依儲存貯體名稱與金鑰參考目標物件。
3. 呼叫 #put，傳入字串或 I/O 物件。

Example

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, "rb") do |file|
```

```
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

使用 REST API

您可以傳送 REST 請求以上傳物件。您可以傳送 PUT 請求，以單一操作上傳資料。如需詳細資訊，請參閱 PUT 物件。

使用 AWS CLI

您可以傳送 PUT 請求，以便在單一操作中上傳多達 5 GB 的物件。如需詳細資訊，請參閱《AWS CLI 命令參考》中的 `PutObject` 範例。

使用分段上傳來上傳和複製物件

您可利用分段上傳，將單一物件以一組組件進行上傳。每個組件都是物件資料的接續部分。您可依任何順序分別上傳這些物件組件。若任何組件的傳輸失敗，您可再次傳輸該組件，而不會影響其他組件。當物件的所有組件都全部上傳完後，Amazon S3 會將這些組件組合起來建立該物件。一般而言，當物件大小達到 100 MB 時，應考慮使用分段上傳，而不是以單次操作上傳物件。

使用分段上傳具備下列優勢：

- 改善輸送量 - 您可平行上傳各組件以改進輸送量。
- 快速從任何網路問題復原 - 組件大小若較小，對於重新開始因為網路發生錯誤而上傳失敗的影響可降到最低。
- 暫停及繼續上傳物件 - 您可在一段時間內上傳物件組件。啟動分段上傳之後就不會過期；您必須明確地完成或中止分段上傳。
- 在您知道最終物件大小前開始上傳 - 您可在建立物件的同時上傳物件。

建議您依照下列方式使用分段上傳：

- 若您透過穩定的高頻寬網路上傳大型物件，使用分段上傳可同時上傳多個物件部分以取得多執行緒效能，因而完全善用可用的頻寬使用量。
- 若是透過不穩定的網路進行上傳，使用分段上傳可避免上傳重新開始，因而此更快從網路故障中復原。使用分段上傳時，只有上傳期間遭到中斷的部分才需要重試上傳。您不需要從頭開始重新上傳物件。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱什麼是 S3 Express One Zone？ 和 目錄值區。如需使用分段上傳搭配 S3 Express One Zone 和目錄儲存貯體的詳細資訊，請參閱 搭配目錄儲存貯體使用多部分上傳。

分段上傳程序

分段上傳是三步驟的程序：啟動上傳功能、上傳物件的各組件，以及在上傳完所有分段後，完成分段上傳。Amazon S3 一收到完整的分段上傳要求時，就會從上傳的組件建構物件，然後您即可像存取儲存貯體中的任何其他物件一樣存取該物件。

您可以列出所有進行中的分段上傳，或是取得特定分段上傳之已上傳組件的清單。本節會一一說明這些操作。

啟動分段上傳

當您傳送要求要啟動分段上傳時，Amazon S3 會傳回具有上傳 ID 的回應，其為分段上傳的唯一識別符。每次上傳分段各組件、列出各組件、完成上傳或停止上傳時，都必須納入此上傳 ID。若希望提供任何中繼資料，說明正在上傳中的物件，您必須在要求中提供它，以啟動分段上傳。

組件上傳

上傳某個分段組件時，除了上傳 ID 之外，還必須指定組件編號。您可選擇 1 到 10,000 之間的任何組件編號。組件編號可找出獨特的某個組件，以及其在上傳中物件內的位置。您選擇的組件編號無須為連續的號碼 (例如，其可為 1、5 和 14)。若使用和前一個上傳組件相同的組件編號上傳新的組件，將會覆寫前一個已上傳的組件。

當您上傳零件時，Amazon S3 會傳回零件的實體標籤 (ETag) 作為回應中的標頭。您必須記錄每個上傳組件的組件編號與 ETag 值。後續的要求中必須包含這些值，才能完成分段上傳。每個部分在上傳時都會有自己的 ETag。但是，一旦多部分上傳完成並合併所有零件，所有零件都將在一個 ETag 下作為總和檢查碼的總和檢查碼。

> **Note**
>
> 在您啟動分段上傳及上傳一或多個組件之後，您必須完成或停止分段上傳，以停止繼續收取存放已上傳組件的費用。只有在完成或停止分段上傳之後，Amazon S3 才會釋出片段儲存體，然後停止收取片段的儲存費用。
>
> 停止分段上傳之後，即無法再次使用該上傳 ID 上傳任何組件。如有任何分段上傳正在進行，在您中止上傳後仍有可能成功或失敗。若要確保釋出所有組件所使用之全部的儲存體，您必須在所有上傳組件都已完成後，再停止分段上傳。

完成分段上傳

當您完成分段上傳時，Amazon S3 會根據組件編號以遞增順序串連各個組件，建立物件。啟動分段上傳要求中若已提供任何物件中繼資料，Amazon S3 就會建立該中繼資料與物件之間的關聯性。成功完成請求之後，這些片段就不再存在。

您的完成分段上傳要求，必須包含上傳 ID 以及組件編號和相對應的 ETag 值的清單。Amazon S3 回應包含的 ETag 可識別獨特的物件資料組合。這個 ETag 不一定是物件數據的 MD5 雜湊。

分段上傳呼叫範例

對於此範例，假設您正在為 100 GB 檔案產生分段上傳。在此情況下，您會為整個程序執行以下 API 呼叫。總共會有 1002 個 API 呼叫。

- 一個 CreateMultipartUpload 呼叫以啟動此程序。
- 1000 項個別 UploadPart 呼叫，每個都上傳 100 MB 的部分，總大小為 100 GB。
- 一個 CompleteMultipartUpload 呼叫以結束此程序。

分段上傳清單

您可列出特定分段上傳的組件或所有進行中之分段上傳。列出組件操作會傳回特定分段上傳之已上傳組件的資訊。Amazon S3 會為每項列出的組件要求，傳回指定分段上傳組件的資訊，上限為 1,000 個組件。若分段組件上傳中有超過 1,000 個組件，您必須傳送一連串的列出組件要求，才可擷取所有組件。請注意，傳回的組件清單不包含尚未完成之上傳的組件。使用列出分段上傳操作，即可取得正在進行中的分段上傳清單。

進行中的分段上傳是您已啟動但尚未完成或已停止的上傳。每個要求最多可傳回 1,000 個分段上傳。若正在進行超過 1,000 個的分段上傳，您必須另行傳送請求以擷取剩餘的分段上傳。傳回的清單僅用於進行驗證。傳送完成分段上傳請求時，請不要使用此清單的結果。而是在上傳 Amazon S3 傳回的組件與相對應之 ETag 值時，保有您自己的組件編號清單。

使用分段上傳操作的檢查總和

當您將物件上傳到 Amazon S3 時，可指定用於 Amazon S3 使用的檢查總和演算法。預設情況下，Amazon S3 使用 MD5 來驗證資料的完整性；但是，您可以指定使用額外的檢查總和演算法。使用 MD5 時，Amazon S3 會在上傳完成後計算整個分段上傳物件的檢查總和。此檢查總和不是整個物件的檢查總和，而是每個單獨部分檢查總和的檢查總和。

當您指示 Amazon S3 使用額外的檢查總和時，Amazon S3 會計算每個部分的檢查總和值並儲存這些值。您可以使用 API 或 SDK 搭配 `GetObject` 或 `HeadObject` 來找回各個部分的檢查總和。如果想要擷取分段上傳 (仍在進行中) 之個別部分的總和檢查值，您可以使用 `ListParts`。

> **Important**
>
> 如果您是使用額外檢查總和的分段上傳，則分段部分編號必須使用連續的部分。使用額外的檢查總和時，如果您嘗試使用非連續部分編號完成分段上傳請求，Amazon S3 會產生 HTTP `500 Internal Server Error` 錯誤。

如需如何使用分段物件檢查總和的詳細資訊，請參閱「檢查物件完整性」。

並行分段上傳操作

在分散式開發環境中，您的應用程式有可能同時對相同的物件啟動數項更新。您的應用程式可能使用相同的物件金鑰，啟動數項分段上傳。然後針對這些每一個上傳，應用程式會上傳各組件，並對 Amazon S3 傳送完成上傳要求，以建立物件。當儲存貯體啟用 S3 版本控制之後，完成分段上傳一律會建立新的版本。若是未啟用版本控制的儲存貯體，在分段上傳啟動與完成之間，可能有一些收到的其他要求已先完成。

> **Note**
>
> 在分段上傳啟動與完成之間，可能有一些收到的其他要求已先完成。例如，若在您以某個金鑰啟動分段上傳之後，其他操作卻在您完成前刪除了該金鑰，則完成分段上傳回應可能顯示物件建立成功，但您根本沒看到該物件。

分段上傳與定價

啟動分段上傳之後，Amazon S3 就會保留所有組件，直到您完成或停止上傳為止。在其整個生命週期內，您都要支付此分段上傳及其相關組件的儲存體、頻寬與要求之費用。

這些部份會根據上傳部份時指定的儲存類別收費。例外情況是部份上傳至 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive。進行 PUT 分段上傳至 S3 Glacier Flexible Retrieval 儲存體類別時，當作 S3 Glacier Flexible Retrieval 預備儲存體以 S3 標準儲存體費率計費，直到上傳完成為止。此外，CreateMultipartUpload 和 UploadPart 均以 S3 標準費率計費。只有 CompleteMultipartUpload 請求才會以 S3 Glacier 彈性擷取費率計費。同樣地，PUT 到 S3 Glacier 深層存檔儲存類別的進行中多部分部分會按 S3 標準儲存費率計費為 S3 Glacier 彈性擷取暫存，直到上傳完成為止，只需按 S3 Glacier 深度存檔費率收取 CompleteMultipartUpload請求費用。

若停止分段上傳，Amazon S3 會刪除上傳成品及所有已上傳的組件，您無須再支付它們的費用。無論指定的儲存空間類別為何，刪除不完整的分段上傳都不會收取提前刪除費用。如需定價的詳細資訊，請參閱 Amazon S3 定價。

> **Note**
>
> 為了將儲存費用降至最低，建議您使用 `AbortIncompleteMultipartUpload` 動作，將生命週期規則設定為在指定天數後刪除不完整的分段上傳。如需詳細了解生命週期規則的建立，以刪除不完整分段上傳，請參閱設定儲存貯體生命週期組態，刪除不完整分段上傳。

分段上傳的 API 支援

這些程式庫提供高階抽象概念，讓您可以輕鬆分段上傳物件。但您可視應用程式的需求，直接使用 REST API。《Amazon Simple Storage Service API 參考》中的下列章節說明了分段上傳的 REST API。

如需使用 AWS Lambda 函數的多部分上傳逐步解說，請參閱使用多部分上傳和傳輸加速將大型物件上傳到 Amazon S3。

- 建立分段上傳
- 上傳片段
- 分段上傳 (複製)
- 完成分段上傳
- 中止分段上傳
- 列出組件
- 列出分段上傳

AWS Command Line Interface 支持多部分上傳

中的下列主題 AWS Command Line Interface 說明多部分上載的作業。

- 啟動分段上傳
- 上傳片段
- 分段上傳 (複製)
- 完成分段上傳
- 中止分段上傳
- 列出組件
- 列出分段上傳

AWS SDK 支持多部分上傳

您可以使用 AWS SDK 部分上傳物件。如需 API 動作支援的 AWS SDK 清單，請參閱：

- 建立分段上傳
- 上傳片段

- 分段上傳 (複製)
- 完成分段上傳
- 中止分段上傳
- 列出組件
- 列出分段上傳

分段上傳 API 與許可

您必須要有必要許可，才可使用分段上傳操作。您可以使用存取控制清單 (ACL)、儲存貯體政策或使用者政策，為個人授予執行這些操作的許可。下表列出使用 ACL、儲存貯體政策或使用者政策時，各種分段上傳操作所需的許可。

動作	必要許可
建立分段上傳	您必須有權對物件執行 `s3:PutObject` 動作，才可建立分段上傳。 儲存貯體擁有者可允許其他委託人執行 `s3:PutObject` 動作。
啟動分段上傳	您必須有權對物件執行 `s3:PutObject` 動作，才可啟動分段上傳。 儲存貯體擁有者可允許其他委託人執行 `s3:PutObject` 動作。
啟動者	識別分段上傳啟動者的容器元素。如果初始器是 AWS 帳戶，則此元素會提供與 Owner 元素相同的資訊。若啟動者是 IAM 使用者，此元素會提供使用者 ARN 與顯示名稱。
上傳片段	您必須有權對物件執行 `s3:PutObject` 動作，才可分段上傳。 儲存貯體擁有者必須允許啟動者對物件執行 `s3:PutObject` 動作，啟動者才可分段上傳該物件。
上傳片段 (複製)	您必須有權對物件執行 `s3:PutObject` 動作，才可分段上傳。因為您分段上傳現有物件，所以必須要能對來源物件進行 `s3:GetObject` 。 啟動者若要分段上傳該物件，儲存貯體擁有者必須允許啟動者對物件執行 `s3:PutObject` 動作。
完成分段上傳	您必須有權對物件執行 `s3:PutObject` 動作，才可完成分段上傳。

動作	必要許可
	儲存貯體擁有者必須允許啟動者對物件執行 s3:PutObject 動作，啟動者才可完成該物件的分段上傳。
停止分段上傳	您必須有權對物件執行 s3:AbortMultipartUpload 動作，才可停止分段上傳。 儲存貯體擁有者和分段上傳啟動者預設都可以做為 IAM 和儲存貯體政策的一部分，來執行此動作。如果初始器是 IAM 使用者，也允許該使用者停止該分段上傳。 AWS 帳戶 使用 VPC 端點政策，分段上傳的啟動者不會自動獲得執行 s3:AbortMultipartUpload 動作的許可。 除了這些預設值之外，儲存貯體擁有者可允許其他委託人對物件執行 s3:AbortMultipartUpload 動作。儲存貯體擁有者可拒絕任何委託人執行 s3:AbortMultipartUpload 動作。
列出組件	您必須有權執行 s3:ListMultipartUploadParts 動作，才可列出分段上傳的各組件。 儲存貯體擁有者預設具備許可，可對儲存貯體列出任何分段上傳之組件。分段上傳的啟動者則有列出特定分段上傳組件的許可。如果分段上傳啟動器是 IAM 使用者，則該 IAM 使用者的 AWS 帳戶 控制項也具有列出該上傳部分的權限。 除了這些預設值之外，儲存貯體擁有者可允許其他委託人對物件執行 s3:ListMultipartUploadParts 動作。儲存貯體擁有者也可拒絕任何委託人執行 s3:ListMultipartUploadParts 動作。
列出分段上傳	您必須有權對儲存貯體執行 s3:ListBucketMultipartUploads 動作，才可列出該儲存貯體進行中的分段上傳。 除此預設值外，儲存貯體擁有者可允許其他委託人對儲存貯體執行 s3:ListBucketMultipartUploads 動作。

動作	必要許可
AWS KMS 加密和解密相關權限	若要使用 AWS Key Management Service (AWS KMS) KMS 金鑰以加密執行多部分上傳，請求者必須擁有金鑰`kms:Decrypt` 和`kms:GenerateDataKey` 動作的權限。這些許可是必要的，因為在加密檔案完成分段上傳之前，Amazon S3 必須從部分加密檔案解密並讀取資料。 如需詳細資訊，請參閱 AWS 知識中心中的使用 AWS KMS key透過加密以將大型檔案上傳至 Amazon S3。 如果您的 IAM 使用者或角色與 KMS 金鑰 AWS 帳戶 相同，則您必須擁有金鑰政策的這些權限。若您的 IAM 使用者或角色屬於與 CMK 不同的帳戶，您必須同時在金鑰政策和您的 IAM 使用者或角色上具備這些許可。

如需 ACL 許可與存取原則許可之間關聯性的資訊，請參閱「ACL 許可與存取政策許可的對應」。如需 IAM 使用者、角色與最佳實務的詳細資訊，請參閱《IAM 使用者指南》中的 IAM 身分 (使用者、群組和角色)。

主題

- 設定儲存貯體生命週期組態，以刪除不完整的分段上傳
- 使用分段上傳來上傳物件
- 使用高階 .NET TransferUtility 類別上傳目錄
- 列出分段上傳
- 追蹤分段上傳
- 中止分段上傳
- 使用分段上傳來複製物件
- Amazon S3 分段上傳限制

設定儲存貯體生命週期組態，以刪除不完整的分段上傳

最佳實務做法建議您使用 `AbortIncompleteMultipartUpload` 動作設定生命週期規則，從而將儲存體費用降至最低。如需中止分段上傳的詳細資訊，請參閱 中止分段上傳。

Amazon S3 支援的儲存貯體生命週期規則，可用以指示 Amazon S3 在該過程啟動後指定的天數內，停止尚未完成的分段上傳。當分段上傳未在指定的時間範圍內完成時，其符合中止操作的資

格。Amazon S3 接著中止分段上傳，並刪除與該分段上傳相關聯的部分。此規則適用於現有的分段上傳以及稍後建立的分段上傳。

下列生命週期組態範例指定了一項規則，其會採取 AbortIncompleteMultipartUpload 動作。

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Prefix></Prefix>
        <Status>Enabled</Status>
        <AbortIncompleteMultipartUpload>
          <DaysAfterInitiation>7</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

在此範例中，規則不會針對 Prefix 元素指定一值 (物件金鑰名稱字首)。因此，規則適用於您啟動分段上傳之儲存貯體中的所有物件。任何已啟動且未在七天內完成的分段上傳，都符合中止操作的資格。中止動作對已完成的分段上傳沒有影響。

如需儲存貯體生命週期組態的詳細資訊，請參閱「管理儲存生命週期」。

> **Note**
>
> 若分段上傳在規則指定的天數內完成，就不會套用 AbortIncompleteMultipartUpload 生命週期動作 (也就是說，Amazon S3 不會採取任何動作)。此外，此動作不適用於物件。此生命週期動作不會刪除任何物件。此外，當您移除任何不完整分段上傳時，不會產生 S3 生命週期的提前刪除費用。

使用 S3 主控台

若要自動管理不完整的分段上傳，您可以使用 S3 主控台建立生命週期規則，以在指定天數後，使儲存貯體中未完成的分段上傳位元組過期。下列程序說明如何新增生命週期規則，以在 7 天後刪除未完成的分段上傳。如需新增生命週期規則的詳細資訊，請參閱 在值區上設定生命週期組態。

新增生命週期規則以中止超過 7 天未完成的分段上傳

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇要建立生命週期規則的儲存貯體名稱。

3. 選擇 Management (管理) 標籤，然後選擇 Create lifecycle rule (建立生命週期規則)。
4. 在 Lifecycle rule name (生命週期規則名稱) 中，輸入規則的名稱。

   在儲存貯體內，名稱必須是唯一的。
5. 選擇生命週期規則的範圍：

   - 若要針對具有特定字首的所有物件建立生命週期規則，請選擇 Limit the scope of this rule using one or more filters (使用一或多個篩選條件限制此規則的範圍)，然後在 Prefix (字首) 欄位中輸入字首。
   - 若要針對儲存貯體中的所有物件建立生命週期規則，請選擇 This rule applies to all objects in the bucket (此規則適用於儲存貯體中的所有物件)，然後選擇 I acknowledge that this rule applies to all objects in the bucket (我確認此規則適用於儲存貯體中的所有物件)。
6. 在 Lifecycle rule actions (生命週期規則動作) 下，選取 Delete expired object delete markers or incomplete multipart uploads (刪除過期物件標記或未完成的分段上傳)。
7. 在 Delete expired delete markers or incomplete multipart uploads (刪除過期刪除標記或未完成的分段上傳) 下，選擇 Delete incomplete multipart uploads (刪除未完成的分段上傳)。
8. 在 Number of days (天數) 欄位中，輸入要刪除幾天後未完成的分段上傳 (在此範例中，7 天)。
9. 選擇建立規則。

使用 AWS CLI

以下 `put-bucket-lifecycle-configuration`AWS Command Line Interface (AWS CLI) 命令會新增指定儲存貯體的生命週期組態。若要使用此命令，請以您的資訊取代 *`user input placeholders`*。

```
aws s3api put-bucket-lifecycle-configuration  \
        --bucket example-s3-bucket1  \
        --lifecycle-configuration filename-containing-lifecycle-configuration
```

下列範例說明如何新增生命週期規則，以使用 AWS CLI中止未完成的分段上傳。其中包含範例 JSON 生命週期組態，以中止超過 7 天未完成的分段上傳。

若要在此範例中使用 CLI 命令，請以您的資訊取代 *`user input placeholders`*。

新增生命週期規則以中止未完成的分段上傳

1. 設定 AWS CLI. 如需說明，請參閱使用 AWS CLI 來透過 Amazon S3 進行開發。

2. 將以下範例生命週期組態儲存至檔案 (例如，*lifecycle.json*)。此範例組態指定了空的字首，因此其會套用至該儲存貯體中的所有物件。若要將組態限制為物件子集，您可以指定字首。

```
{
    "Rules": [
        {
            "ID": "Test Rule",
            "Status": "Enabled",
            "Filter": {
                "Prefix": ""
            },
            "AbortIncompleteMultipartUpload": {
                "DaysAfterInitiation": 7
            }
        }
    ]
}
```

3. 執行下列 CLI 命令，以在您的儲存貯體上設定此生命週期組態。

```
aws s3api put-bucket-lifecycle-configuration   \
--bucket example-s3-bucket1  \
--lifecycle-configuration file://lifecycle.json
```

4. 若要驗證是否已在您的儲存貯體上設定生命週期組態，請使用下列 `get-bucket-lifecycle` 命令擷取生命週期組態。

```
aws s3api get-bucket-lifecycle  \
--bucket example-s3-bucket1
```

5. 若要刪除生命週期組態，請如下使用 `delete-bucket-lifecycle` 命令。

```
aws s3api delete-bucket-lifecycle \
--bucket example-s3-bucket1
```

使用分段上傳來上傳物件

您可以使用分段上傳，以程式設計方式將單一物件上傳到 Amazon S3。

如需詳細資訊，請參閱下列區段。

使用 AWS 軟體開發套件 (高階 API)

某些 AWS SDK 公開了一個高級 API，該 API 通過將完成多部分上傳所需的不同 API 操作合併到單個操作中，從而簡化了多部分上傳。如需詳細資訊，請參閱 使用分段上傳來上傳和複製物件。

如果您需要暫停並繼續分段上傳、在上傳期間變更零件大小，或者不事先知道資料大小，請使用低階 API 方法。用於分段上傳的低級 API 方法提供了其他功能，如需詳細資訊，請參閱。使用 AWS 軟體開發套件 (低階 API)

Java

若要上傳大型檔案，請使用TransferManager類別。這種高級 API 操作可以從文件或流上傳數據。您也可以設定進階選項 (例如分段上傳的組件大小)，或同時上傳多個組件時，所要使用的執行緒數。您也可以選擇是否要設定物件屬性、儲存體方案或 存取控制清單 (ACL)。您可以使用 PutObjectRequest 及 TransferManagerConfiguration 類別設定這些進階選項。

如有可能，TransferManager 會嘗試使用多執行緒一次上傳多個組件。在處理大量內容及高頻寬時，輸送量可能會顯著地增加。

除了檔案上傳功能外，TransferManager 類別也可讓您停止進行中的分段上傳。啟動上傳後，在尚未完成或停止之前，均視為進行中。TransferManager 停止在指定的儲存貯體中，某時間點前開始的所有執行分段上傳作業。

> **Note**
>
> 在您使用資料來源的串流時，TransferManager 類別不會執行並行上傳。

以下範例說明如何使用高階分段上傳 Java API (TransferManager 類別) 作業，上傳物件。如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;
```

```
import java.io.File;

public class HighLevelMultipartUpload {

    public static void main(String[] args) throws Exception {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Object key ***";
        String filePath = "*** Path for file to upload ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();
            TransferManager tm = TransferManagerBuilder.standard()
                    .withS3Client(s3Client)
                    .build();

            // TransferManager processes all transfers asynchronously,
            // so this call returns immediately.
            Upload upload = tm.upload(bucketName, keyName, new File(filePath));
            System.out.println("Object upload started");

            // Optionally, wait for the upload to finish before continuing.
            upload.waitForCompletion();
            System.out.println("Object upload complete");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

若要上傳檔案到 S3 儲存貯體，可使用 `TransferUtility` 類別。當從檔案上傳資料時，您必須提供物件的金鑰名稱。如果不提供的話，API 使用檔案名稱作為金鑰。當從串流上傳資料時，您必須提供物件的金鑰名稱。

若要設定進階上傳選項 (如片段大小、同時上傳多個片段時的執行緒數、中繼資料、儲存方案或 ACL)，請使用 `TransferUtilityUploadRequest` 類別。

> **Note**
>
> 在您使用資料來源的串流時，`TransferUtility` 類別不會執行並行上傳。

下列 C# 範例會將檔案分成多個部分 (組件)，上傳至 Amazon S3 儲存貯體。此顯示如何使用不同的 `TransferUtility.Upload` 多載上傳檔案。每個後續的呼叫都會上傳取代前一個上傳。如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadFileMPUHighLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        private const string keyName = "*** provide a name for the uploaded object
 ***";
        private const string filePath = "*** provide the full path name of the file
 to upload ***";
       // Specify your bucket region (an example region is shown).
       private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
       private static IAmazonS3 s3Client;

       public static void Main()
       {
           s3Client = new AmazonS3Client(bucketRegion);
```

```
            UploadFileAsync().Wait();
        }

        private static async Task UploadFileAsync()
        {
            try
            {
                var fileTransferUtility =
                    new TransferUtility(s3Client);

                // Option 1. Upload a file. The file name is used as the object key
name.
                await fileTransferUtility.UploadAsync(filePath, bucketName);
                Console.WriteLine("Upload 1 completed");

                // Option 2. Specify object key name explicitly.
                await fileTransferUtility.UploadAsync(filePath, bucketName,
keyName);
                Console.WriteLine("Upload 2 completed");

                // Option 3. Upload data from a type of System.IO.Stream.
                using (var fileToUpload =
                    new FileStream(filePath, FileMode.Open, FileAccess.Read))
                {
                    await fileTransferUtility.UploadAsync(fileToUpload,
                                               bucketName, keyName);
                }
                Console.WriteLine("Upload 3 completed");

                // Option 4. Specify advanced settings.
                var fileTransferUtilityRequest = new TransferUtilityUploadRequest
                {
                    BucketName = bucketName,
                    FilePath = filePath,
                    StorageClass = S3StorageClass.StandardInfrequentAccess,
                    PartSize = 6291456, // 6 MB.
                    Key = keyName,
                    CannedACL = S3CannedACL.PublicRead
                };
                fileTransferUtilityRequest.Metadata.Add("param1", "Value1");
                fileTransferUtilityRequest.Metadata.Add("param2", "Value2");

                await fileTransferUtility.UploadAsync(fileTransferUtilityRequest);
                Console.WriteLine("Upload 4 completed");
```

```
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }

        }
    }
}
```

JavaScript

Example

上傳大型檔案。

```
import {
  CreateMultipartUploadCommand,
  UploadPartCommand,
  CompleteMultipartUploadCommand,
  AbortMultipartUploadCommand,
  S3Client,
} from "@aws-sdk/client-s3";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

export const main = async () => {
  const s3Client = new S3Client({});
  const bucketName = "test-bucket";
  const key = "multipart.txt";
  const str = createString();
  const buffer = Buffer.from(str, "utf8");

  let uploadId;
```

```
try {
  const multipartUpload = await s3Client.send(
    new CreateMultipartUploadCommand({
      Bucket: bucketName,
      Key: key,
    }),
  );

  uploadId = multipartUpload.UploadId;

  const uploadPromises = [];
  // Multipart uploads require a minimum size of 5 MB per part.
  const partSize = Math.ceil(buffer.length / 5);

  // Upload each part.
  for (let i = 0; i < 5; i++) {
    const start = i * partSize;
    const end = start + partSize;
    uploadPromises.push(
      s3Client
        .send(
          new UploadPartCommand({
            Bucket: bucketName,
            Key: key,
            UploadId: uploadId,
            Body: buffer.subarray(start, end),
            PartNumber: i + 1,
          }),
        )
        .then((d) => {
          console.log("Part", i + 1, "uploaded");
          return d;
        }),
    );
  }

  const uploadResults = await Promise.all(uploadPromises);

  return await s3Client.send(
    new CompleteMultipartUploadCommand({
      Bucket: bucketName,
      Key: key,
      UploadId: uploadId,
```

```
        MultipartUpload: {
          Parts: uploadResults.map(({ ETag }, i) => ({
            ETag,
            PartNumber: i + 1,
          })),
        },
      }),
    );

    // Verify the output by downloading the file from the Amazon Simple Storage
 Service (Amazon S3) console.
    // Because the output is a 25 MB string, text editors might struggle to open the
 file.
  } catch (err) {
    console.error(err);

    if (uploadId) {
      const abortCommand = new AbortMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
      });

      await s3Client.send(abortCommand);
    }
  }
};
```

Example

下載大型檔案。

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream } from "fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });
```

```
  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: parseInt(start),
    end: parseInt(end),
    length: parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

// When downloading a large file, you might want to break it down into
// smaller pieces. Amazon S3 accepts a Range header to specify the start
// and end of the byte range to be downloaded.
const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    console.log(`Downloading bytes ${nextRange.start} to ${nextRange.end}`);

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
```

```
};

export const main = async () => {
  await downloadInChunks({
    bucket: "my-cool-bucket",
    key: "my-cool-object.txt",
  });
};
```

Go

Example

使用上傳管理員將資料分成多個部分，並同時上傳它們來上傳大型物件。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}
```

```
// UploadLargeObject uses an upload manager to upload data to an object in a bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(bucketName string, objectKey string,
 largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
```

```
    bucketName, objectKey, err)
  }

  return err
}
```

Example

使用下載管理員分段取得資料並同時下載它們來下載大型物件。

```
// DownloadLargeObject uses a download manager to download an object from a bucket.
// The download manager gets the data in parts and writes them to a buffer until all
 of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(bucketName string, objectKey string)
 ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader) {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(context.TODO(), buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

PHP

本主題說明如何使用來自多部分檔案上傳 AWS SDK for PHP 的高階`Aws\S3\Model\MultipartUpload\UploadBuilder`類別。有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

下列 PHP 範例會將檔案上傳至 Amazon S3 儲存貯體。本範例會示範如何設定 `MultipartUploader` 物件的參數。

```
 require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Prepare the upload parameters.
$uploader = new MultipartUploader($s3, '/path/to/large/file.zip', [
    'bucket' => $bucket,
    'key'    => $keyname
]);

// Perform the upload.
try {
    $result = $uploader->upload();
    echo "Upload complete: {$result['ObjectURL']}" . PHP_EOL;
} catch (MultipartUploadException $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Python

下列範例會使用高階分段上傳 Python API (`TransferManager` 類別) 載入物件。

```
import sys
import threading

import boto3
from boto3.s3.transfer import TransferConfig


MB = 1024 * 1024
s3 = boto3.resource("s3")
```

```
class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
```

```
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.
```

```
    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
```

```
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
```

```
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

使用 AWS 軟體開發套件 (低階 API)

該 AWS 開發套件公開了一個與用於多部分上傳的 Amazon S3 REST API 非常類似的低級 API (請參閱。使用分段上傳來上傳和複製物件 如果您需要暫停和繼續分段上傳、在上傳期間變更零件大小，或是事先不知道上傳資料的大小，請使用低階 API。 如果您沒有這些需求，請使用高階 API (請參閱使用 AWS 軟體開發套件 (高階 API))。

Java

以下範例示範如何使用低階 Java 類別上傳檔案。操作步驟如下：

- 使用 AmazonS3Client.initiateMultipartUpload() 方法，經過 InitiateMultipartUploadRequest 物件，開始執行分段上傳。
- 儲存 AmazonS3Client.initiateMultipartUpload() 方法所傳回的上傳 ID。您所提供的上傳 ID，用於後續的每項分段上傳操作。
- 將物件的部分上傳。請為每一部分，呼叫 AmazonS3Client.uploadPart() 方法。您可以在 UploadPartRequest 物件中提供部分資訊。
- 對於每一個部分，將 ETag 儲存在清單 AmazonS3Client.uploadPart() 方法的回應中。請您使用 ETag 值完成分段上傳。
- 呼叫 AmazonS3Client.completeMultipartUpload() 方法，完成分段上傳。

Example

如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class LowLevelMultipartUpload {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String filePath = "*** Path to file to upload ***";

        File file = new File(filePath);
        long contentLength = file.length();
        long partSize = 5 * 1024 * 1024; // Set part size to 5 MB.

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Create a list of ETag objects. You retrieve ETags for each object
 part
            // uploaded,
            // then, after each individual part has been uploaded, pass the list of
 ETags to
            // the request to complete the upload.
            List<PartETag> partETags = new ArrayList<PartETag>();
```

```
            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
InitiateMultipartUploadRequest(bucketName, keyName);
            InitiateMultipartUploadResult initResponse =
s3Client.initiateMultipartUpload(initRequest);

            // Upload the file parts.
            long filePosition = 0;
            for (int i = 1; filePosition < contentLength; i++) {
                // Because the last part could be less than 5 MB, adjust the part
size as
                // needed.
                partSize = Math.min(partSize, (contentLength - filePosition));

                // Create the request to upload a part.
                UploadPartRequest uploadRequest = new UploadPartRequest()
                        .withBucketName(bucketName)
                        .withKey(keyName)
                        .withUploadId(initResponse.getUploadId())
                        .withPartNumber(i)
                        .withFileOffset(filePosition)
                        .withFile(file)
                        .withPartSize(partSize);

                // Upload the part and add the response's ETag to our list.
                UploadPartResult uploadResult = s3Client.uploadPart(uploadRequest);
                partETags.add(uploadResult.getPartETag());

                filePosition += partSize;
            }

            // Complete the multipart upload.
            CompleteMultipartUploadRequest compRequest = new
CompleteMultipartUploadRequest(bucketName, keyName,
                    initResponse.getUploadId(), partETags);
            s3Client.completeMultipartUpload(compRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
```

```
        }
    }
}
```

.NET

下列 C# 範例示範如何使用低階 AWS SDK for .NET 多部分上傳 API 將檔案上傳至 S3 儲存貯體。如需 Amazon S3 分段上傳的資訊，請參閱 使用分段上傳來上傳和複製物件。

> **Note**
>
> 當您使用 AWS SDK for .NET API 上傳大型物件時，可能會在將資料寫入要求串流時發生逾時。您可以使用 `UploadPartRequest` 設定明確逾時。

下列 C# 範例會使用低階分段上傳 API，將檔案上傳至 S3 儲存貯體。如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadFileMPULowLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        private const string keyName = "*** provide a name for the uploaded object
 ***";
        private const string filePath = "*** provide the full path name of the file
 to upload ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
```

```
        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Uploading an object");
            UploadObjectAsync().Wait();
        }

        private static async Task UploadObjectAsync()
        {
            // Create list to store upload part responses.
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

            // Setup information required to initiate the multipart upload.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = bucketName,
                Key = keyName
            };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // Upload parts.
            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                Console.WriteLine("Uploading parts");

                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                        {
                            BucketName = bucketName,
                            Key = keyName,
                            UploadId = initResponse.UploadId,
                            PartNumber = i,
                            PartSize = partSize,
                            FilePosition = filePosition,
```

```
                        FilePath = filePath
                    };

                // Track upload progress.
                uploadRequest.StreamTransferProgress +=
                    new
EventHandler<StreamTransferProgressArgs>(UploadPartProgressEventCallback);

                // Upload a part and add the response to our list.
                uploadResponses.Add(await
s3Client.UploadPartAsync(uploadRequest));

                filePosition += partSize;
            }

            // Setup to complete the upload.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    UploadId = initResponse.UploadId
                 };
            completeRequest.AddPartETags(uploadResponses);

            // Complete the upload.
            CompleteMultipartUploadResponse completeUploadResponse =
                await s3Client.CompleteMultipartUploadAsync(completeRequest);
        }
        catch (Exception exception)
        {
            Console.WriteLine("An AmazonS3Exception was thrown: { 0}",
exception.Message);

            // Abort the upload.
            AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
            {
                BucketName = bucketName,
                Key = keyName,
                UploadId = initResponse.UploadId
            };
           await s3Client.AbortMultipartUploadAsync(abortMPURequest);
        }
```

```
        }
        public static void UploadPartProgressEventCallback(object sender,
 StreamTransferProgressArgs e)
        {
            // Process event.
            Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
        }
    }
}
```

PHP

本主題說明如何使用版本 3 的低階uploadPart方法， AWS SDK for PHP 以多個部分上傳檔案。有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

下列 PHP 範例會使用低階 PHP API 分段上傳，將檔案上傳至 Amazon S3 儲存貯體。

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
$filename = '*** Path to and Name of the File to Upload ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

$result = $s3->createMultipartUpload([
    'Bucket'       => $bucket,
    'Key'          => $keyname,
    'StorageClass' => 'REDUCED_REDUNDANCY',
    'Metadata'     => [
        'param1' => 'value 1',
        'param2' => 'value 2',
        'param3' => 'value 3'
    ]
]);
$uploadId = $result['UploadId'];

// Upload the file in parts.
```

```
try {
    $file = fopen($filename, 'r');
    $partNumber = 1;
    while (!feof($file)) {
        $result = $s3->uploadPart([
            'Bucket'     => $bucket,
            'Key'        => $keyname,
            'UploadId'   => $uploadId,
            'PartNumber' => $partNumber,
            'Body'       => fread($file, 5 * 1024 * 1024),
        ]);
        $parts['Parts'][$partNumber] = [
            'PartNumber' => $partNumber,
            'ETag' => $result['ETag'],
        ];
        $partNumber++;

        echo "Uploading part $partNumber of $filename." . PHP_EOL;
    }
    fclose($file);
} catch (S3Exception $e) {
    $result = $s3->abortMultipartUpload([
        'Bucket'   => $bucket,
        'Key'      => $keyname,
        'UploadId' => $uploadId
    ]);

    echo "Upload of $filename failed." . PHP_EOL;
}

// Complete the multipart upload.
$result = $s3->completeMultipartUpload([
    'Bucket'   => $bucket,
    'Key'      => $keyname,
    'UploadId' => $uploadId,
    'MultipartUpload'    => $parts,
]);
$url = $result['Location'];

echo "Uploaded $filename to $url." . PHP_EOL;
```

使用 AWS SDK for Ruby

第 3 AWS SDK for Ruby 版以兩種方式支援 Amazon S3 分段上傳。第一個選項是可以使用受管檔案上傳。如需詳細資訊，請參閱 AWS 開發人員部落格中的將檔案上傳至 Amazon S3。受管檔案上傳是上傳檔案至儲存貯體的推薦方式。他們具有以下優點：

- 管理大於 15 MB 的物件之分段上傳。
- 以二進位模式正確開啟檔案，避免編碼問題。
- 使用多執行緒平行分段上傳大型物件。

或者，您可以直接使用以下分段上傳用戶端操作。

- create_multipart_upload – 啟動分段上傳並傳回上傳 ID。
- upload_part – 以分段上傳上傳片段。
- upload_part_copy – 複製現有物件資料作為資料來源，以上傳片段。
- complete_multipart_upload – 組合先前已上傳的片段，以完成分段上傳。
- abort_multipart_upload – 停止分段上傳。

使用 REST API

《Amazon Simple Storage Service API 參考》中的下列章節說明了分段上傳的 REST API。

- 啟動分段上傳
- 上傳片段
- 完成分段上傳
- 停止分段上傳
- 列出組件
- 列出分段上傳

使用 AWS CLI

AWS Command Line Interface (AWS CLI) 中的下列各節說明多部分上傳的作業。

- 啟動分段上傳
- 上傳片段
- 分段上傳 (複製)

- 完成分段上傳
- 中止分段上傳
- 列出組件
- 列出分段上傳

您可以使用這些 API 來提出自己的 REST 請求，也可以使用其中一個 AWS SDK。如需 REST API 的詳細資訊，請參閱 使用 REST API。如需 SDK 的詳細資訊，請參閱「使用分段上傳來上傳物件」。

使用高階 .NET TransferUtility 類別上傳目錄

您可以使用 `TransferUtility` 類別，上傳整個目錄。根據預設，API 只會上傳指定目錄之根目錄中的檔案。但您可以指定以遞迴方式，上傳所有子目錄中的檔案。

也可以根據一些篩選條件，指定篩選運算式，來選取指定目錄中的檔案。例如，若只要從目錄上傳 .pdf 檔案，可以指定 `"*.pdf"` 的篩選運算式。

從目錄上傳檔案時，無法指定結果物件的金鑰名稱。Amazon S3 使用原始檔案路徑，建構金鑰名稱。例如，假設您有一個目錄為 `c:\myfolder`，其結構如下：

Example

```
C:\myfolder
      \a.txt
      \b.pdf
      \media\
             An.mp3
```

當您上傳此目錄時，Amazon S3 會使用下列金鑰名稱：

Example

```
a.txt
b.pdf
media/An.mp3
```

Example

下列 C# 範例會將目錄上傳至 Amazon S3 儲存貯體。此顯示如何使用不同的 `TransferUtility.UploadDirectory` 多載上傳目錄。每個後續的呼叫都會上傳取代前一個上傳。

如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadDirMPUHighLevelAPITest
    {
        private const string existingBucketName = "*** bucket name ***";
        private const string directoryPath = @"*** directory path ***";
        // The example uploads only .txt files.
        private const string wildCard = "*.txt";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            UploadDirAsync().Wait();
        }

        private static async Task UploadDirAsync()
        {
            try
            {
                var directoryTransferUtility =
                    new TransferUtility(s3Client);

                // 1. Upload a directory.
                await directoryTransferUtility.UploadDirectoryAsync(directoryPath,
                    existingBucketName);
                Console.WriteLine("Upload statement 1 completed");

                // 2. Upload only the .txt files from a directory
                //    and search recursively.
                await directoryTransferUtility.UploadDirectoryAsync(
                                                directoryPath,
```

```
                                            existingBucketName,
                                            wildCard,
                                            SearchOption.AllDirectories);
                Console.WriteLine("Upload statement 2 completed");

                // 3. The same as Step 2 and some optional configuration.
                //    Search recursively for .txt files to upload.
                var request = new TransferUtilityUploadDirectoryRequest
                {
                    BucketName = existingBucketName,
                    Directory = directoryPath,
                    SearchOption = SearchOption.AllDirectories,
                    SearchPattern = wildCard
                };

                await directoryTransferUtility.UploadDirectoryAsync(request);
                Console.WriteLine("Upload statement 3 completed");
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an object",
 e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Unknown encountered on server. Message:'{0}' when writing an
 object", e.Message);
            }
        }
    }
}
```

列出分段上傳

您可以使用開 AWS 發套件 (低階 API) 擷取 Amazon S3 中進行中的多部分上傳清單。

使用 AWS SDK 列出多部分上傳（低級 API）

Java

下列作業將引導您使用低階 Java 類別，列出儲存貯體上所有進行中的分段上傳。

低階 API 分段上傳列出程序

1	建立 `ListMultipartUploadsRequest` 類別的執行個體，並提供儲存貯體名稱。
2	執行 `AmazonS3Client.listMultipartUploads` 方法。此方法會傳回 `MultipartUploadListing` 類別的執行個體，提供您進行中分段上傳的相關資訊。

下列 Java 程式碼範例示範上述工作。

Example

```
ListMultipartUploadsRequest allMultpartUploadsRequest =
     new ListMultipartUploadsRequest(existingBucketName);
MultipartUploadListing multipartUploadListing =
     s3Client.listMultipartUploads(allMultpartUploadsRequest);
```

.NET

若要列出特定儲存貯體上所有進行中的分段上傳，請使用 AWS SDK for .NET 低階分段上傳 API 的 `ListMultipartUploadsRequest` 類別。`AmazonS3Client.ListMultipartUploads` 方法會傳回 `ListMultipartUploadsResponse` 類別的執行個體，提供進行中分段上傳的相關資訊。

進行中的分段上傳是已使用起始分段上傳請求啟動，但尚未完成或停止的分段上傳。如需 Amazon S3 分段上傳的詳細資訊，請參閱 使用分段上傳來上傳和複製物件。

下列 C# 範例說明如何使用列出值 AWS SDK for .NET 區上所有進行中的多部分上傳。如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
ListMultipartUploadsRequest request = new ListMultipartUploadsRequest
{
  BucketName = bucketName // Bucket receiving the uploads.
};

ListMultipartUploadsResponse response = await
 AmazonS3Client.ListMultipartUploadsAsync(request);
```

PHP

本主題說明如何使用第 3 版中的低階 API 類別，列出值區上 AWS SDK for PHP 所有進行中的多部分上傳作業。有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

下列 PHP 範例示範如何列出儲存貯體上所有進行中的分段上傳。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Retrieve a list of the current multipart uploads.
$result = $s3->listMultipartUploads([
    'Bucket' => $bucket
]);

// Write the list of uploads to the page.
print_r($result->toArray());
```

使用 REST API 列出分段上傳

《Amazon Simple Storage Service API 參考》中的下列章節說明了列出分段上傳的 REST API：

- ListParts列出特定分段上載的已上傳零件。
- ListMultipartUploads列表正在進行的多部分上傳。

使用列出多部分上傳 AWS CLI

中的下列各節 AWS Command Line Interface 說明列出分段上傳的作業。

- list-parts – 列出特定分段上傳的已上傳部分。
- list-multipart-uploads列表正在進行的多部分上傳。

追蹤分段上傳

高階分段上傳 API 提供監聽介面 ProgressListener，在上傳物件至 Amazon S3 時，追蹤上傳進度。進度事件定期發生，並會通知接聽程式已傳輸的位元組數。

Java

Example

```
TransferManager tm = new TransferManager(new ProfileCredentialsProvider());

PutObjectRequest request = new PutObjectRequest(
    existingBucketName, keyName, new File(filePath));

// Subscribe to the event and provide event handler.
request.setProgressListener(new ProgressListener() {
   public void progressChanged(ProgressEvent event) {
    System.out.println("Transferred bytes: " +
      event.getBytesTransfered());
            }
});
```

Example

下列 Java 程式碼會上傳檔案，並使用 ProgressListener 追蹤上傳進度。如需如何建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import java.io.File;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.Upload;

public class TrackMPUProgressUsingHighLevelAPI {

    public static void main(String[] args) throws Exception {
        String existingBucketName = "*** Provide bucket name ***";
        String keyName            = "*** Provide object key ***";
        String filePath           = "*** file to upload ***";
```

```
        TransferManager tm = new TransferManager(new ProfileCredentialsProvider());


        // For more advanced uploads, you can create a request object
        // and supply additional request parameters (ex: progress listeners,
        // canned ACLs, etc.)
        PutObjectRequest request = new PutObjectRequest(
          existingBucketName, keyName, new File(filePath));

        // You can ask the upload for its progress, or you can
        // add a ProgressListener to your request to receive notifications
        // when bytes are transferred.
        request.setGeneralProgressListener(new ProgressListener() {
   @Override
   public void progressChanged(ProgressEvent progressEvent) {
    System.out.println("Transferred bytes: " +
      progressEvent.getBytesTransferred());
   }
  });

        // TransferManager processes all transfers asynchronously,
        // so this call will return immediately.
        Upload upload = tm.upload(request);

        try {
         // You can block and wait for the upload to finish
         upload.waitForCompletion();
        } catch (AmazonClientException amazonClientException) {
         System.out.println("Unable to upload file, upload aborted.");
         amazonClientException.printStackTrace();
        }
    }
}
```

.NET

下列 C# 範例會示範將檔案上傳至 S3 儲存貯體，透過使用 `TransferUtility` 類別，和追蹤上傳作業進度。如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
```

```
using Amazon.S3.Transfer;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TrackMPUUsingHighLevelAPITest
    {
        private const string bucketName = "*** provide the bucket name ***";
        private const string keyName = "*** provide the name for the uploaded object
 ***";
        private const string filePath = " *** provide the full path name of the file
 to upload **";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;


        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            TrackMPUAsync().Wait();
        }

        private static async Task TrackMPUAsync()
        {
            try
            {
                var fileTransferUtility = new TransferUtility(s3Client);

                // Use TransferUtilityUploadRequest to configure options.
                // In this example we subscribe to an event.
                var uploadRequest =
                    new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        FilePath = filePath,
                        Key = keyName
                    };

                uploadRequest.UploadProgressEvent +=
                    new EventHandler<UploadProgressArgs>
                        (uploadRequest_UploadPartProgressEvent);
```

```
                await fileTransferUtility.UploadAsync(uploadRequest);
                Console.WriteLine("Upload completed");
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }

        static void uploadRequest_UploadPartProgressEvent(object sender,
 UploadProgressArgs e)
        {
            // Process event.
            Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
        }
    }
}
```

中止分段上傳

啟動分段上傳之後，開始上傳組件。Amazon S3 會存放這些組件，但只會在您上傳完所有組件，並傳送 `successful` 要求以完成分段上傳之後，才從這些組件建立物件 (您應確認完成分段上傳要求是否成功)。收到完成分段上傳的要求時，Amazon S3 就會組合這些組件來建立物件。若未能成功傳送完成分段上傳要求，Amazon S3 就不會組合這些組件，也不會建立任何物件。

系統會向您收取與已上傳部分相關的所有儲存空間費用。如需詳細資訊，請參閱 分段上傳與定價。因此，請務必完成分段上傳以建立物件，或者停止分段上傳，以刪除任何已上傳的部分。

您可以使用 AWS Command Line Interface (AWS CLI)、REST API 或 AWS 開發套件停止在 Amazon S3 中進行中的多部分上傳。您也可以使用儲存貯體生命週期組態停止未完成的分段上傳。

使用 AWS 軟體開發套件 (高階 API)

Java

此 TransferManager 類別提供停止進行中分段上傳的 abortMultipartUploads 方法。啟動上傳後，在尚未完成或停止之前，均視為進行中。您可以提供 Date 值，讓此 API 停止所有在指定之 Date 前對該儲存貯體啟動，並且仍在進行中的分段上傳。

下列工作將引導您使用高階 Java 類別停止分段上傳。

高階 API 的分段上傳停止程序

1	建立 TransferManager 類別的執行個體。
2	傳遞儲存貯體名稱與 Date 值，以執行 TransferManager.abortMultipartUploads 方法。

下列 Java 程式碼會停止一週前，對特定儲存貯體啟動，並且正在進行大的所有分段上傳。如需如何建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import java.util.Date;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.transfer.TransferManager;

public class AbortMPUUsingHighLevelAPI {

    public static void main(String[] args) throws Exception {
        String existingBucketName = "*** Provide existing bucket name ***";

        TransferManager tm = new TransferManager(new ProfileCredentialsProvider());


        int sevenDays = 1000 * 60 * 60 * 24 * 7;
  Date oneWeekAgo = new Date(System.currentTimeMillis() - sevenDays);

        try {
         tm.abortMultipartUploads(existingBucketName, oneWeekAgo);
        } catch (AmazonClientException amazonClientException) {
         System.out.println("Unable to upload file, upload was aborted.");
```

```
            amazonClientException.printStackTrace();
        }
    }
}
```

> **Note**
>
> 您也可以停止特定的分段上傳。如需詳細資訊，請參閱 使用 AWS 軟體開發套件 (低階 API)。

.NET

下列 C# 範例會示範停止一週前，於指定儲存貯體上開始且正在進行中的所有分段上傳。如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class AbortMPUUsingHighLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            AbortMPUAsync().Wait();
        }

        private static async Task AbortMPUAsync()
        {
            try
            {
```

```
                var transferUtility = new TransferUtility(s3Client);

                // Abort all in-progress uploads initiated before the specified
 date.
                await transferUtility.AbortMultipartUploadsAsync(
                    bucketName, DateTime.Now.AddDays(-7));
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

> **Note**
>
> 您也可以停止特定的分段上傳。如需詳細資訊，請參閱 使用 AWS 軟體開發套件 (低階 API)。

使用 AWS 軟體開發套件 (低階 API)

您可呼叫 `AmazonS3.abortMultipartUpload` 方法，停止進行中的分段上傳。此方法會刪除已上傳至 Amazon S3 的所有部分，並釋出資源。您必須提供上傳 ID、儲存貯體名稱與金鑰名稱。下列 Java 程式碼範例示範如何停止進行中的分段上傳。

若要停止分段上傳，您要提供用於上傳的上傳 ID、儲存貯體和金鑰名稱。在停止分段上傳之後，即無法使用該上傳 ID 上傳其他部分。如需 Amazon S3 分段上傳的詳細資訊，請參閱 使用分段上傳來上傳和複製物件。

Java

下列 Java 程式碼範例會停止進行中的分段上傳。

Example

```
InitiateMultipartUploadRequest initRequest =
    new InitiateMultipartUploadRequest(existingBucketName, keyName);
InitiateMultipartUploadResult initResponse =
                s3Client.initiateMultipartUpload(initRequest);

AmazonS3 s3Client = new AmazonS3Client(new ProfileCredentialsProvider());
s3Client.abortMultipartUpload(new AbortMultipartUploadRequest(
            existingBucketName, keyName, initResponse.getUploadId()));
```

> **Note**
>
> 除了特定的分段上傳之外，您也可以停止於特定時間之前啟動且仍在進行中的所有分段上傳。此清除操作有助於停止您已啟動但未完成或停止的舊分段上傳。如需詳細資訊，請參閱使用 AWS 軟體開發套件 (高階 API)。

.NET

下列 C# 範例示範如何停止分段上傳。如需包含下列程式碼的完整 C# 範例，請參閱 使用 AWS 軟體開發套件 (低階 API)。

```
AbortMultipartUploadRequest abortMPURequest = new AbortMultipartUploadRequest
{
    BucketName = existingBucketName,
    Key = keyName,
    UploadId = initResponse.UploadId
};
await AmazonS3Client.AbortMultipartUploadAsync(abortMPURequest);
```

您也可以中止在特定時間之前啟動的所有進行中分段上傳。此清除操作有助於中止未完成或已中止的分段上傳。如需詳細資訊，請參閱 使用 AWS 軟體開發套件 (高階 API)。

PHP

此範例顯示如何使用第 3 版中的類別中止 AWS SDK for PHP 進行中的多部分上傳。有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。範例中使用 `abortMultipartUpload()` 方法。

有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
$uploadId = '*** Upload ID of upload to Abort ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Abort the multipart upload.
$s3->abortMultipartUpload([
    'Bucket'   => $bucket,
    'Key'      => $keyname,
    'UploadId' => $uploadId,
]);
```

使用 REST API

如需有關使用 REST API 停止多部分上傳的詳細資訊，請參閱 Amazon 簡單儲存服務 API 參考AbortMultipartUpload中的。

使用 AWS CLI

如需有關使用停止多部分上傳 AWS CLI 的詳細資訊，請參閱《AWS CLI 命令參考》abort-multipart-upload中的〈〉。

使用分段上傳來複製物件

本節中的範例示範如何使用分段上傳 API 來複製大於 5 GB 的物件。您可以透過單一操作來複製小於 5 GB 的物件。如需詳細資訊，請參閱 複製、移動和重新命名物件。

使用 AWS 軟體開發套件

若要使用低階 API 複製物件，請執行以下步驟：

- 呼叫 `AmazonS3Client.initiateMultipartUpload()` 方法以啟動分段上傳。

- 儲存 AmazonS3Client.initiateMultipartUpload() 方法傳回之回應物件中的上傳 ID。您所提供的上傳 ID，用於每項分段上傳操作。
- 複製所有的部分。為了每一個您需要複製的部分，建立一個新的 CopyPartRequest 類別執行個體。提供部分資訊，包含來源和目的地儲存貯體名稱、來源和目標物件金鑰、上傳 ID、部分的第一和最後的位元組位置與部分編號。
- 儲存 AmazonS3Client.copyPart() 方法呼叫的回應。每一個回應包含 ETag 數值與上傳部分的編號。您需要此資訊才成完成分段上傳。
- 呼叫 AmazonS3Client.completeMultipartUpload() 方法完成複製操作。

Java

Example

以下範例示範如何使用 Amazon S3 低階 Java API 執行分段複製。如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class LowLevelMultipartCopy {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String sourceBucketName = "*** Source bucket name ***";
        String sourceObjectKey = "*** Source object key ***";
        String destBucketName = "*** Target bucket name ***";
        String destObjectKey = "*** Target object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
```

```
                    .withRegion(clientRegion)
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
InitiateMultipartUploadRequest(destBucketName,
                    destObjectKey);
            InitiateMultipartUploadResult initResult =
s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest(sourceBucketName, sourceObjectKey);
            ObjectMetadata metadataResult =
s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();

            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make
sure
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize -
1);

                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(sourceBucketName)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(destBucketName)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }
```

```
            // Complete the upload request to concatenate all uploaded parts and
 make the
            // copied object available.
            CompleteMultipartUploadRequest completeRequest = new
 CompleteMultipartUploadRequest(
                    destBucketName,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
        return etags;
    }
}
```

.NET

下列 C# 範例示範如何使用將大於 5 GB 的 AWS SDK for .NET Amazon S3 物件從一個來源位置複製到另一個來源位置，例如從一個儲存貯體複製到另一個儲存貯體。若要複製小於 5 GB 的物件，請使用「使用 AWS 軟體開發套件」中，所描述的單一操作複製程序。如需 Amazon S3 分段上傳的詳細資訊，請參閱 使用分段上傳來上傳和複製物件。

此範例說明如何使用 AWS SDK for .NET 多部分上傳 API，將大於 5 GB 的 Amazon S3 物件從一個 S3 儲存貯體複製到另一個儲存貯體。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CopyObjectUsingMPUapiTest
    {
        private const string sourceBucket = "*** provide the name of the bucket with
 source object ***";
        private const string targetBucket = "*** provide the name of the bucket to
 copy the object to ***";
        private const string sourceObjectKey = "*** provide the name of object to
 copy ***";
        private const string targetObjectKey = "*** provide the name of the object
 copy ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Copying an object");
            MPUCopyObjectAsync().Wait();
        }
        private static async Task MPUCopyObjectAsync()
        {
            // Create a list to store the upload part responses.
            List<UploadPartResponse> uploadResponses = new
 List<UploadPartResponse>();
            List<CopyPartResponse> copyResponses = new List<CopyPartResponse>();

            // Setup information required to initiate the multipart upload.
            InitiateMultipartUploadRequest initiateRequest =
                new InitiateMultipartUploadRequest
                {
                    BucketName = targetBucket,
                    Key = targetObjectKey
```

```
                };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // Save the upload ID.
            String uploadId = initResponse.UploadId;

            try
            {
                // Get the size of the object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = sourceBucket,
                    Key = sourceObjectKey
                };

                GetObjectMetadataResponse metadataResponse =
                    await s3Client.GetObjectMetadataAsync(metadataRequest);
                long objectSize = metadataResponse.ContentLength; // Length in
bytes.

                // Copy the parts.
                long partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
                for (int i = 1; bytePosition < objectSize; i++)
                {
                    CopyPartRequest copyRequest = new CopyPartRequest
                    {
                        DestinationBucket = targetBucket,
                        DestinationKey = targetObjectKey,
                        SourceBucket = sourceBucket,
                        SourceKey = sourceObjectKey,
                        UploadId = uploadId,
                        FirstByte = bytePosition,
                        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
                        PartNumber = i
                    };

                    copyResponses.Add(await s3Client.CopyPartAsync(copyRequest));
```

```
                bytePosition += partSize;
            }

            // Set up to complete the copy.
            CompleteMultipartUploadRequest completeRequest =
            new CompleteMultipartUploadRequest
            {
                BucketName = targetBucket,
                Key = targetObjectKey,
                UploadId = initResponse.UploadId
            };
            completeRequest.AddPartETags(copyResponses);

            // Complete the copy.
            CompleteMultipartUploadResponse completeUploadResponse =
                await s3Client.CompleteMultipartUploadAsync(completeRequest);
        }
        catch (AmazonS3Exception e)
        {
            Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
        }
        catch (Exception e)
        {
            Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
        }
    }
  }
}
```

使用 REST API

《Amazon Simple Storage Service API 參考》中的下列章節說明了分段上傳的 REST API。若要複製現有的物件，請使用分段上傳 (複製) API，並在要求中新增 `x-amz-copy-source` 要求標頭以指定來源物件。

- 啟動分段上傳
- 上傳片段
- 分段上傳 (複製)

- 完成分段上傳
- 中止分段上傳
- 列出組件
- 列出分段上傳

您可以使用這些 API 來提出自己的 REST 請求，也可以使用我們所提供的其中一個 SDK。如需搭配使用「分段上傳」的詳細資訊 AWS CLI，請參閱使用 AWS CLI。如需 SDK 的詳細資訊，請參閱「AWS SDK 支持多部分上傳」。

Amazon S3 分段上傳限制

下表提供分段上傳核心規格。如需詳細資訊，請參閱 使用分段上傳來上傳和複製物件。

項目	規格
物件大小上限	5 TiB
每次上傳的組件數目上限	10,000
組件編號	1 到 10,000 (含)
組件大小	5 MiB 至 5 GiB。您的分段上傳的最後一部分沒有大小下限。
列出組件要求的傳回組件數上限	1000
列出分段上傳要求所傳回的分段上傳數目上限	1000

複製、移動和重新命名物件

此CopyObject作業會建立已存放在 Amazon S3 中的物件副本。

您可以在單一原子作業中建立最多 5 GB 的物件副本。不過，若要複製大於 5 GB 的物件，您必須使用多部分上傳。如需詳細資訊，請參閱 the section called “複製物件”。

使用 CopyObject 操作，您可以：

- 建立物件的其他複本。
- 透過複製物件並刪除原始物件來重新命名物件。
- 將物件從一個值區複製或移動到另一個值區，包括跨值區 AWS 區域 (例如，從`us-west-1`到`eu-west-2`)。當您移動物件時，Amazon S3 會將物件複製到指定的目的地，然後刪除來源物件。

  > **Note**
  >
  > 跨越複製或移動物件 AWS 區域 會產生頻寬費用。如需詳細資訊，請參閱 Amazon S3 定價。

- 變更物件中繼資料。每個 Amazon S3 物件都有中繼資料。此元數據是一組名稱-值對。您可以在上載物件時設定物件中繼資料。上載物件之後，您就無法修改物件中繼資料。修改物件中繼資料唯一的方式是製作物件的複本，再設定中繼資料。若要這麼做，請在複製作業中，設定與來源和目標相同的物件。

  一些對象元數據是系統元數據，另一些是用戶定義的 您可以控制某些系統中繼資料。例如，您可以控制要用於物件的儲存區類別和伺服器端加密類型。當您複製物件時，也會一併複製使用者控制的系統中繼資料及使用者定義的中繼資料。Amazon S3 會重設系統控制的中繼資料。例如，複製物件時，Amazon S3 會重設複製物件的建立日期。您不需要在複製請求中設定任何這些系統控制的中繼資料值。

  複製物件時，可能會決定要更新部分的中繼資料值。例如，若來源物件設定為使用 S3 Standard 儲存體，可以為物件複本選擇使用 S3 Intelligent-Tiering。也有可能決定要改變來源物件上一部分的使用者定義中繼資料值。如果選擇在複製期間更新任何物件之使用者可設定的中繼資料 (系統或使用者定義)，則您必須在要求中明確指定存在於來源物件上之所有使用者可設定的中繼資料，即使只變更其中一個中繼資料值亦然。

  如需物件中繼資料的詳細資訊，請參閱「使用物件中繼資料」。

複製已封存和還原的物件

若來源物件封存在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 中，即必須先還原暫存副本，才可將物件複製到其他儲存貯體。如需封存物件的資訊，請參閱 轉換為 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 儲存類別 (物件封存)。

S3 冰川彈性擷取或 S3 Glacier 深層存檔儲存類別中還原的物件不支援 Amazon S3 主控台中的複製操作。若要複製這些還原的物件，請使用 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 Amazon S3 REST API。

複製加密物件

Amazon S3 會自動加密複製到 S3 儲存貯體的所有新物件。若您未在複製請求中指定加密資訊，目標物件的加密設定會設為目的地儲存貯體的預設加密組態。根據預設，所有儲存貯體都有基本層級的加密組態，其中包含使用伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3)。如果目的地儲存貯體的預設加密組態使用伺服器端加密 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 或客戶提供的加密金鑰 (SSE-C)，Amazon S3 會使用對應的 KMS 金鑰或客戶提供的金鑰來加密目標物件副本。

複製物件時，如果您想要對目標物件使用不同類型的加密設定，可以請求 Amazon S3 使用 KMS 金鑰、Amazon S3 受管金鑰或客戶提供的金鑰來加密目標物件。如果請求中的加密設定與目的地儲存貯體的預設加密組態不同，會優先使用請求中的加密設定。如果複製的來源物件使用 SSE-C 加密，您必須在請求中提供必要的加密資訊，以便 Amazon S3 能夠解密要複製的物件。如需詳細資訊，請參閱 使用加密來保護資料。

複製物件時使用總和檢查

複製物件時，您可以選擇對物件使用不同的檢查總和演算法。無論您選擇使用相同演算法還是新的演算法，Amazon S3 都會在複製物件後計算新的檢查總和值。Amazon S3 不會直接複製檢查總和的值。使用多部分上傳所載入之物件的總和檢查碼值可能會變更。如需如何計算此檢查總和的詳細資訊，請參閱「對分段上傳使用部分檢查總和」。

在單一要求中複製多個物件

若要透過單一請求複製多個 Amazon S3 物件，您也可以使用 S3 Batch 操作。您可以為 S3 批次操作提供一份要進行操作的物件清單。S3 批次操作會呼叫相應的 API 操作來執行指定操作。單一批次作業任務可在包含數 EB 資料的數十億個物件上執行指定的操作。

S3 批次操作功能會追蹤進度、傳送通知，並存放所有動作的詳細完成報告，提供完整受管、可稽核、無伺服器的體驗。您可以透過 Amazon S3 主控台、 AWS CLI AWS 開發套件或 REST API 使用 S3 Batch 操作。如需詳細資訊，請參閱 the section called “批次操作基礎知識”。

將物件複製到目錄值區

如需將物件複製到目錄值區的相關資訊，請參閱將物件複製到目錄儲存貯體。如需將 Amazon S3 快速單區域儲存類別與目錄儲存貯體搭配使用的相關資訊，請參閱什麼是 S3 Express One Zone？和目錄值區。

複製物件

若要複製物件，請使用下列方法。

使用 S3 主控台

> **Note**
>
> - 使用 Amazon S3 主控台複製物件時，您必須擁有`s3:ListAllMyBuckets`許可。控制台需要此權限才能驗證複製操作。如需授與此權限的原則範例，請參閱the section called “身分型政策範例”。
>
> 如果要複製具有使用者定義標籤的物件，您也必須擁有該`s3:GetObjectTagging`權限。如果您要複製沒有使用者定義標籤但大小超過 16 MB 的物件，您還必須擁有 s3:`GetObjectTagging` 權限。
>
> 如果目的地儲存貯體原則拒絕該`s3:GetObjectTagging`動作，則會複製物件而不使用使用者定義的標籤，而且您會收到錯誤訊息。
> - 使用 S3 主控台無法複製使用客戶提供的加密金鑰 (SSE-C) 加密的物件。若要複製使用 SSE-C 加密的物件，請使用 AWS CLI、 AWS 開發套件或 Amazon S3 REST API。
> - Amazon S3 主控台不支援跨區域複製使用 SSE-KMS 加密的物件。若要跨區域複製使用 SSE-KMS 加密的物件，請使用 AWS CLI、 AWS 開發套件或 Amazon S3 REST API。

複製物件

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇「值區」，然後選擇「一般用途值區」頁標。導覽至 Amazon S3 儲存貯體或資料夾，其中包含您要複製的物件。
3. 選取物件名稱左側的核取方塊，以複製這些物件。
4. 在 [動作] 功能表上，從顯示的選項清單中選擇 [複製]。
5. 選取目的地類型和目的地帳戶。若要指定目的地路徑，請選擇 Browse S3 (瀏覽 S3)，導覽至目的地，然後選取目的地左側的核取方塊。選擇右下角的 Choose destination (選擇目的地)。

   或者，輸入目的地路徑。
6. 如果您沒有啟用儲存貯體版本控制，系統可能會要求您確認覆寫具有相同名稱的現有物件。如果這沒有問題，請選取核取方塊並繼續。如果您要保留此儲存貯體中所有版本的物件，請選取 Enable Bucket Versioning (啟用儲存貯體版本控制)。您也可以更新預設加密和 S3 物件鎖定內容。

7. 在額外的檢查總和中，選擇使用現有的檢查總和函數，或是用新的檢查總和函數取代來複製物件。上傳物件時，您可以選擇指定用於驗證資料完整性的檢查總和演算法。複製物件時，您可以選擇一個新函數。如果原先沒有指定額外的檢查總和，您可以使用複製選項的部分新增一個檢查總和。

   > **Note**
   >
   > 即使您選擇使用相同的檢查總和函數，如果複製物件的大小超過 16 MB，則檢查總和的值可能會發生變化。檢查總和的值可能會因為計算分段上傳檢查總和而變更。如需複製物件時檢查總和會如何變化的詳細資訊，請參閱「對分段上傳使用部分檢查總和」。

   若要變更檢查總和函數，請選擇Replace with a new checksum function (替換為新的檢查總和函數)。從方塊中選擇新的檢查總和函數。複製物件時，將使用指定的演算法計算和儲存新的檢查總和。
8. 選擇右下角的 Copy (複製)。Amazon S3 會將您的物件複製到目的地。

使用 AWS 軟體開發套件

本節中的範例示範如何以單一操作複製最大可達 5 GB 的物件。若要複製大於 5 GB 的物件，您必須使用多部分上傳。如需詳細資訊，請參閱 使用分段上傳來複製物件。

Java

Example

下面的範例使用 AWS SDK for Java在 Amazon S3 中複製物件。如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;

import java.io.IOException;

public class CopyObjectSingleOperation {
```

```
    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String sourceKey = "*** Source object key *** ";
        String destinationKey = "*** Destination object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Copy the object into a new object in the same bucket.
            CopyObjectRequest copyObjRequest = new CopyObjectRequest(bucketName,
 sourceKey, bucketName, destinationKey);
            s3Client.copyObject(copyObjRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

下列 C# 範例會使用高階層 AWS SDK for .NET ，在單一作業中複製最大 5 GB 的物件。對於大於 5 GB 的物件，請使用 使用分段上傳來複製物件 中所述的分段上傳複製範例。

此範例會建立最大 5 GB 之物件的複本。如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;
```

```
namespace Amazon.DocSamples.S3
{
    class CopyObjectTest
    {
        private const string sourceBucket = "*** provide the name of the bucket with
 source object ***";
        private const string destinationBucket = "*** provide the name of the bucket
 to copy the object to ***";
        private const string objectKey = "*** provide the name of object to copy
 ***";
        private const string destObjectKey = "*** provide the destination object key
 name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Copying an object");
            CopyingObjectAsync().Wait();
        }

        private static async Task CopyingObjectAsync()
        {
            try
            {
                CopyObjectRequest request = new CopyObjectRequest
                {
                    SourceBucket = sourceBucket,
                    SourceKey = objectKey,
                    DestinationBucket = destinationBucket,
                    DestinationKey = destObjectKey
                };
                CopyObjectResponse response = await
 s3Client.CopyObjectAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
```

```
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

本主題將引導您使用第 3 版的類別，將 Amazon S3 中的單一物件和多個物件、從一個儲存貯體複製 AWS SDK for PHP 到另一個儲存貯體，或在同一個儲存貯體中複製單一物件和多個物件。

有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

下列 PHP 範例說明如何在 Amazon S3 內複製單一物件的`copyObject()`方法。它也會示範如何透過使用`getcommand()`方法的批次呼叫來`CopyObject`製作物件的多個複本。

複製物件

1	使用 `Aws\S3\S3Client` 類別建構函數，建立 Amazon S3 用戶端的執行個體。
2	若要製作物件的多個副本，您可以執行一批對 Amazon S3 用戶端getCommand()方法的呼叫，該方法是從Aws\CommandInterface類別繼承而來。您需要提供 `CopyObject` 命令作為第一個引數，並提供內含來源儲存貯體、來源金鑰名稱、目標儲存貯體與目標金鑰名稱的陣列，作為第二個引數。

```
 require 'vendor/autoload.php';

use Aws\CommandPool;
use Aws\Exception\AwsException;
use Aws\ResultInterface;
use Aws\S3\S3Client;

$sourceBucket = '*** Your Source Bucket Name ***';
$sourceKeyname = '*** Your Source Object Key ***';
$targetBucket = '*** Your Target Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region' => 'us-east-1'
```

```
]);

// Copy an object.
$s3->copyObject([
    'Bucket' => $targetBucket,
    'Key' => "$sourceKeyname-copy",
    'CopySource' => "$sourceBucket/$sourceKeyname",
]);

// Perform a batch of CopyObject operations.
$batch = array();
for ($i = 1; $i <= 3; $i++) {
    $batch[] = $s3->getCommand('CopyObject', [
        'Bucket' => $targetBucket,
        'Key' => "{targetKeyname}-$i",
        'CopySource' => "$sourceBucket/$sourceKeyname",
    ]);
}
try {
    $results = CommandPool::batch($s3, $batch);
    foreach ($results as $result) {
        if ($result instanceof ResultInterface) {
            // Result handling here
        }
        if ($result instanceof AwsException) {
            // AwsException handling here
        }
    }
} catch (Exception $e) {
    // General error handling here
}
```

Python

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                          that wraps object actions in a class-like structure.
```

```
        """
        self.object = s3_object
        self.key = self.object.key
```

```
    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
key.
                            This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise
```

Ruby

下列任務會引導您完成使用Ruby類別，將 Amazon S3 中的物件從一個儲存貯體複製到另一個儲存貯體或同一個儲存貯體內。

複製物件

1	將 Amazon S3 模組化寶石用於版本 3 AWS SDK for Ruby、需求`aws-sdk-s3` 並提供您 AWS 的登入資料。如需如何提供憑證的詳細資訊，請參閱「使用 AWS 帳戶 或 IAM 使用者登入資料提出請求」。
2	提供請求資訊，例如來源值區名稱、來源金鑰名稱、目的地值區名稱和目的地金鑰。

下列Ruby程式碼範例會使用將物件從一個值區複製到另一個值區的`#copy_object`方法，示範先前的工作。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
```

```
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

使用 REST API

此範例說明如何使用 Amazon S3 REST API 複製物件。如需 REST API 的詳細資訊，請參閱 CopyObject。

此範例會將 flotsam 物件從 *example-s3-bucket1* 儲存貯體複製到 jetsam 儲存貯體的 *example-s3-bucket2* 物件，並保留其中繼資料。

```
PUT /jetsam HTTP/1.1
Host: example-s3-bucket2.s3.amazonaws.com
x-amz-copy-source: /example-s3-bucket1/flotsam
Authorization: AWS AKIAIOSFODNN7EXAMPLE:ENoSbxYByFA0UGLZUqJN5EUnLDg=
Date: Wed, 20 Feb 2008 22:12:21 +0000
```

簽章已依據下列資訊產生。

```
PUT\r\n
\r\n
\r\n
Wed, 20 Feb 2008 22:12:21 +0000\r\n

x-amz-copy-source:/example-s3-bucket1/flotsam\r\n
/example-s3-bucket2/jetsam
```

Amazon S3 會傳回下列回應，其中指定物件的 ETag 及上次修改時間。

```
HTTP/1.1 200 OK
x-amz-id-2: Vyaxt7qEbzv34BnSu5hctyyNSlHTYZFMWK4FtzO+iX8JQNyaLdTshL0KxatbaOZt
x-amz-request-id: 6B13C3C5B34AF333
Date: Wed, 20 Feb 2008 22:13:01 +0000

Content-Type: application/xml
Transfer-Encoding: chunked
Connection: close
Server: AmazonS3
<?xml version="1.0" encoding="UTF-8"?>

<CopyObjectResult>
   <LastModified>2008-02-20T22:13:01</LastModified>
   <ETag>"7e9c608af58950deeb370c98608ed097"</ETag>
</CopyObjectResult>
```

使用 AWS CLI

您也可以使用 AWS Command Line Interface (AWS CLI) 複製 S3 物件。如需詳細資訊，請參閱 AWS CLI 命令參考中的 copy-object。

如需有關的資訊 AWS CLI，請參閱什麼是 AWS Command Line Interface? 在《AWS Command Line Interface 使用者指南》中。

如何移動物件

若要移動物件，請使用下列方法。

使用 S3 主控台

> **Note**
>
> - 如果您要移動具有使用者定義標籤的物件，您必須擁有該`s3:GetObjectTagging`權限。如果您要移動沒有使用者定義標籤但大小超過 16 MB 的物件，您還必須擁有 s3:GetObjectTagging 權限。
>
> 如果目的地儲存貯體政策拒絕該`s3:GetObjectTagging`動作，物件將不會移動使用者定義的標籤，而且您會收到錯誤訊息。

- 使用客戶提供的加密金鑰 (SSE-C) 加密的物件無法使用 Amazon S3 主控台移動。若要移動使用 SSE-C 加密的物件，請使用 AWS CLI、 AWS 開發套件或 Amazon S3 REST API。
- 移動資料夾時，請等待「移動」作業完成，然後再對資料夾進行其他變更。
- 您無法在 Amazon S3 主控台中使用 S3 存取點別名做為 Move 操作的來源或目的地。

如何移動物件

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇「值區」，然後選擇「一般用途值區」頁標。導覽至您要移動的物件所在的 Amazon S3 儲存貯體或資料夾。
3. 選取要移動的物件名稱左側的核取方塊。
4. 在 [動作] 功能表上選擇 [移動]。
5. 若要指定目的地路徑，請選擇 Browse S3 (瀏覽 S3)，導覽至目的地，然後選取目的地左側的核取方塊。選擇右下角的 Choose destination (選擇目的地)。

   或者，輸入目的地路徑。
6. 如果您沒有啟用儲存貯體版本控制，系統可能會要求您確認覆寫具有相同名稱的現有物件。如果這沒有問題，請選取核取方塊並繼續。如果您要保留此儲存貯體中所有版本的物件，請選取 Enable Bucket Versioning (啟用儲存貯體版本控制)。您也可以更新預設加密和物件鎖定內容。
7. 選擇右下角的 Move (移動)。Amazon S3 會將您的物件移動到目的地資料夾。

Note

- 此動作會以更新的設定建立所有指定物件的複本、更新指定位置中的上次修改日期，以及將刪除標記新增至原始物件。
- 此動作會更新儲存貯體版本控制、加密、物件鎖定功能及封存物件的中繼資料。

使用 AWS CLI

您也可以使用 AWS Command Line Interface (AWS CLI) 移動 S3 物件。如需詳細資訊，請參閱 AWS CLI 命令參考中的 mv。

如需有關的資訊 AWS CLI，請參閱什麼是 AWS Command Line Interface? 在《AWS Command Line Interface 使用者指南》中。

重新命名物件

欲重新命名物件，請遵循下列步驟。

> **Note**
>
> - 重新命名物件會建立具有新上次修改日期的物件副本，然後將刪除標記新增至原始物件。
> - 預設加密的儲存貯體設定會自動套用至任何未加密的指定物件。
> - 您無法使用 Amazon S3 主控台使用客戶提供的加密金鑰 (SSE-C) 重新命名物件。若要重新命名使用 SSE-C 加密的物件，請使用 AWS CLI、 AWS 開發套件或 Amazon S3 REST API 來複製具有新名稱的物件。
> - 如果此儲存貯體使用 S3 物件擁有權的儲存貯體擁有者強制設定，則不會複製物件存取控制清單 (ACL)。
> - 如果要重新命名具有使用者定義標籤的物件，您必須擁有該`s3:GetObjectTagging`權限。如果要重新命名沒有使用者定義標籤但大小超過 16 MB 的物件，您還必須擁有 s3:`GetObjectTagging` 權限。
>
>   如果目的地儲存貯體政策拒絕該`s3:GetObjectTagging`動作，物件將會重新命名，但使用者定義的標籤會從物件中移除，而且您會收到錯誤訊息。

重新命名物件

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇「值區」，然後選擇「一般用途值區」頁標。導覽至包含您要重新命名之物件的 Amazon S3 儲存貯體或資料夾。
3. 選取您要重新命名之物件名稱左側的核取方塊。
4. 請在「操作」菜單上選擇「重命名對象」。
5. 在「新物件名稱」方塊中，輸入物件的新名稱。
6. 選擇保存更改在右下角。Amazon S3 重命名您的對象。

下載物件

本節說明如何從 Amazon S3 儲存貯體下載物件。您可以將物件存放在 Amazon S3 的一或多個儲存貯體中，而且每個物件的大小上限可達 5 TB。任何未封存的 Amazon S3 物件都可即時存取。而封存的物件必須先還原才能下載。如需下載已封存物件的詳細資訊，請參閱 the section called “下載封存的物件”。

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 Amazon S3 REST API 下載單一物件。若要在不撰寫任何程式碼或執行任何命令的情況下從 S3 下載物件，請使用 S3 主控台。如需詳細資訊，請參閱 the section called “下載物件”。

若要下載多個物件 AWS CloudShell，請使用 AWS CLI、或 AWS SDK。如需詳細資訊，請參閱 the section called “下載多個物件”。

如果您需要下載物件的一部分，您可以使用額外的參數搭配 AWS CLI 或 REST API 來指定您要下載的位元組。如需詳細資訊，請參閱 the section called “下載物件的一部分”。

如果您需要下載您未擁有的物件，可以請物件擁有者產生預先簽章的 URL 讓您下載物件。如需詳細資訊，請參閱 the section called “從另一個 AWS 帳戶下載物件”。

當您在 AWS 網路外部下載物件時，需要支付資料傳輸費用。 AWS 網絡內的數據傳輸是免費的 AWS 區域，但是任何GET請求都將向您收取費用。如需資料傳輸成本和資料擷取收費的詳細資訊，請參閱 Amazon S3 定價。

主題

- 下載物件
- 下載多個物件
- 下載物件的一部分
- 從另一個 AWS 帳戶下載物件
- 下載封存的物件
- 下載物件的故障排除

下載物件

您可以使用 Amazon S3 主控台 AWS CLI、 AWS 開發套件或 REST API 下載物件。

使用 S3 主控台

本節說明如何使用 Amazon S3 主控台從 S3 儲存貯體下載物件。

> **Note**
>
> - 您一次只能下載一個物件。
> - 如果您使用 Amazon S3 主控台下載的物件，且其金鑰名稱結尾為句號 (.)，則會移除所下載物件的金鑰名稱中的句號。若要保留下載物件名稱末尾的期間，您必須使用 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 Amazon S3 REST API。

從 S3 儲存貯體下載物件

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇您要從中下載物件的儲存貯體名稱。
3. 您可使用下列任一方式從 S3 儲存貯體下載物件：
   - 勾選物件旁的核取方塊，然後選擇下載。如果您要將物件下載到特定資料夾，請在動作選單上選擇下載為。
   - 如果您要下載特定版本的物件，請開啟顯示版本 (位於搜尋方塊旁)。勾選您要的物件版本旁的核取方塊，然後選擇下載。如果您要將物件下載到特定資料夾，請在動作選單上選擇下載為。

使用 AWS CLI

以下 `get-object` 範例命令顯示如何使用 AWS CLI 從 Amazon S3 下載物件。此命令會從儲存貯體 *`example-s3-bucket1`* 取得物件 *`folder/my_image`*。物件將下載到名為 *`my_downloaded_image`* 的檔案中。

```
aws s3api get-object --bucket example-s3-bucket1 --key folder/
my_image my_downloaded_image
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 get-object。

使用 AWS 軟體開發套件

如需如何使用 AWS SDK 下載物件的範例，請參閱搭GetObject配 AWS 開發套件或 CLI 使用。

如需使用不同 AWS SDK 的一般資訊，請參閱使用開發 AWS 套件使用 Amazon S3 進行開發。

使用 REST API

您可以使用 REST API 從 Amazon S3 擷取物件。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 GetObject。

下載多個物件

您可以使用 AWS CloudShell、或 AWS SDK 下載多個物件。 AWS CLI

AWS CloudShell 在中使用 AWS Management Console

AWS CloudShell 是一個以瀏覽器為基礎的預先驗證殼層，您可以直接從. AWS Management Console

如需有關的詳細資訊 AWS CloudShell，請參閱什麼是 CloudShell？ 在《AWS CloudShell 使用者指南》中。

> **Important**
>
> 使用 AWS CloudShell，您的主目錄的存儲空間高達每 AWS 區域 1GB。因此，您無法同步物件總計超過此數量的儲存貯體。如要了解更多限制，請參閱《AWS CloudShell 使用者指南》中的 Service Quotas 和限制。

若要使用下載物件 AWS CloudShell

1. 請登入 AWS Management Console 並開啟 CloudShell 主控台，網址為 https://console.aws.amazon.com/cloudshell/。
2. 執行下列命令，將值區中的物件同步到 CloudShell。下列指令會同步來自名為的值區中的物件，*example-s3-bucket1*並建立名為 *temp* in CloudShell 的資料夾。 CloudShell 將您的物件同步到此資料夾。若要使用此命令，請以您自己的資訊取代 *user input placeholders*。

   ```
   aws s3 sync s3://example-s3-bucket1 ./temp
   ```

   > **Note**
   >
   > 若要執行模式比對以排除或包含特定物件，您可以使用 `--exclude "value"` 和 `--include "value"` 參數搭配 `sync` 命令。

3. 執行下列命令，將名為 *temp* 的資料夾中的物件壓縮成名為 *temp.zip* 的檔案。

```
zip temp.zip -r temp/
```

4. 選擇動作選單，然後選擇下載檔案。
5. 輸入檔案名稱 **temp.zip**，然後選擇下載。
6. (選擇性) 刪除*temp.zip*檔案和已同步至中*temp*資料夾的物件 CloudShell。在 AWS CloudShell 中，您擁有每個 AWS 區域最多 1 GB 的持久性儲存空間。

   您可以使用下列範例命令來刪除您的 .zip 檔案和資料夾。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

   ```
   rm temp.zip && rm -rf temp/
   ```

使用 AWS CLI

下列範例顯示如何使用下 AWS CLI 載指定目錄或前置詞下的所有檔案或物件。此命令會將儲存貯體 *example-s3-bucket1* 中的所有物件複製到您目前的目錄。若要使用此範例命令，請使用您的儲存貯體名稱取代 *example-s3-bucket1*。

```
aws s3 cp s3://example-s3-bucket1 . --recursive
```

下列命令會將儲存貯體 *example-s3-bucket1* 中字首為 *logs* 的所有物件下載到您目前的目錄。此命令還會使用 --exclude 和 --include 參數僅複製字尾為 *.log* 的物件。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3 cp s3://example-s3-bucket1/logs/ . --recursive --exclude "*" --include "*.log"
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 cp。

使用 AWS 軟體開發套件

如需如何使用 AWS 開發套件下載 Amazon S3 儲存貯體中所有物件的範例，請參閱將 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的所有物件下載至本機目錄。

如需使用不同 AWS SDK 的一般資訊，請參閱使用開發 AWS 套件使用 Amazon S3 進行開發。

下載物件的一部分

您可以使用 AWS CLI 或 REST API 下載物件的一部分。若要這麼做，請使用額外的參數來指定要下載物件的哪一部分。

使用 AWS CLI

下列範例命令會針對名為 *example-s3-bucket1* 的儲存貯體中，名為 *folder/my_data* 的物件的某個位元組範圍執行 GET 請求。在請求中，位元組範圍必須加上字首 bytes=。部分物件會下載到名為 *my_data_range* 的輸出檔案中。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3api get-object --bucket example-s3-bucket1 --key folder/my_data --range
 bytes=0-500 my_data_range
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 get-object。

如需 HTTP Range 標頭的詳細資訊，請參閱 RFC Editor 網站上的 RFC 9110。

> **Note**
>
> Amazon S3 不支援在單一 GET 請求中擷取多重資料範圍。

使用 REST API

您可以使用 REST API 中的 partNumber 和 Range 參數從 Amazon S3 擷取物件的部分。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 GetObject。

從另一個 AWS 帳戶下載物件

您可以使用預先簽章的 URL 授予他人對您的物件的限時存取權，而不需更新您的儲存貯體政策。

預先簽章的 URL 可以在瀏覽器中輸入，或由程式用來下載物件。URL 所使用的認證是產生 URL 之 AWS 使用者的認證。建立 URL 之後，任何人只要擁有預先簽章的 URL，都可以下載對應的物件，直到 URL 過期為止。

使用 S3 主控台中的預先簽章 URL

您可以執行以下步驟，使用 Amazon S3 主控台產生預先簽章 URL 以共用物件。當使用主控台時，預先簽章 URL 的最長時效為自建立時間起算 12 小時。

使用 Amazon S3 主控台產生預先簽章的 URL

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇儲存貯體名稱，該儲存貯體包含您要為其建立預先簽章 URL 的物件。
4. 在 Objects (物件) 清單中，選取要為其建立預先簽章 URL 的物件。
5. 在物件動作選單中，選擇使用預先簽章的 URL 來共用。
6. 指定預先簽章 URL 的有效期限。
7. 選擇 Create presigned URL (建立預先簽章的 URL)。
8. 當確認訊息出現時，URL 會自動複製到剪貼簿。如果您需要再次複製預先簽章的 URL，您將會看到一個複製按鈕。
9. 若要下載物件，請將 URL 貼入任何瀏覽器中，物件就會嘗試下載。

如需預先簽章的 URL 及其他建立方法的詳細資訊，請參閱 使用預先簽章的 URL。

下載封存的物件

若要降低不常存取物件的儲存成本，您可以封存這些物件。當您封存物件時，該物件會移入低成本的儲存空間，這表示您無法即時存取該物件。若要下載封存的物件，您必須先將它還原。

根據儲存類別而定，還原封存的物件可能需要幾分鐘到數小時不等的時間。您可以使用 Amazon S3 主控台、S3 Batch 操作、Amazon S3 REST API、 AWS 開發套件和 AWS Command Line Interface (AWS CLI) 來還原存檔物件。

如需說明，請參閱還原已封存的物件。還原封存的物件之後，就可以下載該物件。

下載物件的故障排除

當您嘗試從 Amazon S3 下載物件時，許可不足或不正確的儲存貯體或 AWS Identity and Access Management (IAM) 使用者政策可能會導致錯誤。這些問題經常會導致拒絕存取 (403 禁止) 錯誤，使得 Amazon S3 無法允許資源存取。

如要了解導致拒絕存取 (403 禁止) 錯誤的常見原因，請參閱 針對 Amazon S3 中的拒絕存取 (403 禁止) 錯誤進行疑難排解。

檢查物件完整性

Amazon S3 使用檢查總和值來驗證您上傳到 Amazon S3 或從 Amazon S3 下載的資料完整性。此外，您可以請求為儲存在 Amazon S3 中的任何物件計算另一個檢查總和值。您可以從多種檢查總和演算法中進行選擇，以便在上傳或複製資料時使用。Amazon S3 使用此演算法運算額外的檢查總和值並將其儲存為物件中繼資料的一部分。若要進一步了解如何使用其他總和檢查來驗證資料完整性，請參閱教學課程：使用其他總和檢查來檢查 Amazon S3 中資料的完整性。

上傳物件時，您也可以選擇將預先計算的檢查總和作為請求的一部分。Amazon S3 會將提供的檢查總和與其使用指定演算法計算的檢查總和進行比較。如果兩個值不相符，Amazon S3 會回報錯誤。

使用支持的檢查總和演算法

Amazon S3 為您提供選擇用於在上傳或下載過程中驗證資料的檢查總和演算法選項。您可以選擇以下 Secure Hash 演算法 (SHA) 或循環冗餘檢查 (CRC) 資料完整性演算法之一：

- CRC32
- CRC32C
- SHA-1
- SHA-256

上傳物件時，您可以指定要使用的演算法：

- 當您使用時 AWS Management Console，請選取您要使用的總和檢查碼演算法。執行此操作時，您可以選擇指定物件的檢查總和值。Amazon S3 接收物件時，它會使用您指定的演算法計算檢查總和。如果兩個檢查總和值不相符，Amazon S3 會產生錯誤。
- 當您使用的是 SDK，您可以將 `x-amz-sdk-checksum-algorithm` 參數設定為您希望 Amazon S3 在計算檢查總和時使用的演算法。Amazon S3 會自動計算檢查總和值。
- 當您使用 REST API 時，請勿使用 `x-amz-sdk-checksum-algorithm` 參數。相反，您可以使用指定演算法的標頭之一 (例如 `x-amz-checksum-crc32`)。

如需上傳物件的詳細資訊，請參閱「上傳物件」。

若要將這些檢查總和值應用到任何一個已上傳到 Amazon S3 的物件，您可以複製該物件。複製物件時，您可以指定使用現有檢查總和演算法還是使用新的演算法。在使用任何支持的機制複製物件時，您可以指定檢查總和演算法，包括 S3 批次操作。如需 S3 批次操作的詳細資訊，請參閱「在 Amazon S3 物件上執行大規模批次操作」。

> **Important**
>
> 如果您使用的是具有額外檢查總和的分段上傳，則分段號部分編號必須使用連續的部分編號。使用額外的檢查總和時，如果您嘗試使用非連續部分編號完成分段上傳請求，則 Amazon S3 會產生 HTTP `500 Internal Server Error` 錯誤。

上傳物件後，您可以取得檢查總和值，並將其與使用相同演算法計算的預先計算值或先前存儲的檢查總和值進行比較。

使用 S3 主控台

若要進一步了解如何使用主控台，以及如何指定上傳物件時要使用的檢查總和演算法，請參閱 上傳物件 和教學課程：使用其他檢查總和來檢查 Amazon S3 中資料的完整性。

使用 AWS 軟體開發套件

下列範例說明如何使用 AWS SDK 上傳具有多部分上傳的大型檔案、下載大型檔案，以及驗證多部分上傳檔案，所有這些都使用 SHA-256 進行檔案驗證。

Java

Example 範例：使用 SHA-256 上傳、下載和驗證大型檔案

如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import software.amazon.awssdk.auth.credentials.AwsCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
```

```
    import software.amazon.awssdk.services.s3.model.GetObjectRequest;
    import software.amazon.awssdk.services.s3.model.GetObjectResponse;
    import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
    import software.amazon.awssdk.services.s3.model.ObjectAttributes;
    import software.amazon.awssdk.services.s3.model.PutObjectTaggingRequest;
    import software.amazon.awssdk.services.s3.model.Tag;
    import software.amazon.awssdk.services.s3.model.Tagging;
    import software.amazon.awssdk.services.s3.model.UploadPartRequest;
    import software.amazon.awssdk.services.s3.model.UploadPartResponse;

    import java.io.File;
    import java.io.FileInputStream;
    import java.io.FileOutputStream;
    import java.io.IOException;
    import java.io.InputStream;
    import java.io.OutputStream;
    import java.nio.ByteBuffer;
    import java.security.MessageDigest;
    import java.security.NoSuchAlgorithmException;
    import java.util.ArrayList;
    import java.util.Base64;
    import java.util.List;

    public class LargeObjectValidation {
        private static String FILE_NAME = "sample.file";
        private static String BUCKET = "sample-bucket";
        //Optional, if you want a method of storing the full multipart object
 checksum in S3.
        private static String CHECKSUM_TAG_KEYNAME = "fullObjectChecksum";
        //If you have existing full-object checksums that you need to validate
 against, you can do the full object validation on a sequential upload.
        private static String SHA256_FILE_BYTES = "htCM5g7ZNdoSw8bN/
mkgiAhXt5MFoVowVg+LE9aIQmI=";
        //Example Chunk Size - this must be greater than or equal to 5MB.
        private static int CHUNK_SIZE = 5 * 1024 * 1024;

        public static void main(String[] args) {
            S3Client s3Client = S3Client.builder()
                    .region(Region.US_EAST_1)
                    .credentialsProvider(new AwsCredentialsProvider() {
                        @Override
                        public AwsCredentials resolveCredentials() {
                            return new AwsCredentials() {
                                @Override
```

```
                                public String accessKeyId() {
                                    return Constants.ACCESS_KEY;
                                }

                                @Override
                                public String secretAccessKey() {
                                    return Constants.SECRET;
                                }
                            };
                        }
                    })
                    .build();
            uploadLargeFileBracketedByChecksum(s3Client);
            downloadLargeFileBracketedByChecksum(s3Client);
            validateExistingFileAgainstS3Checksum(s3Client);
        }

        public static void uploadLargeFileBracketedByChecksum(S3Client s3Client) {
            System.out.println("Starting uploading file validation");
            File file = new File(FILE_NAME);
            try (InputStream in = new FileInputStream(file)) {
                MessageDigest sha256 = MessageDigest.getInstance("SHA-256");
                CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
                        .bucket(BUCKET)
                        .key(FILE_NAME)
                        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
                        .build();
                CreateMultipartUploadResponse createdUpload =
s3Client.createMultipartUpload(createMultipartUploadRequest);
                List<CompletedPart> completedParts = new ArrayList<CompletedPart>();
                int partNumber = 1;
                byte[] buffer = new byte[CHUNK_SIZE];
                int read = in.read(buffer);
                while (read != -1) {
                    UploadPartRequest uploadPartRequest =
UploadPartRequest.builder()

.partNumber(partNumber).uploadId(createdUpload.uploadId()).key(FILE_NAME).bucket(BUCKET).ch
                    UploadPartResponse uploadedPart =
s3Client.uploadPart(uploadPartRequest,
RequestBody.fromByteBuffer(ByteBuffer.wrap(buffer, 0, read)));
                    CompletedPart part =
CompletedPart.builder().partNumber(partNumber).checksumSHA256(uploadedPart.checksumSHA256()
```

```
                    completedParts.add(part);
                    sha256.update(buffer, 0, read);
                    read = in.read(buffer);
                    partNumber++;
                }
                String fullObjectChecksum =
Base64.getEncoder().encodeToString(sha256.digest());
                if (!fullObjectChecksum.equals(SHA256_FILE_BYTES)) {
                    //Because the SHA256 is uploaded after the part is uploaded; the
upload is bracketed and the full object can be fully validated.

s3Client.abortMultipartUpload(AbortMultipartUploadRequest.builder().bucket(BUCKET).key(FILE
                    throw new IOException("Byte mismatch between stored checksum and
upload, do not proceed with upload and cleanup");
                }
                CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder().parts(completedParts).build();
                CompleteMultipartUploadResponse completedUploadResponse =
s3Client.completeMultipartUpload(

CompleteMultipartUploadRequest.builder().bucket(BUCKET).key(FILE_NAME).uploadId(createdUplo
                Tag checksumTag =
Tag.builder().key(CHECKSUM_TAG_KEYNAME).value(fullObjectChecksum).build();
                //Optionally, if you need the full object checksum stored with the
file; you could add it as a tag after completion.

s3Client.putObjectTagging(PutObjectTaggingRequest.builder().bucket(BUCKET).key(FILE_NAME).t
            } catch (IOException | NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
            GetObjectAttributesResponse
                    objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                    .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
            System.out.println(objectAttributes.objectParts().parts());
            System.out.println(objectAttributes.checksum().checksumSHA256());
        }

        public static void downloadLargeFileBracketedByChecksum(S3Client s3Client) {
            System.out.println("Starting downloading file validation");
            File file = new File("DOWNLOADED_" + FILE_NAME);
            try (OutputStream out = new FileOutputStream(file)) {
                GetObjectAttributesResponse
```

```
                        objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                        .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
                //Optionally if you need the full object checksum, you can grab a
tag you added on the upload
                List<Tag> objectTags =
s3Client.getObjectTagging(GetObjectTaggingRequest.builder().bucket(BUCKET).key(FILE_NAME).b
                String fullObjectChecksum = null;
                for (Tag objectTag : objectTags) {
                    if (objectTag.key().equals(CHECKSUM_TAG_KEYNAME)) {
                        fullObjectChecksum = objectTag.value();
                        break;
                    }
                }
                MessageDigest sha256FullObject =
MessageDigest.getInstance("SHA-256");
                MessageDigest sha256ChecksumOfChecksums =
MessageDigest.getInstance("SHA-256");

                //If you retrieve the object in parts, and set the ChecksumMode to
enabled, the SDK will automatically validate the part checksum
                for (int partNumber = 1; partNumber <=
objectAttributes.objectParts().totalPartsCount(); partNumber++) {
                    MessageDigest sha256Part = MessageDigest.getInstance("SHA-256");
                    ResponseInputStream<GetObjectResponse> response =
s3Client.getObject(GetObjectRequest.builder().bucket(BUCKET).key(FILE_NAME).partNumber(part
                    GetObjectResponse getObjectResponse = response.response();
                    byte[] buffer = new byte[CHUNK_SIZE];
                    int read = response.read(buffer);
                    while (read != -1) {
                        out.write(buffer, 0, read);
                        sha256FullObject.update(buffer, 0, read);
                        sha256Part.update(buffer, 0, read);
                        read = response.read(buffer);
                    }
                    byte[] sha256PartBytes = sha256Part.digest();
                    sha256ChecksumOfChecksums.update(sha256PartBytes);
                    //Optionally, you can do an additional manual validation again
the part checksum if needed in addition to the SDK check
                    String base64PartChecksum =
Base64.getEncoder().encodeToString(sha256PartBytes);
                    String base64PartChecksumFromObjectAttributes =
objectAttributes.objectParts().parts().get(partNumber - 1).checksumSHA256();
```

```
                    if (!
base64PartChecksum.equals(getObjectResponse.checksumSHA256()) || !
base64PartChecksum.equals(base64PartChecksumFromObjectAttributes)) {
                        throw new IOException("Part checksum didn't match for the
 part");
                    }
                    System.out.println(partNumber + " " + base64PartChecksum);
                }
                //Before finalizing, do the final checksum validation.
                String base64FullObject =
 Base64.getEncoder().encodeToString(sha256FullObject.digest());
                String base64ChecksumOfChecksums =
 Base64.getEncoder().encodeToString(sha256ChecksumOfChecksums.digest());
                if (fullObjectChecksum != null && !
fullObjectChecksum.equals(base64FullObject)) {
                    throw new IOException("Failed checksum validation for full
 object");
                }
                System.out.println(fullObjectChecksum);
                String base64ChecksumOfChecksumFromAttributes =
 objectAttributes.checksum().checksumSHA256();
                if (base64ChecksumOfChecksumFromAttributes != null && !
base64ChecksumOfChecksums.equals(base64ChecksumOfChecksumFromAttributes)) {
                    throw new IOException("Failed checksum validation for full
 object checksum of checksums");
                }
                System.out.println(base64ChecksumOfChecksumFromAttributes);
                out.flush();
            } catch (IOException | NoSuchAlgorithmException e) {
                //Cleanup bad file
                file.delete();
                e.printStackTrace();
            }
        }

        public static void validateExistingFileAgainstS3Checksum(S3Client s3Client)
 {
            System.out.println("Starting existing file validation");
            File file = new File("DOWNLOADED_" + FILE_NAME);
            GetObjectAttributesResponse
                    objectAttributes =
 s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                    .objectAttributes(ObjectAttributes.OBJECT_PARTS,
 ObjectAttributes.CHECKSUM).build());
```

```
            try (InputStream in = new FileInputStream(file)) {
                MessageDigest sha256ChecksumOfChecksums =
 MessageDigest.getInstance("SHA-256");
                MessageDigest sha256Part = MessageDigest.getInstance("SHA-256");
                byte[] buffer = new byte[CHUNK_SIZE];
                int currentPart = 0;
                int partBreak =
 objectAttributes.objectParts().parts().get(currentPart).size();
                int totalRead = 0;
                int read = in.read(buffer);
                while (read != -1) {
                    totalRead += read;
                    if (totalRead >= partBreak) {
                        int difference = totalRead - partBreak;
                        byte[] partChecksum;
                        if (totalRead != partBreak) {
                            sha256Part.update(buffer, 0, read - difference);
                            partChecksum = sha256Part.digest();
                            sha256ChecksumOfChecksums.update(partChecksum);
                            sha256Part.reset();
                            sha256Part.update(buffer, read - difference,
 difference);
                        } else {
                            sha256Part.update(buffer, 0, read);
                            partChecksum = sha256Part.digest();
                            sha256ChecksumOfChecksums.update(partChecksum);
                            sha256Part.reset();
                        }
                        String base64PartChecksum =
 Base64.getEncoder().encodeToString(partChecksum);
                        if (!
base64PartChecksum.equals(objectAttributes.objectParts().parts().get(currentPart).checksumSH
 {
                            throw new IOException("Part checksum didn't match S3");
                        }
                        currentPart++;
                        System.out.println(currentPart + " " + base64PartChecksum);
                        if (currentPart <
 objectAttributes.objectParts().totalPartsCount()) {
                            partBreak +=
 objectAttributes.objectParts().parts().get(currentPart - 1).size();
                        }
                    } else {
                        sha256Part.update(buffer, 0, read);
```

```
                    }
                    read = in.read(buffer);
                }
                if (currentPart != objectAttributes.objectParts().totalPartsCount())
 {
                    currentPart++;
                    byte[] partChecksum = sha256Part.digest();
                    sha256ChecksumOfChecksums.update(partChecksum);
                    String base64PartChecksum =
 Base64.getEncoder().encodeToString(partChecksum);
                    System.out.println(currentPart + " " + base64PartChecksum);
                }

                String base64CalculatedChecksumOfChecksums =
 Base64.getEncoder().encodeToString(sha256ChecksumOfChecksums.digest());
                System.out.println(base64CalculatedChecksumOfChecksums);
                System.out.println(objectAttributes.checksum().checksumSHA256());
                if (!
base64CalculatedChecksumOfChecksums.equals(objectAttributes.checksum().checksumSHA256()))
 {
                    throw new IOException("Full object checksum of checksums don't
 match S3");
                }

            } catch (IOException | NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
        }
    }
```

使用 REST API

您可以傳送 REST 要求以上傳具有總和檢查碼值的物件，以驗證資料的完整性。PutObject您也可以使用GetObject或HeadObject擷取物件的總和檢查碼值。

使用 AWS CLI

您可以傳送 PUT 請求，以便在單一操作中上傳多達 5 GB 的物件。如需詳細資訊，請參閱《AWS CLI 命令參考》中的 `PutObject`。您也可以使用 `get-object` 和 `head-object` 以擷取已上傳物件的檢查總和以驗證資料的完整性。

如需相關資訊，請參閱AWS Command Line Interface 使用者指南中的 Amazon S3 CLI 常見問答集。

上傳物件時使用 Content-MD5

上傳後驗證物件完整性的另一種方法，是在上傳物件時提供物件的 MD5 摘要。如果要計算物件的 MD5 摘要，您可以使用 `Content-MD5` 標頭搭配 PUT 命令提供摘要。

上傳物件後，Amazon S3 會計算物件的 MD5 摘要，並將其與您提供的值進行比較。僅當兩個摘要匹配時，請求才會成功。

MD5 摘要不是強制需求，但您可以將其用於上傳過程以驗證物件的完整性。

使用 Content-MD5 和 ETag 驗證上傳的物件

物件的實體標籤 (ETag) 表示物件的特定版本。請記得，ETag 只會反映物件內容的變更，而非其中繼資料的變更。如果僅變更物件的中繼資料，則 ETag 將保持不變。

根據物件的不同，物件的 ETag 可能是物件資料的 MD5 摘要：

- 如果物件是由 `PutObject`、`PostObject`,或 `CopyObject` 操作所建立，或是通過 AWS Management Console，並且該物件也是採用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密或加密，則物件的 ETag 是該物件資料的 MD5 摘要。
- 如果物件是由`PutObject`、`PostObject`或`CopyObject`作業或透過、建立 AWS Management Console，且該物件使用客戶提供的金鑰 (SSE-C) 或伺服器端加密 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS) 加密，則該物件的 ETag 不是其物件資料的 MD5 摘要。
- 如果物件是由 `Multipart Upload` 或者 `Part Copy` 操作所建立，則無論加密方法為何，物件的 ETag 都不會是 MD5 摘要。如果物件大於 16 MB，則會以分段上傳的方式 AWS Management Console 上傳或複製該物件，因此 ETag 不會是 MD5 摘要。

對於物件 ETag 是否為 `Content-MD5` 摘要，您可以將物件的 ETag 值與計算值或先前儲存的 `Content-MD5` 摘要進行比較。

使用追蹤檢查總和

將物件上傳到 Amazon S3 時，您可以為物件提供預先計算的總和檢查碼，也可以使用 AWS SDK 代表您自動建立結尾總和檢查碼。如果您決定使用追蹤檢查總和，則 Amazon S3 會使用您指定的演算法自動產生檢查總和，並在上傳過程中使用該演算法驗證物件的完整性。

若要在使用 AWS SDK 時建立結尾總和檢查碼，請使用偏好的演算法填入`ChecksumAlgorithm`參數。SDK 使用該演算法來計算物件 (或對象物件) 的檢查總和，並自動將其附加到上傳請求的最後。此行為可節省您的時間，因為 Amazon S3 一次同時執行驗證和上傳您的資料。

> **Important**
>
> 如果您使用的是 S3 物件 Lambda，則對 S3 物件 Lambda 的所有請求都使用 `s3-object-lambda` 而不是 `s3`。此行為會影響追蹤檢查總和值的簽名。如需 S3 Object Lambda 的詳細資訊，請參閱 使用 S3 Object Lambda 轉換物件。

對分段上傳使用部分檢查總和

當物件上傳到 Amazon S3 時，它們可以作為單個物件上傳，也可以通過分段上傳過程上傳。通過主控台上傳大於 16 MB 的物件會使用分段上傳自動上傳。如需分段上傳的詳細資訊，請參閱「使用分段上傳來上傳和複製物件」。

當物件使用分段上傳上傳時，該物件的 ETag 不會是整個物件的 MD5 摘要 。Amazon S3 會在上傳時計算每個部分的 MD5 摘要。MD5 摘要用於確定最終物件的 ETag。Amazon S3 將 MD5 摘要的位元連接在一起，然後計算這些 MD5 摘要的連接值。建立 ETag 的最後步驟是 Amazon S3 在最後新增一個帶有總部分數的破折號。

例如，若物件的 ETag 為 `C9A5A6878D97B48CC965C1E41859F034-14`，請考慮使用分段上傳來上傳該物件。此案例中,`C9A5A6878D97B48CC965C1E41859F034` 是連接所有 MD5 摘要的連接值。所以，`-14` 表示有 14 個部分與此物件的分段上傳關聯。

如果您已為多部分物件啟用了其他檢查總和，則 Amazon S3 會使用指定的檢查總和演算法計算每個部分的檢查總和。已完成物件的檢查總和計算方式與 Amazon S3 為分段上傳計算 MD5 摘要的方式相同。您可以使用此檢查總和來驗證物件的完整性。

若要擷取有關物件的資訊，包括構成整個物件的零件數目，您可以使用此GetObjectAttributes作業。通過額外的檢查總和，您還可以復原包含每個部分檢查總和的值。

對於已完成的上傳，您可以使用GetObject或HeadObject操作並指定與單一零件對齊的零件編號或位元組範圍，以取得個別零件的總和檢查碼。如果您想要擷取仍在進行中的多部分上傳的個別部分的總和檢查碼值，可以使用。ListParts

由於 Amazon S3 計算分段上傳物件的檢查總和的方式，物件的檢查總和值可能會因為複製而變更。如果您使用的是開發套件或 REST API 並進行呼叫 CopyObject，Amazon S3 會根據 `CopyObject` API 操作的大小限制複製任何物件。無論物件是由單個請求上傳還是作為分段上傳的一部分，Amazon S3 都會將此複製作為單獨操作進行。使用複製命令，物件的檢查總和是完整物件的直接檢查總和。如果對象最初是使用分段上傳上傳，即使資料沒有變更，檢查總和的值也會變更。

> **Note**
>
> 超過大小上限的物件，`CopyObject` API 操作必須使用分段複製命令。

> **Important**
>
> 當您使用執行某些操作時 AWS Management Console，如果物件大小超過 16 MB，Amazon S3 會使用多部分上傳。這種情況下，檢查總和不會是完整物件的直接檢查總和，而是基於每個部分的計算的檢查總和值。
> 例如，若物件的大小為 100 MB，請考慮作為單個部分直接使用 REST API 上傳該物件。在這種情況下，檢查總和是整個物件的檢查總和。如果您稍後使用主控台重命名該、複製該物件，更改儲存類別或編輯中繼資料，則 Amazon S3 將使用分段上傳功能來更新物件。因此，Amazon S3 以各個部分的檢查總和值計算，為物件建立一個新的檢查總和值。
> 先前的主控台操作清單不是您可以採取的所有可能動作的完整清單， AWS Management Console 這些動作會導致 Amazon S3 使用多部分上傳功能更新物件。請記住，無論使用主控台執行大小超過 16 MB 的物件，檢查總和的值很有可能不是整個物件的檢查總和值。

刪除 Amazon S3 物件

您可以使用 Amazon S3 主控台、 AWS 開發套件 AWS Command Line Interface (AWS CLI) 或 REST API，直接從 Amazon S3 刪除一或多個物件。因為 S3 儲存貯體中的所有物件都會導致儲存成本，所以您應該刪除不再需要的物件。例如，如果您要收集日誌檔案，最好在不再需要它們時將其刪除。您可以設定生命週期規則以自動刪除如日誌檔案等物件。如需詳細資訊，請參閱 the section called “設定生命週期組態”。

如需 Amazon S3 功能與定價的相關資訊，請參閱 Amazon S3 定價。

刪除物件時有下列 API 選項：

- 刪除單一物件 – Amazon S3 提供 `DELETE` (`DeleteObject`) API 操作，以用來透過單一 HTTP 要求刪除一個物件。
- 刪除多個物件 – Amazon S3 提供多物件刪除 (`DeleteObjects`) API 操作，以用來透過單一 HTTP 要求來刪除最多 1,000 個物件。

從未啟用版本控制的儲存貯體中刪除物件時，您只提供物件索引鍵名稱。不過，從啟用版本控制的儲存貯體中刪除物件時，您可以選擇性地提供物件的版本 ID，以刪除物件的特定版本。

以編程方式從啟用版本控制的儲存貯體中刪除物件

如果儲存貯體已啟用版本控制，則儲存貯體中可以有同一個物件的多個版本。使用已啟用版本控制的儲存貯體時，刪除 API 操作會啟用下列選項：

- 指定未使用版本控制的刪除要求 — 您只指定物件的索引鍵，而非版本 ID。在此情況下，Amazon S3 會建立刪除標記，並在回應中傳回其版本 ID。這會讓物件從儲存貯體中消失。如需物件版本控制與刪除標記概念的資訊，請參閱「在 S3 儲存貯體中使用版本控制」。
- 指定版本控制的刪除要求 — 您同時指定索引鍵與版本 ID。在此情況下，可能會有下列兩個結果：
  - 如果版本 ID 對應至特定物件版本，則 Amazon S3 會刪除物件的特定版本。
  - 如果版本 ID 對應至該物件的刪除標記，則 Amazon S3 會將刪除標記刪除。這會讓物件重新出現在儲存貯體中。

刪除啟用 MFA 功能之儲存貯體中的物件

刪除已啟用多重要素驗證 (MFA) 之儲存貯體中的物件時，請注意以下項目：

- 如果您提供無效的 MFA 字符，則請求一律會失敗。
- 如果您擁有已啟用 MFA 的儲存貯體，並且提出已使用版本控制的刪除請求 (您提供物件索引鍵與版本 ID)，則在未提供有效的 MFA 字符時，請求會失敗。此外，對已啟用 MFA 的儲存貯體使用多物件刪除 API 操作時，如果任何刪除是已使用版本控制的刪除請求 (亦即，您指定物件索引鍵與版本 ID)，則在未提供 MFA 字符時，整個請求會失敗。

但是，在下列情況下，要求會成功：

- 如果您擁有已啟用 MFA 的儲存貯體，並且提出未使用版本控制的刪除請求 (您不是要刪除有版本控制的物件)，且不提供 MFA 字符，刪除會成功。
- 如果您有只指定未使用版本控制物件的多物件刪除請求，要在已啟用 MFA 的儲存貯體中執行刪除作業，而您不提供 MFA 字符，刪除會成功。

如需 MFA Delete 的資訊，請參閱「設定 MFA Delete」。

主題

刪除單一物件

您可以使用 Amazon S3 主控台或 DELETE API 從 S3 儲存貯體中刪除單一現有物件。如需在 Amazon S3 中使用刪除物件的詳細資訊，請參閱「刪除 Amazon S3 物件」。

因為 S3 儲存貯體中的所有物件都會導致儲存成本，所以您應該刪除不再需要的物件。例如，如果您要收集日誌檔案，最好在不再需要它們時將其刪除。您可以設定生命週期規則以自動刪除如日誌檔案等物件。如需詳細資訊，請參閱 the section called “設定生命週期組態”。

如需 Amazon S3 功能與定價的相關資訊，請參閱 Amazon S3 定價。

使用 S3 主控台

請依照下列步驟使用 Amazon S3 主控台從儲存貯體中刪除單一物件。

> **Warning**
>
> 當您在 Amazon S3 主控台中永久刪除物件或指定物件版本時，刪除作業無法復原。

若要刪除已啟用或暫停版本化的物件

> **Note**
>
> 如果暫停版本控制的儲存貯體中物件的版本 ID 標記為NULL，S3 會永久刪除該物件，因為沒有先前的版本存在。但是，如果針對已暫停版本控制的儲存貯體中的物件列出了有效的版本 ID，則 S3 會為刪除的物件建立刪除標記，同時保留該物件的先前版本。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Bucket name (儲存貯體名稱) 清單中，選擇您要刪除物件所在的儲存貯體名稱。
3. 選取物件，然後選擇 [刪除]。
4. 確認刪除「刪除物件」中「指定的物件」下的物件清單的步驟 」文字方塊中，輸入**delete**。

若要永久刪除已啟用版本控制的值區中的特定物件版本

> **Warning**
>
> 當您在 Amazon S3 中永久刪除特定物件版本時，刪除作業無法復原。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Bucket name (儲存貯體名稱) 清單中，選擇您要刪除物件所在的儲存貯體名稱。
3. 選取您要刪除的物件。
4. 選擇「顯示版本」切換。
5. 選取物件版本，然後選擇「刪除」。
6. 若要確認永久刪除「指定物件」下列出的特定物件版本，請在「刪除物件？」 」文字方塊中，輸入永久刪除。Amazon S3 會永久刪除特定物件版本。

若要永久刪除未啟用版本控制的 Amazon S3 儲存貯體中的物件

> **Warning**
>
> 當您永久刪除 Amazon S3 中的物件時，刪除作業無法復原。此外，對於任何未啟用版本控制的值區，刪除都是永久性的。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Bucket name (儲存貯體名稱) 清單中，選擇您要刪除物件所在的儲存貯體名稱。
3. 選取物件，然後選擇 [刪除]。
4. 若要確認永久刪除「指定物件」下列出的物件，請在「刪除物件？」 」文字方塊中，輸入永久刪除。

> **Note**
>
> 如果您在刪除物件時遇到任何問題，請參閱我想永久刪除版本控制的物件。

使用 AWS 軟體開發套件

下列範例說明如何使用 AWS SDK 從值區刪除物件。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 DELETE 物件。

如果您有已啟用 S3 版本控制的儲存貯體，則您有下列選項：

- 透過指定版本 ID 來刪除物件的特定版本。
- 若要刪除未指定版本 ID 的物件，Amazon S3 會在此物件上新增刪除標記。

如需 S3 版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。

Java

Example 範例 1：刪除物件 (未啟用版本控制的儲存貯體)

下列範例假定情況為尚未啟用版本控制的儲存貯體，或尚沒有任何版本 IDs 的物件。在刪除要求中，您可僅指定物件金鑰，而不用版本 ID。

如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectRequest;

import java.io.IOException;

public class DeleteObjectNonVersionedBucket {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
```

```
                    .build();

            s3Client.deleteObject(new DeleteObjectRequest(bucketName, keyName));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example 範例 2：刪除物件 (啟用版本控制的儲存貯體)

下列範例會示範刪除版本控制的儲存貯體裡之物件。範例將示範刪除指定的物件版本，依照指定物件金鑰名稱和版本 ID 即可達成。

此範例執行下列操作：

1. 新增範例物件至儲存貯體。Amazon S3 會傳回最近新增之物件的版本 ID。範例中，會在刪除要求中使用此版本 ID。
2. 刪除指定的物件版本，依照指定物件金鑰名稱和版本 ID 即可達成。如果此物件沒有其他的版本，則 Amazon S3 會刪除所有的物件。不然的話，Amazon S3 只會刪除指定的版本。

> **Note**
>
> 您可以傳送 `ListVersions` 要求，取得物件的版本 ID。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
```

```
import com.amazonaws.services.s3.model.DeleteVersionRequest;
import com.amazonaws.services.s3.model.PutObjectResult;

import java.io.IOException;

public class DeleteObjectVersionEnabledBucket {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Check to ensure that the bucket is versioning-enabled.
            String bucketVersionStatus =
 s3Client.getBucketVersioningConfiguration(bucketName).getStatus();
            if (!bucketVersionStatus.equals(BucketVersioningConfiguration.ENABLED))
 {
                System.out.printf("Bucket %s is not versioning-enabled.",
 bucketName);
            } else {
                // Add an object.
                PutObjectResult putResult = s3Client.putObject(bucketName, keyName,
                        "Sample content for deletion example.");
                System.out.printf("Object %s added to bucket %s\n", keyName,
 bucketName);

                // Delete the version of the object that we just created.
                System.out.println("Deleting versioned object " + keyName);
                s3Client.deleteVersion(new DeleteVersionRequest(bucketName, keyName,
 putResult.getVersionId()));
                System.out.printf("Object %s, version %s deleted\n", keyName,
 putResult.getVersionId());
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
```

```
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

下列範例示範如何從已版本控制和未進行版本控制的儲存貯體中刪除物件。如需 S3 版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。

Example 刪除未進行版本控制的儲存貯體中之物件

下列 C# 範例會刪除未進行版本控制的儲存貯體中之物件。此範例假定情況為尚未有版本 IDs 的物件，也就是說您不必指定版本 ID。僅指定物件金鑰。

如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DeleteObjectNonVersionedBucketTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** object key ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            DeleteObjectNonVersionedBucketAsync().Wait();
        }
```

```
        private static async Task DeleteObjectNonVersionedBucketAsync()
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(deleteObjectRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
        }
    }
}
```

Example 刪除版本控制的儲存貯體中之物件

下列 C# 範例會刪除版本控制的儲存貯體裡之物件。範例刪除指定的物件版本，藉指定物件金鑰名稱和版本 ID 即可達成。

此程式碼會執行下列任務：

1. 對您所指定的儲存貯體啟用 S3 版本控制 (如果 S3 版本控制早已啟用，則不受此步驟影響)。
2. 新增範例物件至儲存貯體。據此，Amazon S3 會傳回最近新增之物件的版本 ID。範例中，會在刪除要求中使用此版本 ID。
3. 刪除範例版本，依照指定物件金鑰名稱和版本 ID 即可達成。

> **Note**
>
> 您也可以傳送 `ListVersions` 要求，取得物件的版本 ID。

```
var listResponse = client.ListVersions(new ListVersionsRequest { BucketName
 = bucketName, Prefix = keyName });
```

如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DeleteObjectVersion
    {
        private const string bucketName = "*** versioning-enabled bucket name ***";
        private const string keyName = "*** Object Key Name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            CreateAndDeleteObjectVersionAsync().Wait();
        }

        private static async Task CreateAndDeleteObjectVersionAsync()
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(keyName);

                // Delete the object by specifying an object key and a version ID.
                DeleteObjectRequest request = new DeleteObjectRequest
                {
                    BucketName = bucketName,
```

```
                    Key = keyName,
                    VersionId = versionID
                };
                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
        }

        static async Task<string> PutAnObject(string objectKey)
        {
            PutObjectRequest request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                ContentBody = "This is the content body!"
            };
            PutObjectResponse response = await client.PutObjectAsync(request);
            return response.VersionId;
        }
    }
}
```

PHP

此範例顯示如何使用版本 3 中的類別，從未建立版本化的值 AWS SDK for PHP 區刪除物件。如需刪除已使用版本控制之儲存貯體中物件的資訊，請參閱「使用 REST API」。

有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

下列 PHP 範例會刪除儲存貯體中的物件。因為此範例說明如何刪除未進行版本控制的儲存貯體的物件，所以它僅在刪除要求中提供儲存貯體名稱和物件索引鍵 (非版本 ID)。

```
<?php
```

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// 1. Delete the object from the bucket.
try
{
    echo 'Attempting to delete ' . $keyname . '...' . PHP_EOL;

    $result = $s3->deleteObject([
        'Bucket' => $bucket,
        'Key'    => $keyname
    ]);

    if ($result['DeleteMarker'])
    {
        echo $keyname . ' was deleted or does not exist.' . PHP_EOL;
    } else {
        exit('Error: ' . $keyname . ' was not deleted.' . PHP_EOL);
    }
}
catch (S3Exception $e) {
    exit('Error: ' . $e->getAwsErrorMessage() . PHP_EOL);
}

// 2. Check to see if the object was deleted.
try
{
    echo 'Checking to see if ' . $keyname . ' still exists...' . PHP_EOL;

    $result = $s3->getObject([
        'Bucket' => $bucket,
        'Key'    => $keyname
    ]);
```

```
    echo 'Error: ' . $keyname . ' still exists.';
}
catch (S3Exception $e) {
    exit($e->getAwsErrorMessage());
}
```

Javascript

```
import { DeleteObjectCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js" // Helper function that creates Amazon
 S3 service client module.

export const bucketParams = { Bucket: "BUCKET_NAME", Key: "KEY" };

export const run = async () => {
  try {
    const data = await s3Client.send(new DeleteObjectCommand(bucketParams));
    console.log("Success. Object deleted.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

使用 AWS CLI

若要一個請求刪除一個物件，請使用 DELETE API。如需詳細資訊，請參閱 DELETE 物件。如需有關使用 CLI 刪除物件的詳細資訊，請參閱 delete-object。

使用 REST API

您可以使用 AWS SDK 刪除物件。但也可視應用程式所需，直接傳送 REST 要求。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 DELETE 物件。

刪除多個物件

因為 S3 儲存貯體中的所有物件都會導致儲存成本，所以您應該刪除不再需要的物件。例如，如果您要收集日誌檔案，最好在不再需要它們時將其刪除。您可以設定生命週期規則以自動刪除如日誌檔案等物件。如需詳細資訊，請參閱 the section called “設定生命週期組態”。

如需 Amazon S3 功能與定價的相關資訊，請參閱 Amazon S3 定價。

您可以使用 Amazon S3 主控台、 AWS 開發套件或 REST API，同時從 S3 儲存貯體刪除多個物件。

使用 S3 主控台

請依照下列步驟使用 Amazon S3 主控台從儲存貯體中刪除多個物件。

Warning

- 刪除指定的物件後，即無法復原。
- 此動作會刪除所有指定的物件。刪除資料夾時，請等待刪除動作完成，然後再將新物件加入至資料夾。否則，系統也可能會刪除新物件。
- 在未啟用版本控制的情況下刪除儲存貯體中的物件時，Amazon S3 將永久刪除這些物件。
- 刪除已啟用或暫停儲存貯體版本控制的儲存貯體中的物件時，Amazon S3 會建立刪除標記。如需詳細資訊，請參閱使用刪除標記。

若要刪除已啟用或暫停版本化的物件

Note

如果暫停版本控制的儲存貯體中物件的版本 ID 標記為NULL，S3 會永久刪除這些物件，因為沒有先前的版本存在。但是，如果針對已暫停版本控制的儲存貯體中的物件列出了有效的版本 ID，則 S3 會為刪除的物件建立刪除標記，同時保留物件的先前版本。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在「值區名稱」清單中，選擇您要從中刪除物件的值區名稱。
3. 選取物件，然後選擇「刪除」。
4. 確認刪除「刪除物件」中「指定的物件」下的物件清單的步驟 」文字方塊中，輸入**delete**。

若要永久刪除已啟用版本控制的值區中的特定物件版本

> **Warning**
>
> 當您永久刪除 Amazon S3 中的特定物件版本時，刪除作業無法復原。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在「值區名稱」清單中，選擇您要從中刪除物件的值區名稱。
3. 選取您要刪除的 物件。
4. 選擇「顯示版本」切換。
5. 選取物件版本，然後選擇「刪除」。
6. 若要確認永久刪除「指定物件」下列出的特定物件版本，請在「刪除物件？」 」文字方塊中，輸入永久刪除。Amazon S3 會永久刪除特定物件版本。

若要永久刪除未啟用版本控制的 Amazon S3 儲存貯體中的物件

> **Warning**
>
> 當您永久刪除 Amazon S3 中的物件時，刪除作業無法復原。此外，對於任何未啟用版本控制的值區，刪除都是永久性的。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在「值區名稱」清單中，選擇您要從中刪除物件的值區名稱。
3. 選取物件，然後選擇「刪除」。
4. 若要確認永久刪除「指定物件」下列示的物件，請在「刪除物件？」 」文字方塊中，輸入永久刪除。

> **Note**
>
> 如果您在刪除物件時遇到任何問題，請參閱我想永久刪除版本控制的物件。

使用 AWS 軟體開發套件

如需如何使用 AWS SDK 刪除多個物件的範例，請參閱搭DeleteObjects配 AWS 開發套件或 CLI 使用。

如需使用不同 AWS SDK 的一般資訊，請參閱使用開發 AWS 套件使用 Amazon S3 進行開發。

使用 REST API

您可以使用 AWS SDK 使用多物件刪除 API 刪除多個物件。但也可視應用程式所需，直接傳送 REST 要求。

如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的刪除多個物件。

整理、列出和使用物件

在 Amazon S3 中，您可以使用字首來整理存儲體。字首是儲存貯體中物件的邏輯分組。字首值類似於目錄名稱，可在儲存貯體的相同目錄底下存放類似的資料。當您以程式設計方式上傳物件時，您可以使用字首來整理資料。

在 Amazon S3 主控台中，字首稱為資料夾。您可以透過在 S3 主控台中導航到儲存貯體，來檢視所有物件和資料夾。您也可以檢視每個物件的相關資訊，包括物件屬性。

如需在 Amazon S3 中列出和整理資料的詳細資訊，請參閱下列主題。

主題

- 使用字首整理物件
- 以程式設計方式列出物件索引鍵
- 在 Amazon S3 主控台中使用資料夾整理物件
- 在 Amazon S3 主控台中檢視物件概觀
- 在 Amazon S3 主控台中檢視物件屬性

使用字首整理物件

您可以使用字首來整理儲存在 Amazon S3 儲存貯體中的資料。字首是物件索引鍵名稱開頭的字元字串。字首可以是任意長度，以物件索引鍵名稱的最大長度 (1,024 個位元組) 為準。您可以將字首視為以類似於目錄的方式組織資料的一種方式。但是，字首不是目錄。

按字首搜尋會將結果限制為僅以指定字首為開頭的那些索引鍵。分隔符號會導致清單操作將共用常見字首的所有金鑰彙總至單一摘要清單結果。

字首與分隔符號參數的目的，在於協助您以階層方式整理以及瀏覽金鑰。若要執行這項操作，請先為您的儲存貯體選取一個在任何預期的金鑰名稱中都不會出現的分隔符號，例如斜線 (/)。您可以使用另一個字元作為分隔符號。斜線 (/) 字元沒有唯一性，是常見的字首分隔符號。接下來，串連所有包含的階層層級，並以分隔符號分隔每個層級，藉此建構您的金鑰名稱。

例如，如果您存放城市的相關資訊，可能自然而然地會先依洲別、國家/地區，然後再依省/市或州來整理這些城市。因為這些名稱一般不包含標點符號，所以您可以使用斜線 (/) 作為分隔符號。下列範例使用斜線 (/) 分隔符號。

- Europe/France/Nouvelle-Aquitaine/Bordeaux
- North America/Canada/Quebec/Montreal
- North America/USA/Washington/Bellevue
- North America/USA/Washington/Seattle

如果您以這種方式存放全球每個城市的資料，會很難管理一般金鑰的命名空間。利用清單作業使用 `Prefix` 與 `Delimiter`，您就可以使用已建立的階層來列出資料。例如，若要列出美國的所有州別，請設定 `Delimiter='/'` 和 `Prefix='North America/USA/'`。若要列出您在加拿大擁有資料的所有省份，請設定 `Delimiter='/'` 及 `Prefix='North America/Canada/'`。

如需詳細了解分隔符號、字首和巢狀資料夾，請參閱字首與巢狀資料夾的差異。

使用字首和分隔符號列出物件

如果發出具有分隔符號的清單要求，可讓您只瀏覽階層中的某一個層級，而略過巢狀於更深層級的金鑰 (可能有數百萬個) 並加以彙總。例如，假設您有具有下列索引鍵的儲存貯體 (*DOC-EXAMPLE-BUCKET*)：

```
sample.jpg

photos/2006/January/sample.jpg

photos/2006/February/sample2.jpg

photos/2006/February/sample3.jpg

photos/2006/February/sample4.jpg
```

此範本儲存貯體在根層級只有 sample.jpg 物件。若只要列出儲存貯體中的根層級物件，可以對儲存貯體傳送使用斜線 (/) 分隔符號字元的 GET 要求。Amazon S3 會在回應中傳回 sample.jpg 物件金鑰，因為它並未包含 / 分隔符號字元。其他所有金鑰則包含此分隔符號字元。Amazon S3 會將這些索引鍵分組，並傳回字首值為 photos/ 的單一 CommonPrefixes 元素，這是從這些索引鍵開頭到第一次出現指定的分隔符號為止的子字串。

Example

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>DOC-EXAMPLE-BUCKET</Name>
  <Prefix></Prefix>
  <Marker></Marker>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>sample.jpg</Key>
    <LastModified>2011-07-24T19:39:30.000Z</LastModified>
    <ETag>&amp;quot;d1a7fb5eab1c16cb4f7cf341cf188c3d&amp;quot;</ETag>
    <Size>6</Size>
    <Owner>
      <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
      <DisplayName>displayname</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <CommonPrefixes>
    <Prefix>photos/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

如需以程式設計方式列出物件索引鍵的詳細資訊，請參閱「以程式設計方式列出物件索引鍵」。

以程式設計方式列出物件索引鍵

在 Amazon S3 中，索引鍵可以依字首列出。您可以為相關索引鍵名稱選擇共用字首，並使用特殊字元標示這些索引鍵，以劃分層次結構。然後，您可以使用清單操作，以階層方式選取和瀏覽索引鍵。類似於檔案存放在檔案系統目錄中的方式。

Amazon S3 提供清單操作，讓您可以列舉儲存貯體中所包含的金鑰。這些金鑰會依儲存貯體及字首選取列出。例如，假設有一個名為 "dictionary" 的儲存貯體，其中包含一個代表所有英文文字的索引

鍵。您可以進行呼叫，列出該儲存貯體中以字母 "q" 開頭的所有金鑰。清單結果一律會依 UTF-8 二進位順序傳回。

SOAP 與 REST 清單操作都會傳回 XML 文件，其中包含相符金鑰的名稱，以及每個金鑰所識別之物件的相關資訊。

> **Note**
>
> HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。我們建議您使用 REST API 或 AWS 開發套件，而不是使用 SOAP。

皆為同一個字首並以特殊分隔符號結尾的金鑰群組，可依該共同字首彙總以方便列出。應用程式如此即可用階層方式組織及瀏覽其金鑰，就像是將檔案分組到檔案系統中不同的目錄一樣。

例如，若要展開 dictionary 儲存貯體以包含英文以外的文字時，可以在每個字前面加上其語言和分隔符號 (例如 "`French/logical`") 來組成索引鍵。使用此命名配置和階層式清單功能，您就可以只擷取法文文字清單。您也可以瀏覽最上層的可用語言清單，而無須逐一查看詞典編纂的所有金鑰。如需列出清單方面的詳細資訊，請參閱「使用字首整理物件」。

REST API

也可視應用程式所需，直接傳送 REST 要求。您可以傳送 GET 要求以傳回儲存貯體中的部分或所有物件，或使用挑選準則以傳回儲存貯體中一部分的物件。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 GET Bucket (列出物件) 第 2 版。

清單實作效率

清單效能不會受儲存貯體中的索引鍵總數所影響。它也不會受到 `prefix`、`marker`、`maxkeys` 或 `delimiter` 引數的存在或不存在的影響。

逐一查看多頁結果

因為儲存貯體幾乎可以包含不限數目的金鑰，所以完整的清單查詢結果可能會相當龐大。為管理大型結果集，Amazon S3 API 支援利用分頁方式將其分成多個回應。每個列出金鑰回應最多會傳回含有 1,000 個金鑰的頁面，並指出回應是否遭到截斷。您會傳送一系列清單金鑰要求，直到您收到所有金鑰為止。 AWS SDK 包裝函式庫提供相同的分頁。

範例

下列程式碼範例會示範如何使用`ListObjects`。

CLI

AWS CLI

下列範例會使用`list-objects`命令來顯示指定值區中所有物件的名稱：

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size:
 Size}'
```

此範例使用`--query`引數將輸出篩選為每個物件的`list-objects`索引鍵值和大小

如需有關物件的詳細資訊，請參閱 Amazon S3 開發人員指南中的使用 Amazon S3 物件。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListObjects中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會擷取值區「測試檔案」中所有項目的相關資訊。

```
Get-S3Object -BucketName test-files
```

示例 2：此命令從存儲桶「測試文件」中檢索有關項目「sample.txt」的信息。

```
Get-S3Object -BucketName test-files -Key sample.txt
```

示例 3：此命令從存儲桶「測試文件」中檢索有關前綴為「sample」的所有項目的信息。

```
Get-S3Object -BucketName test-files -KeyPrefix sample
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListObjects式參考中的。

在 Amazon S3 主控台中使用資料夾整理物件

在 Amazon S3 中，儲存貯體與物件是主要資源，而且物件會存放在儲存貯體中。Amazon S3 採用單層式結構，而不是您在檔案系統中看到的階層結構。但為了簡化組織，Amazon S3 主控台支援將資料夾概念作為分組物件的方法。控制台透過使用分組物件的共享名稱前置詞來完成此操作。換句話說，分組物件的名稱使用常見的字串作為字首。此常見字串或共用字首是資料夾名稱。物件名稱也稱為金鑰。

例如，您可以在主控台中建立名稱為 photos 的資料夾，並在其中儲存名稱為 myphoto.jpg 的物件。物件接著會與金鑰名稱 photos/myphoto.jpg 一起存放，而 photos/ 是字首。

以下是其他兩個範例：

- 如果您的儲存貯體中有三個物件 (logs/date1.txt、logs/date2.txt 與 logs/date3.txt)，則主控台會顯示名為 logs 的資料夾。如果您在主控台中開啟該資料夾，則會看到三個物件：date1.txt、date2.txt 與 date3.txt。
- 如果您有名為 photos/2017/example.jpg 的物件，則主控台會顯示名為 photos 的資料夾，其中包含資料夾 2017。資料夾 2017 將包含物件 example.jpg。

資料夾內可以有資料夾，但儲存貯體內不可以有儲存貯體。您可以直接將物件上傳並複製至資料夾。您可以建立、刪除資料夾並將其設為公開，但無法對其重新命名。物件可以從某個資料夾複製至另一個資料夾。

> **Important**
>
> 當您在 Amazon S3 中建立資料夾時，S3 會建立一個 0 位元組的物件，其索引鍵設定為您提供的資料夾名稱。例如，如果您在儲存貯體中建立名為 photos 的資料夾時，Amazon S3 主控台會建立一個帶有索引鍵 photos/ 的 0 位元組物件。主控台建立此物件以支援資料夾的想法。
>
> Amazon S3 主控台會將索引鍵名稱中以正斜線 (/) 字元做為最後一個 (結尾) 字元的所有物件都視為資料夾 (例如 examplekeyname/)。您無法使用 Amazon S3 主控台來上傳有索引鍵名稱結尾為 / 字元的物件。不過，您可以使用 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 REST API，上傳以 Amazon S3 API 尾隨/命名的物件。
>
> 名稱結尾是 / 的物件會顯示為 Amazon S3 主控台中的資料夾。Amazon S3 主控台不會顯示這類物件的內容與中繼資料。使用主控台複製名稱結尾為 / 的物件時，會在目的地位置中建立新的資料夾，但不會複製物件的資料與中繼資料。

主題

建立資料夾

本節說明如何使用 Amazon S3 主控台建立資料夾。

> **Important**
>
> 如果儲存貯體政策會防止使用者在沒有標籤、中繼資料或存取控制清單 (ACL) 承授者的情況下，將物件上傳至此儲存貯體，您就無法使用下列程序建立資料夾。您必須改為上傳空的資料夾，並在上傳組態中指定下列設定。

若要建立資料夾

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇要建立資料夾的儲存貯體名稱。
4. 如果您的儲存貯體政策阻止在未加密的情況下將物件上傳到此儲存貯體，則必須選擇 Server-side encryption (伺服器端加密) 下的 Enable (啟用)。
5. 選擇 Create folder (建立資料夾)。
6. 輸入資料夾的名稱 (例如 **`favorite-pics`**)。然後選擇 Create Folder (建立資料夾)。

設定公有資料夾

除非您特別要求使用公有資料夾或儲存貯體，否則建議封鎖所有對 Amazon S3 資料夾和儲存貯體的公開存取。將資料夾設為公有時，所有網際網路使用者都可以檢視集合在該資料夾中的所有物件。

您可以在 Amazon S3 主控台中將資料夾設為公有。您也可以建立依字首限制資料存取的儲存貯體政策，將資料夾設為公有。如需詳細資訊，請參閱 適用於 Amazon S3 的 Identity and Access Management。

> **Warning**
>
> 在 Amazon S3 主控台中將資料夾設為公有後，就無法再次將其設為私有。您必須改為為公有資料夾中每個物件分別設定權限，才能改變由公有設為私有。如需詳細資訊，請參閱 設定 ACL。

主題

計算資料夾大小

本節說明如何使用 Amazon S3 主控台計算資料夾大小。

計算資料夾大小

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇存放資料夾的儲存貯體名稱。
4. 在 Objects (物件) 清單中，選取資料夾名稱旁的核取方塊。
5. 選擇 Actions (動作)，然後選擇 Calculate total size (計算總大小)。

> **Note**
>
> 在您離開頁面之後，資料夾資訊 (包括總大小) 將不再可用。如果您想再次查看，則必須再次計算總大小。

> **Important**
>
> - 當您對儲存貯體內指定的物件或資料夾使用 Calculate total size (計算總大小) 動作時，Amazon S3 會計算物件總數和總儲存大小。不過，未完成或進行中的分段上傳，以及先前或非目前的版本不會計入物件總數或總大小中。此動作只會針對存放在儲存貯體中之每個物件的目前或最新版本，計算物件總數和總大小。
>
> 例如，如果儲存貯體中有兩個物件版本，則 Amazon S3 中的儲存計算器只會將它們視為一個物件。因此，Amazon S3 主控台中計算的物件總數可能與 S3 儲存鏡頭中顯示的物件計數指標和 Amazon CloudWatch 指標報告的數量不同`NumberOfObjects`。同樣地，儲存總大小也可能與 S3 儲存鏡頭中顯示的總儲存量指標以及中顯示的`BucketSizeBytes`指標不同 CloudWatch。

- 如果計算大型資料夾總大小的時間過長，請考慮使用 Amazon S3 清查和 Amazon S3 Select 作為替代方案。首先，建立 S3 清查組態，在詳細目錄報告中包含大型資料夾每個物件的大小中繼資料。最多可能需要 48 小時的時間來提供首次 S3 清查報告。發佈詳細目錄報告時，請使用 S3 Select SUM 運算式查詢庫存報告，以彙總資料夾中物件的大小。如需詳細資訊，請參閱 使用 S3 主控台設定清查 及 SUM 範例。

刪除資料夾

本節說明如何使用 Amazon S3 主控台刪除 S3 儲存貯體中的資料夾。

如需 Amazon S3 功能與定價的相關資訊，請參閱 Amazon S3。

刪除 S3 儲存貯體中的資料夾

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇要刪除資料夾的儲存貯體名稱。
3. 在 Objects (物件) 清單中，選取要刪除的資料夾和物件旁邊的核取方塊。
4. 選擇刪除。
5. 在 Delete objects (刪除物件) 頁面中，確認畫面上已列出所選要刪除的資料夾名稱。
6. 在 Delete objects (刪除物件) 方塊中，輸入 **`delete`** 並選擇 Delete objects (刪除物件)。

> **Warning**
>
> 此動作會刪除所有指定的物件。刪除資料夾時，請等待刪除動作完成，然後再將新物件加入至資料夾。否則，系統也可能會刪除新物件。

在 Amazon S3 主控台中檢視物件概觀

您可以使用 Amazon S3 主控台來檢視物件的概觀。主控台在同一個位置提供了物件的所有重要資訊。

若要開啟物件的詳細資訊頁面

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

2. 在 Buckets (儲存貯體) 清單中，選擇包含該物件的儲存貯體名稱。
3. 在 Objects (物件) 清單中，選擇想要檢視其概觀的物件名稱。

   物件詳細資訊頁面隨即開啟。
4. 若要下載物件，選擇 Object actions (物件動作)，然後選擇 Download (下載)。若要將物件的路徑複製到剪貼簿，在 Object URL (物件 URL) 下選擇 URL。
5. 若儲存貯體已啟用版本控制，選擇 Versions (版本) 可列出物件版本。
   - 若要下載物件版本，選取某個版本 ID 旁邊的核取方塊，選擇 Actions (動作)，然後選擇 Download (下載)。
   - 若要刪除某個物件版本，選取某個版本 ID 旁邊的核取方塊，然後選擇 Delete (刪除)。

> **Important**
>
> 只有在刪除最新版 (目前版本) 的物件時，才能取消刪除物件。您無法取消刪除已刪除的舊版物件。

在 Amazon S3 主控台中檢視物件屬性

您可以使用 Amazon S3 主控台來檢視物件的屬性，包括儲存類別、加密設定、標籤和中繼資料。

檢視物件的屬性

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇包含該物件的儲存貯體名稱。
3. 在 Objects (物件) 清單中，選擇您要檢視其屬性的物件名稱。

   物件的 Object overview (物件概觀) 隨即開啟。您可以向下捲動以檢視物件屬性。
4. 在 Object overview (物件概觀) 頁面上，您可以設定物件的下列屬性。

> **Note**
>
> - 如果您變更儲存類別、加密或中繼資料，則會建立新物件來取代舊物件。如果啟用 S3 版本控制，則系統會建立物件的新版本，且現有物件會變成較舊的版本。變更屬性的角色也會成為新物件或 (物件版本) 的擁有者。
> - 如果您變更具有使用者定義標籤之物件的「儲存類別」、「加密」或「中繼資料」屬性，您必須擁有`s3:GetObjectTagging`權限。如果您要變更沒有使用者定義標籤但大小超過 16 MB 的物件的這些屬性，您也必須擁有`s3:GetObjectTagging`權限。
>
>   如果目的地儲存貯體原則拒絕該`s3:GetObjectTagging`動作，則會更新物件的這些屬性，但使用者定義的標籤會從物件中移除，而且您會收到錯誤訊息。

a. Storage class (儲存類別) – Amazon S3 中的每個物件都有相關聯的儲存類別。您選擇使用的儲存體方案取決於物件的存取頻率。S3 物件的預設儲存方案是 STANDARD。您可以選擇要在上傳物件時使用的儲存體方案。如需儲存體方案的詳細資訊，請參閱「使用 Amazon S3 儲存體方案」。

   若要在上傳物件之後變更儲存方案，請選擇 Storage class (儲存方案)。選擇您想要的儲存方案，然後選擇 Save (儲存)。

b. Server-side encryption settings (伺服器端加密設定) — 您可以使用伺服器端加密來加密 S3 物件。如需詳細資訊，請參閱 使用 AWS KMS (SSE-KMS) 指定伺服器端加密 或 使用 Amazon S3 受管金鑰 (SSE-S3) 指定伺服器端加密 。

c. Metadata (中繼資料) – Amazon S3 中的每個物件都有名稱/值對代表其中繼資料。如需將中繼資料新增至 S3 物件的資訊，請參閱「在 Amazon S3 主控台中編輯物件中繼資料」。

d. Tags (標籤) — 您可以將標籤新增至 S3 物件來分類儲存體。如需詳細資訊，請參閱 使用標籤分類儲存空間。

e. 物件鎖定法律訴訟保留與保留 — 您可以防止物件遭到刪除。如需詳細資訊，請參閱 使用 S3 物件鎖定。

使用預先簽章的 URL

若要授予對 Amazon S3 中物件的有限時間存取權限，而不更新儲存貯體政策，您可以使用預先簽章 URL。您可以在瀏覽器中輸入預先簽章的 URL，或由程式用來下載物件。預先簽署的 URL 所使用的認證是產生 URL 之 AWS 使用者的認證。

您也可以使用預先簽章的 URL，允許某人將特定物件上傳到您的 Amazon S3 儲存貯體。這允許上傳，而不需要另一方擁有 AWS 安全認證或權限。如果儲存貯體中已具備預先簽章 URL 中指定之相同金鑰的物件，則 Amazon S3 會使用上傳的物件來取代現有物件。

您可以多次使用此預先簽章 URL，直到到期日期和時間為止。

當您建立預先簽章 URL 時，必須提供安全憑證，然後指定下列項目：

- Amazon S3 儲存貯體
- 物件金鑰 (如果下載，則此物件將位於 Amazon S3 儲存貯體中，如果上傳，則這是要上傳的檔案名稱)
- HTTP 方法 (GET 用於下載物件，或 PUT 用於上傳)
- 到期時間間隔

目前，Amazon S3 預先簽章 URL 不支援在上傳物件時使用下列資料完整性檢查總和演算法 (CRC32、CRC32C、SHA-1、SHA-256)。若要在上傳物件後驗證其完整性，您可以在透過預先簽章 URL 上傳物件時，提供物件的 MD5 摘要。如需有關物件完整性的詳細資訊，請參閱 檢查物件完整性。

主題

誰可以建立預先簽章的 URL

任何具備有效安全憑證的使用者，均可建立預先簽章的 URL。但為了讓某人能順利存取物件，預先簽章的 URL 必須由有權執行預先簽章的 URL 做為基礎之操作的人員來建立。

以下是您可用以建立預先簽章 URL 的憑證類型：

- IAM 執行個體設定檔 - 有效期限最長 6 小時。
- AWS Security Token Service - 有效期在使用長期安全憑證簽署時最長為 36 小時，或為暫時憑證的持續時間 (以先到者為準)。

- IAM 使用者 — 當您使用 AWS 簽名版本 4 時，有效期最長為 7 天。

  若要建立有效期限最長 7 天的預先簽章 URL，請先將 IAM 使用者憑證 (存取金鑰和私密金鑰) 委派給您用於建立預先簽章 URL 的方法。

> **Note**
>
> 如果使用暫時憑證建立了預先簽章的 URL，則 URL 會在憑證過期時過期。一般而言，預先簽署的 URL 會在您用來建立該 URL 的認證撤銷、刪除或停用時過期。即使 URL 是以較晚的到期時間建立也一樣。有關臨時安全登入資料生命週期，請參閱 IAM 使用者指南中的比較 AWS STS API 操作。

預先簽章網址的到期時間

預先簽章的 URL 在產生 URL 時指定的期間內會保持有效。如果您使用 Amazon S3 主控台建立預先簽章的 URL，到期時間可以設定在 1 分鐘到 12 小時之間。如果您使用 AWS CLI 或 AWS SDK，到期時間最多可設定為 7 天。

如果您使用暫時權杖建立預先簽署的 URL，則 URL 會在權杖到期時過期。一般而言，預先簽署的 URL 會在您用來建立該 URL 的認證撤銷、刪除或停用時過期。即使 URL 是以較晚的到期時間建立也一樣。如需有關您使用的認證如何影響到期時間的詳細資訊，請參閱 誰可以建立預先簽章的 URL。

Amazon S3 會在 HTTP 請求時，檢查已簽署的 URL 中的過期日期和時間。例如，如果用戶端在到期前一刻才開始下載大型檔案，則即使在下載期間過期了，下載也會繼續。然而，如果連線中斷並且用戶端在到期時間過後嘗試重新啟動下載，則下載會失敗。

限制預先簽章的 URL 功能

預先簽章的 URL 的功能，受到建立它的使用者許可所限制。實質上，預先簽章的 URL 是一種承載符記，可為擁有這些網址的客戶授與存取權。因此，我們建議您妥善保護它們。以下幾種方法可供您用來限制預先簽章的 URL 使用。

AWS 簽名版本 4

若要在使用 AWS 第 4 版簽署程序 (SigV4) 驗證預先簽章 URL 請求時強制執行特定行為，您可以在儲存貯體政策和存取點政策中使用條件金鑰。例如，下列儲存貯體政策，使用 `s3:signatureAge` 條件來拒絕任何 *`example-s3-bucket1`* 儲存貯體中物件上的 Amazon S3 預先簽章 URL 請求 (如果簽章超過 10 分鐘)。若要使用此範例，請以您自己的資訊取代 *`user input placeholders`*。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Deny a presigned URL request if the signature is more than 10 min
 old",
         "Effect": "Deny",
         "Principal": {"AWS":"*"},
         "Action": "s3:*",
         "Resource": "arn:aws:s3:::example-s3-bucket1/*",
         "Condition": {
            "NumericGreaterThan": {
               "s3:signatureAge": 600000
            }
         }
      }
   ]
}
```

如需政策金鑰相關 AWS 簽章版本 4 的詳細資訊，請參閱 Amazon 簡單儲存服務 API 參考中的AWS 簽章版本 4 身份驗證。

網路路徑限制

如果您想要限制使用預先簽署的 URL 和所有 Amazon S3 存取特定網路路徑，您可以撰寫 AWS Identity and Access Management (IAM) 政策。您可以在進行呼叫的 IAM 主體、Amazon S3 儲存貯體，或兩者上設定政策。

IAM 主體的網路路徑限制需要這些憑證的使用者從指定的網路發出請求。儲存貯體或存取點上的限制要求所有對該資源的請求都來自指定網路。這些限制也適用於預先簽章的 URL 案例之外。

您使用的 IAM 全域條件金鑰取決於端點類型。如果您正在使用 Amazon S3 的公有端點，請使用 `aws:SourceIp`。如果您正在使用虛擬私有雲端 (VPC) 端點到 Amazon S3，請使用 `aws:SourceVpc` 或 `aws:SourceVpce`。

下列 IAM 政策聲明要求主體 AWS 只能從指定的網路範圍存取。由於此政策聲明，所有存取均必須源自該範圍。這包含某人使用 Amazon S3 預先簽章 URL 的情況。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
{
    "Sid": "NetworkRestrictionForIAMPrincipal",
    "Effect": "Deny",
```

```
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "NotIpAddressIfExists": {"aws:SourceIp": "IP-address-range"},
        "BoolIfExists": {"aws:ViaAWSService": "false"}
    }
}
```

如需使用aws:SourceIp AWS 全域條件金鑰將 Amazon S3 儲存貯體存取限制在特定網路範圍內的其他儲存貯體政策範例，請參閱根據特定 IP 地位址管理存取。

使用預先簽章的 URL 來共用物件

根據預設，所有 Amazon S3 物件皆為私有，只有物件擁有者才具有存取這些物件的許可。不過，物件擁有者可以透過建立預先簽章的 URL 與其他人共用物件。預先簽章的 URL 會使用安全認證授與下載物件的時間限制許可。URL 可以在瀏覽器中輸入，也可以由程式用來下載物件。預先簽署的 URL 所使用的認證是產生 URL 之 AWS 使用者的認證。

如需預先簽章的 URL 一般資訊，請參閱 使用預先簽章的 URL。

您可以使用 Amazon S3 主控台、適用於視覺工作室的 AWS 資源管理器 (Windows) 或 AWS Toolkit for Visual Studio Code，建立預先簽署的 URL 來共用物件，而不需要撰寫任何程式碼。您也可以使用 AWS Command Line Interface (AWS CLI) 或 AWS SDK，以程式設計方式產生預先簽署的 URL。

使用 S3 主控台

您可以執行以下步驟，使用 Amazon S3 主控台產生預先簽章 URL 以共用物件。當使用主控台時，預先簽章 URL 的最長過期時間為自建立時間起算 12 小時。

使用 Amazon S3 主控台產生預先簽章的 URL

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇儲存貯體名稱，該儲存貯體包含您要為其建立預先簽章 URL 的物件。
4. 在 Objects (物件) 清單中，選取要為其建立預先簽章 URL 的物件。
5. 在物件動作選單中，選擇使用預先簽章的 URL 來共用。
6. 指定預先簽章 URL 的有效期限。

7. 選擇 Create presigned URL (建立預先簽章的 URL)。
8. 出現確認提示時，URL 會自動複製到剪貼簿。如果您需要再次複製預先簽章的 URL，您將會看到一個複製按鈕。

使用 AWS CLI

下列範例 AWS CLI 命令會產生用於從 Amazon S3 儲存貯體共用物件的預先簽署 URL。使用時 AWS CLI，預先簽署 URL 的到期時間上限為建立之日起 7 天。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
aws s3 presign s3://example-s3-bucket1/mydoc.txt --expires-in 604800
```

> **Note**
>
> 對於 2019 年 3 月 20 日之後 AWS 區域 推出的所有內容，您需要指定endpoint-url和AWS ##請求。如需所有 Amazon S3 區域和端點的清單，請參閱《AWS 一般參考》中的區域與端點。

```
aws s3 presign s3://example-s3-bucket1/mydoc.txt --expires-in 604800 --region af-south-1 --endpoint-url https://s3.af-south-1.amazonaws.com
```

如需詳細資訊，請參閱 AWS CLI 命令參考中的 presign。

使用 AWS 軟體開發套件

如需使用 AWS SDK 產生用於共用物件的預先簽署 URL 的範例，請參閱使用開發套件為 Amazon S3 建立預先簽署的 URL。 AWS

當您使用 AWS SDK 產生預先簽署的 URL 時，到期時間上限為建立之日起 7 天。

> **Note**
>
> 對於 2019 年 3 月 20 日之後 AWS 區域 推出的所有內容，您需要指定endpoint-url和AWS ##請求。如需所有 Amazon S3 區域和端點的清單，請參閱《AWS 一般參考》中的區域與端點。

> **Note**
>
> 使用 AWS SDK 時，「標記」屬性必須是標頭，而不是查詢參數。所有其他屬性都可作為預先簽署 URL 的參數傳遞。

使用 AWS Toolkit for Visual Studio （視窗）

> **Note**
>
> 目前， AWS Toolkit for Visual Studio 不支援適用於 Mac 的視覺工作室。

1. AWS Toolkit for Visual Studio 使用下列指示，安裝和設定 Toolkit for Visual Studio AWS Toolkit for Visual Studio 者指南中進行安裝。
2. Connect 用「 AWS 使用AWS Toolkit for Visual Studio 者指南」 AWS中的下列步驟連線到。
3. 在標示為「AWS 檔案總管」的左側面板中，按兩下包含物件的值區。
4. 在您希望產生預先簽署 URL 的物件上按一下滑鼠右鍵，然後選取「建立預先簽署的 URL...」。 。
5. 在彈出式視窗中，設定預先簽署 URL 的到期日期和時間。
6. 物件索引鍵應根據您選取的物件預先填入。
7. 選擇 GET 以指定將使用此預先簽章的 URL 來下載物件。
8. 選擇產生按鈕。
9. 若要複製剪貼簿連結，請選擇複製。
10. 若要使用產生的預先簽署 URL，請將 URL 貼到任何瀏覽器中。

使用 AWS Toolkit for Visual Studio Code

如果您使用的是 Visual Studio Code，也可以使用 AWS Toolkit for Visual Studio Code產生預先簽章的物件 URL，而無須撰寫任何程式碼。如需一般資訊，請參閱《AWS Toolkit for Visual Studio Code 使用者指南》中的 AWS Toolkit for Visual Studio Code。

如需有關如何安裝的指示 AWS Toolkit for Visual Studio Code，請參閱《使用指南》 AWS Toolkit for Visual Studio Code中的AWS Toolkit for Visual Studio Code 〈安裝〉。

1. Connect 用「 AWS 使用AWS Toolkit for Visual Studio Code 者指南」 AWS Toolkit for Visual Studio Code中的下列步驟連線到。

2. 選擇左側面板中的 AWS 標誌代碼。
3. 在 EXPLORER 下，選取 S3。
4. 選擇儲存貯體和檔案，然後開啟 (按一下滑鼠右鍵) 內容功能表。
5. 選擇產生預先簽章的 URL，然後設定到期時間 (以分鐘為單位)。
6. 按 Enter 鍵，預先簽章的 URL 就會複製到您的剪貼簿。

使用預先簽章的 URL 上傳物件

您可以使用預先簽章的 URL，允許某人將物件上傳到您的 Amazon S3 儲存貯體。使用預先簽署的 URL 將允許上傳，而不需要另一方擁有 AWS 安全認證或權限。預先簽章的 URL 受到建立它的使用者許可所限制。也就是說，如果您收到上傳物件的預先簽章 URL，則只有該 URL 建立者具有上傳該物件的必要許可時，您才能上傳物件。

當有人使用 URL 上傳物件時，Amazon S3 會於指定儲存貯體建立物件。如果儲存貯體中已具備您在預先簽章 URL 中指定之相同金鑰的物件，則 Amazon S3 會使用上傳的物件來取代現有的物件。上傳後，儲存貯體擁有者將擁有該物件。

如需預先簽章的 URL 一般資訊，請參閱 使用預先簽章的 URL。

您可使用 AWS Explorer for Visual Studio 建立上傳物件的預先簽章 URL，而無須撰寫任何程式碼。您可以使用 AWS SDK，透過編寫程式的方式產生預先簽章的 URL。

使用 AWS Toolkit for Visual Studio （視窗）

> **Note**
>
> 目前， AWS Toolkit for Visual Studio 不支援適用於 Mac 的視覺工作室。

1. AWS Toolkit for Visual Studio 使用下列指示，安裝和設定 Toolkit for Visual Studio AWS Toolkit for Visual Studio 者指南中進行安裝。
2. Connect 用「 AWS 使用AWS Toolkit for Visual Studio 者指南」 AWS中的下列步驟連線到。
3. 在標示為 AWS Explorer 的左側面板中，以滑鼠右鍵按一下要將物件上傳至的值區。
4. 選擇建立預先簽署的 URL... 。
5. 在彈出式視窗中，設定預先簽署 URL 的到期日期和時間。

6. 對於「物件索引鍵」，設定要上載的檔案名稱。您要上傳的檔案必須與此名稱完全相符。如果儲存貯體中已存在具有相同物件金鑰的物件，Amazon S3 會以新上傳的物件取代現有物件。
7. 選擇 PUT 以指定將使用此預先簽章的 URL 來上傳物件。
8. 選擇產生按鈕。
9. 若要複製剪貼簿連結，請選擇複製。
10. 若要使用此 URL，您可以傳送包含 `curl` 命令的 PUT 請求。包含檔案的完整路徑和預先簽署的 URL 本身。

```
curl -X PUT -T "/path/to/file" "presigned URL"
```

使用 AWS 軟體開發套件

如需使用 AWS SDK 產生用於上傳物件的預先簽署 URL 的範例，請參閱使用開發套件為 Amazon S3 建立預先簽署的 URL。 AWS

當您使用 AWS SDK 產生預先簽署的 URL 時，到期時間上限為建立之日起 7 天。

> **Note**
>
> 對於 2019 年 3 月 20 日之後 AWS 區域 推出的所有內容，您需要指定`endpoint-url`和`AWS ##`請求。如需所有 Amazon S3 區域和端點的清單，請參閱《AWS 一般參考》中的區域與端點。

使用 S3 Object Lambda 轉換物件

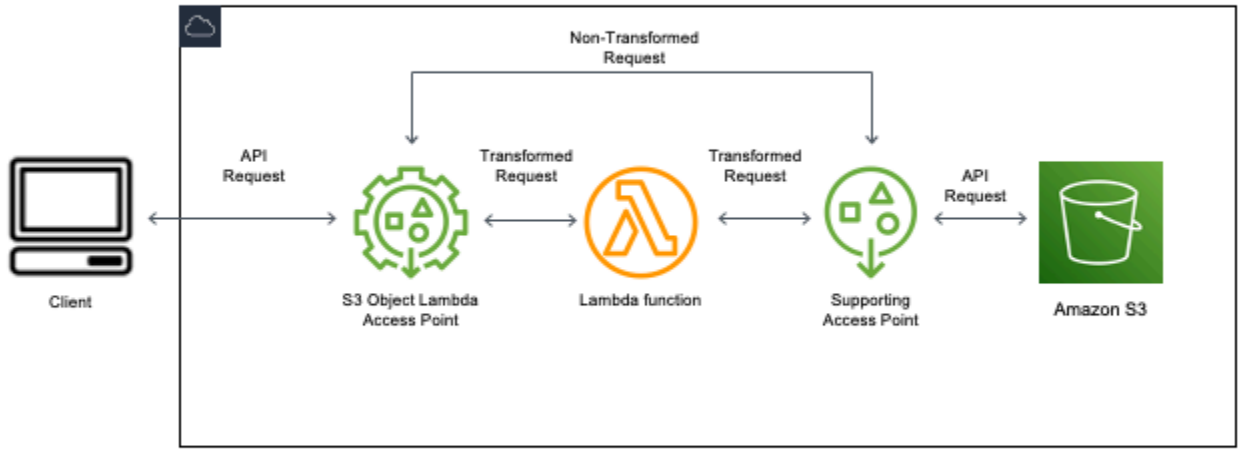
藉助 Amazon S3 Object Lambda，您可將自己的程式碼新增至 Amazon S3 `GET`、`LIST` 和 `HEAD` 請求，以便在資料傳回應用程式時對其做出修改和處理。您可以使用自訂程式碼修改 S3 `GET` 請求傳回的資料，以執行資料列篩選、動態調整浮水印影像大小、修訂機密資料以及更多動作。您也可以使用 S3 Object Lambda 修改 S3 `LIST` 請求的輸出，以建立儲存貯體中所有物件的自訂檢視，以及建立 S3 `HEAD` 請求來修改物件中繼資料 (例如物件名稱和大小)。您可以使用 S3 物件 Lambda 做為 Amazon CloudFront 分發的來源，為最終使用者量身打造資料，例如自動調整影像大小、將舊格式轉碼 (例如從 JPEG 轉換為 WebP)，或剝離中繼資料。如需詳細資訊，請參閱部 AWS 落格文章將 Amazon S3 物件 Lambda 與 Amazon 搭配使用 CloudFront。由 AWS Lambda 函數提供支援，您的程式碼會在完全受管理的基礎設施上執行 AWS。使用 S3 Object Lambda 可減少建立和存放資料衍生副本或執行代理的需求，並且全程無需變更您的應用程式。

S3 Object Lambda 的運作方式

S3 物件 Lambda 使用 AWS Lambda 函數來自動處理標準 S3 或GETLIST請HEAD求的輸出。 AWS Lambda 是一種無伺服器運算服務，可執行客戶定義的程式碼，而不需管理基礎運算資源。您可以編寫和執行自己的自訂 Lambda 函數，從而根據您的特定使用案例，量身定製資料轉換。

在設定 Lambda 函數之後，您可將其連接至 S3 Object Lambda 服務端點，稱為 Object Lambda 存取點。Object Lambda 存取點使用標準 S3 存取點 (稱為支援存取點) 來存取 Amazon S3。

當您將請求傳送到 Object Lambda 存取點時，Amazon S3 會自動呼叫您的 Lambda 函數。使用 S3 GET、LIST 或 HEAD 請求透過 Object Lambda 存取點擷取的任何資料都會將一個轉換的結果傳回給應用程式。所有其他請求都會正常處理，如下圖所示。



本節中的主題描述了如何使用 S3 Object Lambda。

主題

- 建立 Object Lambda 存取點
- 使用 Amazon S3 Object Lambda 存取點
- S3 Object Lambda 存取點的安全考量

- 撰寫 S3 Object Lambda 存取點的 Lambda 函數
- 使用 AWS 內 Lambda 函數
- S3 Object Lambda 的最佳實務和指導方針
- S3 Object Lambda 教學課程
- 偵錯 S3 Object Lambda

建立 Object Lambda 存取點

一個 Object Lambda 存取點只與一個標準存取點相關聯，因此只會與一個 Amazon S3 儲存貯體相關聯。若要建立 Object Lambda 存取點，您需要下列資源：

- Amazon S3 儲存貯體。如需建立儲存貯體的資訊，請參閱 the section called “建立儲存貯體”。
- 標準 S3 存取點 當您使用 Object Lambda 存取點時，此標準存取點稱為支援存取點。如需建立標準存取點的相關資訊，請參閱 the section called “建立存取點”。
- 一個 AWS Lambda 函數。您可以建立自己的 Lambda 函數，也可以使用預先建置的函數。如需建立 Lambda 函數的詳細資訊，請參閱 the section called “撰寫 Lambda 函數”。如需預先建置函數的詳細資訊，請參閱使用 AWS 內 Lambda 函數。
- (選擇性) AWS Identity and Access Management (IAM) 政策。Amazon S3 存取點支援 IAM 資源政策，可讓您依資源、使用者或其他條件控制對存取點的使用。如需建立這些政策的詳細資訊，請參閱 the section called “設定 IAM 政策”。

下列各節描述如何建立 Object Lambda 存取點，方法為使用：

- 該 AWS Management Console
- 該 AWS Command Line Interface （ AWS CLI ）
- 一個 AWS CloudFormation 模板
- 該 AWS Cloud Development Kit (AWS CDK)

如需如何使用 REST API 建立 Object Lambda 存取點的資訊，請參閱《Amazon Simple Storage Service API 參考》中的 `CreateAccessPointForObjectLambda`。

建立 Object Lambda 存取點

使用下列其中一個程序建立 Object Lambda 存取點。

使用 S3 主控台

使用主控台建立 Object Lambda 存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在導覽列中，選擇目前顯示的名稱 AWS 區域。接下來，選擇您要切換到的區域。
3. 在左側的導覽窗格中，選擇 Object Lambda Access Points (Object Lambda 存取點)。
4. 在 Object Lambda Access Points (Object Lambda 存取點) 頁面上，選擇 Create Object Lambda access point (建立 Object Lambda 存取點)。
5. 對於 Object Lambda Access Point name (Object Lambda 存取點名稱)，請輸入您要用於存取點的名稱。

   與標準存取點一樣，也有命名 Object Lambda 存取點的規則。如需詳細資訊，請參閱 命名 Amazon S3 存取點的規則。
6. 對於 Supporting Access Point (支援存取點)，請輸入或瀏覽至您要使用的標準存取點。存取點必須與您要轉換 AWS 區域 的物件位於相同的位置。如需建立標準存取點的相關資訊，請參閱 the section called “建立存取點”。
7. 在轉換組態下，您可以新增一個函數，針對 Object Lambda 存取點轉換資料。執行以下任意一項：

   - 如果您的帳戶中已有 AWS Lambda 函數，您可以在叫用 Lambda 函數下選擇它。您可以在此處輸入 Lambda 函數的 Amazon 資源名稱 (ARN)， AWS 帳戶 或從下拉式功能表中選擇 Lambda 函數。
   - 如果您希望使用內置函數，請在 AWS 構建函數下AWS 選擇函數名稱，然後選擇創建 Lambda 函數。這會帶您前往 Lambda 主控台，您可以在其中將內建函數部署到 AWS 帳戶. 如需內建函數的詳細資訊，請參閱 使用 AWS 內 Lambda 函數。

   在 S3 APIs (S3 API) 下，選擇一或多個要叫用的 API 操作。對於每個選取的 API，您都必須指定一個要叫用的 Lambda 函數。
8. (選用) 在 Payload (承載) 下，新增您想要提供給 Lambda 函數作為輸入的 JSON 文字。您可以針對叫用相同 Lambda 函數的不同 Object Lambda 存取點以不同參數設定承載，藉此擴充 Lambda 函數的彈性。

> **Important**
>
> 使用 Object Lambda 存取點時，請確定承載不包含任何機密資訊。

9. (選用) 對於 Range and part number (範圍和組件編號)，如果想要處理具有範圍和組件編號標頭的 GET 和 HEAD 請求，您必須啟用此選項。啟用此選項會確認您的 Lambda 函數可以辨識並處理這些請求。如需範圍標頭和組件編號的詳細資訊，請參閱 使用 Range 和 partNumber 標頭。
10. (選用) 對於請求指標，請選擇啟用或停用，將 Amazon S3 監控新增至您的 Object Lambda 存取點。請求指標以標準 Amazon 費 CloudWatch率計費。
11. (選用) 在 Object Lambda Access Point policy (Object Lambda 存取點政策) 下，設定資源政策。資源政策會授予所指定 Object Lambda 存取點的許可，並且可依資源、使用者或其他條件控制存取點的使用。如需有關 Object Lambda 存取點資源政策的詳細資訊，請參閱 設定 Object Lambda 存取點的 IAM 政策。
12. 在 Block Public Access settings for this Object Lambda Access Point (此 Object Lambda 存取點的封鎖公開存取設定)下，選取要套用的封鎖公開存取設定。根據預設，新 Object Lambda 存取點的所有封鎖公開存取設定都會啟用，建議將預設設定保持啟用狀態。Amazon S3 目前不支援在您建立了 Object Lambda 存取點之後，變更 Object Lambda 存取點的封鎖公開存取設定。

    如需使用 Amazon S3 封鎖公開存取的詳細資訊，請參閱 管理存取點的公開存取。
13. 選擇 Create Object Lambda Access Point (建立 Object Lambda 存取點)。

使用 AWS CLI

若要使用 AWS CloudFormation 範本建立物件 Lambda 存取點

> **Note**
>
> 若要使用下列命令，請以您自己的資訊取代 *user input placeholders*。

1. `s3objectlambda_deployment_package.zip`在 S3 物件 Lambda 預設組態下載 AWS Lambda 函數部署套件。
2. 執行下列 `put-object` 命令，將套件上傳至 Amazon S3 儲存貯體。

```
aws s3api put-object --bucket Amazon S3 bucket name --key
 s3objectlambda_deployment_package.zip --body release/
s3objectlambda_deployment_package.zip
```

3. s3objectlambda_defaultconfig.yaml在 S3 物件 Lambda 預設組態下載 AWS CloudFormation 範本。
4. 執行下列 deploy 命令，將範本部署到您的 AWS 帳戶。

```
aws cloudformation deploy --template-file s3objectlambda_defaultconfig.yaml \
 --stack-name AWS CloudFormation stack name \
 --parameter-overrides ObjectLambdaAccessPointName=Object Lambda Access Point name
 \
  SupportingAccessPointName=Amazon S3 access point S3BucketName=Amazon S3 bucket \
  LambdaFunctionS3BucketName=Amazon S3 bucket containing your Lambda package \
  LambdaFunctionS3Key=Lambda object key LambdaFunctionS3ObjectVersion=Lambda object
 version \
  LambdaFunctionRuntime=Lambda function runtime --capabilities capability_IAM
```

您可以設定此 AWS CloudFormation 範本來呼叫GETHEAD、和 LIST API 作業的 Lambda。如需修改範本預設組態的詳細資訊，請參閱 the section called “使用自動化 S3 物件 Lambda 設定 AWS CloudFormation”。

若要使用建立物件 Lambda 存取點 AWS CLI

> **Note**
>
> 若要使用下列命令，請以您自己的資訊取代 *user input placeholders*。

下列範例會為帳戶 *111122223333* 中的儲存貯體 *DOC-EXAMPLE-BUCKET1* 建立名為 *my-object-lambda-ap* 的 Object Lambda 存取點。此範例假設已建立名為 *example-ap* 的標準存取點。若要取得有關建立標準存取點的資訊，請參閱 the section called “建立存取點”。

這個例子使用 AWS 預構建的函數decompress。如需預先建置函數的詳細資訊，請參閱the section called “使用 AWS 內置函數”。

1. 建立儲存貯體。在此範例中，我們將使用 *DOC-EXAMPLE-BUCKET1*。如需建立儲存貯體的資訊，請參閱 the section called “建立儲存貯體”。

2. 建立標準的存取點並將其連接到您的儲存貯體。在此範例中，我們將使用 *example-ap*。如需建立標準存取點的相關資訊，請參閱 the section called “建立存取點”。
3. 執行以下任意一項：
   - 在您的帳戶中建立您要用來轉換 Amazon S3 物件的 Lambda 函數。如需建立 Lambda 函數的詳細資訊，請參閱 the section called “撰寫 Lambda 函數”。若要搭配使用您的自訂函數 AWS CLI，請參閱AWS Lambda 開發人員指南 AWS CLI中的搭配使用 Lambda。
   - 使用 AWS 預先建置的 Lambda 函數。如需預先建置函數的詳細資訊，請參閱使用 AWS 內 Lambda 函數。
4. 建立名為 `my-olap-configuration.json` 的 JSON 組態檔案。在此組態中，請為您在先前步驟中建立的 Lambda 函數提供支援存取點和 Amazon Resource Name (ARN)，或為您正在使用的預建函數提供 ARN。

   Example

```
{
    "SupportingAccessPoint" : "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
    "TransformationConfigurations": [{
        "Actions" : ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
        "ContentTransformation" : {
            "AwsLambda": {
                "FunctionPayload" : "{\"compressionType\":\"gzip\"}",
                "FunctionArn" : "arn:aws:lambda:us-east-1:111122223333:function/
compress"
            }
        }
    }]
}
```

5. 執行 `create-access-point-for-object-lambda` 命令以建立 Object Lambda 存取點。

```
aws s3control create-access-point-for-object-lambda --account-id 111122223333 --
name my-object-lambda-ap --configuration file://my-olap-configuration.json
```

6. (選用) 建立名為 `my-olap-policy.json` 的 JSON 政策檔案。

   新增 Object Lambda 存取點資源政策可依資源、使用者或其他條件控制存取點的使用。此資源政策會將帳戶 *444455556666* 的 `GetObject` 許可授予指定的 Object Lambda 存取點。

Example

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "Grant account 444455556666 GetObject access",
            "Effect": "Allow",
            "Action": "s3-object-lambda:GetObject",
            "Principal": {
                "AWS": "arn:aws:iam::444455556666:root"
            },
            "Resource": "your-object-lambda-access-point-arn"
        }
    ]
}
```

7. (選用) 執行 `put-access-point-policy-for-object-lambda` 命令以設定您的資源政策。

```
aws s3control put-access-point-policy-for-object-lambda --account-id 111122223333
 --name my-object-lambda-ap --policy file://my-olap-policy.json
```

8. (選用) 指定承載。

有效負載是可選的 JSON，您可以將其作為輸入提供給 AWS Lambda 函數。您可以針對叫用相同 Lambda 函數的不同 Object Lambda 存取點以不同參數設定承載，藉此擴充 Lambda 函數的彈性。

下列 Object Lambda 存取點組態顯示具有兩個參數的承載。

```
{
 "SupportingAccessPoint": "AccessPointArn",
 "CloudWatchMetricsEnabled": false,
 "TransformationConfigurations": [{
  "Actions": ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
  "ContentTransformation": {
   "AwsLambda": {
    "FunctionArn": "FunctionArn",
    "FunctionPayload": "{\"res-x\": \"100\",\"res-y\": \"100\"}"
   }
  }
```

```
 }]
}
```

下列 Object Lambda 存取點組態顯示具有一個參數、且已啟用 GetObject-Range、GetObject-PartNumber、HeadObject-Range 和 HeadObject-PartNumber 的承載。

```
{
    "SupportingAccessPoint":"AccessPointArn",
    "CloudWatchMetricsEnabled": false,
    "AllowedFeatures": ["GetObject-Range", "GetObject-PartNumber", "HeadObject-
Range", "HeadObject-PartNumber"],
    "TransformationConfigurations": [{
        "Action": ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
        "ContentTransformation": {
            "AwsLambda": {
                "FunctionArn":"FunctionArn",
                "FunctionPayload": "{\"compression-amount\": \"5\"}"
            }
        }
    }]
}
```

Important

使用 Object Lambda 存取點時，請確定承載不包含任何機密資訊。

使用主 AWS CloudFormation 控台和範本

您可以使用 Amazon S3 提供的預設組態建立 Object Lambda 存取點。您可以從GitHub 儲存庫下載 AWS CloudFormation 範本和 Lambda 函數原始程式碼，並部署這些資源以設定功能性物件 Lambda 存取點。

如需修改 AWS CloudFormation 範本預設組態的資訊，請參閱the section called “使用自動化 S3 物件 Lambda 設定 AWS CloudFormation”。

如 AWS CloudFormation 需使用不使用範本來設定物件 Lambda 存取點的相關資訊，請參閱使用AWS CloudFormation 者指南*AWS::S3ObjectLambda::AccessPoint*中的。

上傳 Lambda 函數部署套件

1. `s3objectlambda_deployment_package.zip`在 S3 物件 Lambda 預設組態下載 AWS Lambda 函數部署套件。
2. 將套件上傳至 Amazon S3 儲存貯體。

使用 AWS CloudFormation 主控台建立物件 Lambda 存取點

1. `s3objectlambda_defaultconfig.yaml`在 S3 物件 Lambda 預設組態下載 AWS CloudFormation 範本。
2. 登入 AWS 管理主控台，然後開啟 AWS CloudFormation 主控台，網址為 https://console.aws.amazon.com/cloudformation。
3. 執行以下任意一項：
   - 如果您從未使用 AWS CloudFormation 過，請在 AWS CloudFormation 首頁上選擇 [建立堆疊]。
   - 如果您 AWS CloudFormation 之前曾使用過，請在左側導覽窗格中選擇「堆疊」。選擇 Create stack (建立堆疊)，然後選擇 With new resources (standard) (使用新資源 (標準))。
4. 對於 Prepare template (準備範本)，請選擇 Template is ready (範本已準備就緒)。
5. 對於 Specify template (指定範本)，選擇 Upload a template file (上傳範本檔案) 並上傳 `s3objectlambda_defaultconfig.yaml`。
6. 選擇 Next (下一步)。
7. 在 Specify stack details (指定堆疊詳細資訊) 頁面上，輸入堆疊的名稱。
8. 在 Parameters (參數) 區段中，指定堆疊範本中定義的以下參數。
   a. 對於 CreateNewSupportingAccessPoint，請執行下列其中一個動作：
      - 如果您已有範本上傳所在之 S3 儲存貯體的支援存取點，請選擇 false。
      - 如果您要針對此儲存貯體建立新的存取點，請選擇 true。
   b. 對於 EnableCloudWatchMonitoring，請根據您是否要啟用 Amazon CloudWatch 請求指標和警示，選擇真或假。
   c. (選擇性) 針對 LambdaFunctionPayload，新增您要提供給 Lambda 函數做為輸入的 JSON 文字。您可以針對叫用相同 Lambda 函數的不同 Object Lambda 存取點以不同參數設定承載，藉此擴充 Lambda 函數的彈性。

> **Important**
>
> 使用 Object Lambda 存取點時，請確定承載不包含任何機密資訊。

d. 針對 LambdaFunctionRuntime，輸入您偏好的 Lambda 函數執行階段。可用選項為 `nodejs14.x`、`python3.9`、`java11`。

e. 對於 LambdaFunctionS3 BucketName，請輸入您上傳部署套件的 Amazon S3 儲存貯體名稱。

f. 對於 LambdaFunctionS3Key，請輸入您上傳部署套件的 Amazon S3 物件金鑰。

g. 對於 LambdaFunctionS3 ObjectVersion，請輸入您上傳部署套件的 Amazon S3 物件版本。

h. 在中 ObjectLambdaAccessPointName，輸入物件 Lambda 存取點的名稱。

i. 對於 S3 BucketName，請輸入將與您的物件 Lambda 存取點關聯的 Amazon S3 儲存貯體名稱。

j. 在中 SupportingAccessPointName，輸入支援存取點的名稱。

> **Note**
>
> 這是與您在上一步驟中所選擇 Amazon S3 儲存貯體相關聯的存取點。如果您的 Amazon S3 儲存貯體沒有任何關聯的存取點，可以透過選擇 true 為來設定範本為您建立一個CreateNewSupportingAccessPoint。

9. 選擇下一步。
10. 在 Configure stack options (設定堆疊選項) 頁面，選擇 Next (下一步)。

    如需此頁面上選用設定的詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的設定 AWS CloudFormation 堆疊選項。
11. 在 Review (檢閱) 頁面上，選擇 Create stack (建立堆疊)。

使用 AWS Cloud Development Kit (AWS CDK)

如需使用設定物件 Lambda 存取點的詳細資訊 AWS CDK，請參閱 AWS Cloud Development Kit (AWS CDK) API 參考中的`AWS::S3ObjectLambda`建構程式庫。

使用 CloudFormation範本自動化 S3 物件 Lambda 設定

您可以使用 AWS CloudFormation 範本快速建立 Amazon S3 物件 Lambda 存取點。 CloudFormation 範本會自動建立相關資源、設定 AWS Identity and Access Management (IAM) 角色，並設定可透過物件 Lambda 存取點自動處理請求的 AWS Lambda 函數。使用 CloudFormation 範本，您可以實作最佳做法、改善安全性狀態，以及減少手動程序所造成的錯誤。

此GitHub 存儲庫包含 CloudFormation 模板和 Lambda 函數源代碼。如需如何使用範本的指示，請參閱the section called “建立 Object Lambda 存取點”。

範本中提供的 Lambda 函數不會執行任何轉換。相反，它會依原狀從您的 S3 儲存貯體傳回您的物件。您可以複製函數，並新增自己的轉換程式碼，在資料傳回應用程式時對其進行修改和處理。如需修改函數的詳細資訊，請參閱 the section called “修改 Lambda 函數” 和 the section called “撰寫 Lambda 函數”。

修改範本

建立新的支援存取點

S3 Object Lambda 使用兩個存取點，一個 Object Lambda 存取點和一個標準 S3 存取點，這些稱為支援存取點。當您對 Object Lambda 存取點提出請求時，S3 會代表您叫用 Lambda，或將請求委派給支援的存取點 (視 S3 Object Lambda 組態而定)。您可以傳遞下列參數作為部署範本時 `aws cloudformation deploy` 命令的一部分，建立新的支援存取點。

```
CreateNewSupportingAccessPoint=true
```

設定函數承載

您可以透過部署範本時傳遞下列參數作為 `aws cloudformation deploy` 命令之一部分的方式，設定承載，以提供補充資料給 Lambda 函數。

```
LambdaFunctionPayload="format=json"
```

啟用 Amazon CloudWatch 監控

您可以在部署範本時，將下列參數做為`aws cloudformation deploy`命令的一部分傳遞來啟用 CloudWatch 監視。

```
EnableCloudWatchMonitoring=true
```

此參數可啟用 Amazon S3 請求指標的物件 Lambda 存取點，並建立兩個 CloudWatch警示來監控用戶端和伺服器端錯誤。

> **Note**
>
> Amazon CloudWatch 用量將產生額外費用。如需 Amazon S3 請求指標的詳細資訊，請參閱監控與記錄存取點。
> 如需定價詳細資訊，請參閱 CloudWatch 定價。

設定佈建並行

若要減少延遲，您可以編輯範本，為 Lambda 函數支援的 Object Lambda 存取點設定佈建並行，以便包含 `Resources` 下的下列行。

```
LambdaFunctionVersion:
      Type: AWS::Lambda::Version
      Properties:
        FunctionName: !Ref LambdaFunction
        ProvisionedConcurrencyConfig:
            ProvisionedConcurrentExecutions: Integer
```

> **Note**
>
> 您將需要針對佈建並行支付額外費用。如需佈建並行的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的管理 Lambda 佈建並行。
> 如需定價詳細資訊，請參閱 AWS Lambda 定價。

修改 Lambda 函數

變更 **`GetObject`** 請求的標頭值

根據預設，Lambda 函數會將所有標頭 (但 `Content-Length` 和 `ETag` 除外) 從預先簽章的 URL 請求轉送到 `GetObject` 用戶端。根據 Lambda 函數中的轉換程式碼，您可以選擇將新標頭值傳送至 `GetObject` 用戶端。

您可以更新 Lambda 函數，透過在 `WriteGetObjectResponse` API 操作中傳遞新標頭值來傳送這些值。

例如，如果 Lambda 函數將 Amazon S3 物件中的文字翻譯為不同的語言，則您可以在 Content-Language 標頭傳遞新值。您可以修改 writeResponse 函數來執行此操作，如下所示：

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest
    },
    ...headers,
    ContentLanguage: 'my-new-language'
  }).promise();
}
```

如需受支援標頭的完整清單，請參閱 Amazon Simple Storage Service API 參考中的 WriteGetObjectResponse。

傳回中繼資料標頭

您可以更新 Lambda 函數，透過在 WriteGetObjectResponse API 操作請求中傳遞新標頭值來傳送這些值。

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest,
      'my-new-header': 'my-new-value'
    },
```

```
    ...headers
  }).promise();
}
```

傳回新的狀態碼

您可以將自訂狀態碼傳回至 GetObject 用戶端，方法為在 WriteGetObjectResponse API 操作請求中傳遞該自訂狀態碼。

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest
    },
    ...headers,
    StatusCode: Integer
  }).promise();
}
```

如需受支援狀態碼的完整清單，請參閱 Amazon Simple Storage Service API 參考中的 WriteGetObjectResponse。

將 **Range** 和 **partNumber** 參數套用至來源物件

依預設， CloudFormation 範本建立的物件 Lambda 存取點可以處理Range和partNumber參數。Lambda 函數會將請求的範圍或組件編號套用至轉換物件。若要這樣做，函數必須下載整個物件並執行轉換。在某些情況下，您的轉換物件範圍可能會完全映射至您的來源物件範圍。這表示請求來源物件上的位元組範圍 A-B 並執行轉換，可能會產生與請求整個物件、執行轉換，以及傳回轉換物件上的位元組範圍 A-B 相同的結果。

在這種情況下，您可以變更 Lambda 函數實作，將範圍或組件編號直接套用至來源物件。此方法會減少整體函數延遲和所需的記憶體。如需詳細資訊，請參閱 the section called “使用 Range 和 partNumber 標頭”。

停用 `Range` 和 `partNumber` 處理

依預設， CloudFormation 範本建立的物件 Lambda 存取點可以處理`Range`和`partNumber`參數。如果不需要此行為，則您可以從範本中移除下列行來停用此行為：

```
AllowedFeatures:
  - GetObject-Range
  - GetObject-PartNumber
  - HeadObject-Range
  - HeadObject-PartNumber
```

轉換大型物件

預設情況下，Lambda 函數會先處理記憶體中的整個物件，然後才能開始將回應串流至 S3 Object Lambda。您可以在執行轉換時修改函數以串流回應。這樣做有助於降低轉換延遲並縮小 Lambda 函數記憶體大小。如需範例實作，請參閱串流壓縮內容範例。

使用 Amazon S3 Object Lambda 存取點

透過 Amazon S3 Object Lambda 存取點發出請求，與透過其他存取點發出請求的運作方式相同。如需如何透過存取點發出請求的詳細資訊，請參閱 使用存取點。您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 Amazon S3 REST API，透過物件 Lambda 存取點提出請求。

> **Important**
>
> Object Lambda 存取點的 Amazon Resource Name (ARN) 使用 `s3-object-lambda` 的服務名稱。因此，Object Lambda 存取點 ARN 以 `arn:aws::s3-object-lambda` 為開頭 (而不是 `arn:aws::s3`)，其會與其他存取點一起使用。

如何為 Object Lambda 存取點尋找 ARN

若要將物件 Lambda 存取點與 AWS CLI 或 AWS SDK 搭配使用，您需要知道物件 Lambda 存取點的 Amazon 資源名稱 (ARN)。下列範例示範如何使用 Amazon S3 主控台或 AWS CLI，來尋找 Object Lambda 存取點的 ARN。

使用 S3 主控台

使用主控台為 Object Lambda 存取點尋找 ARN

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側的導覽窗格中，選擇 Object Lambda Access Points (Object Lambda 存取點)。
3. 選擇要複製 ARN 的 Object Lambda 存取點旁的選項按鈕。
4. 選擇 Copy ARN (複製 ARN)。

使用 AWS CLI

若要尋找物件 Lambda 存取點的 ARN，請使用 AWS CLI

1. 若要擷取與 AWS 帳戶相關聯的 Object Lambda 存取點的清單，請執行以下命令。執行指令之前，請先*111122223333*以您的 ID 取代帳戶 AWS 帳戶 ID。

```
aws s3control list-access-points-for-object-lambda --account-id 111122223333
```

2. 檢閱命令輸出，尋找要使用的 Object Lambda 存取點 ARN。先前命令的輸出看起來應與下列範例類似。

```
{
    "ObjectLambdaAccessPointList": [
        {
            "Name": "my-object-lambda-ap",
            "ObjectLambdaAccessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/my-object-lambda-ap"
        },
        ...
    ]
}
```

如何針對您的 S3 儲存貯體 Object Lambda 存取點使用儲存貯體樣式別名

在您建立 Object Lambda 存取點時，Amazon S3 會自動為您的 Object Lambda 存取點產生唯一的別名。您可以針對存取點資料平面操作在請求中使用此別名，而不是使用 Amazon S3 儲存貯體名稱或 Object Lambda 存取點 Amazon Resource Name (ARN)。如需這些操作的清單，請參閱 存取點與 AWS 服務的相容性。

Object Lambda 存取點別名是在與 Amazon S3 儲存貯體相同的命名空間內建立的。此別名會自動產生且無法變更。針對現有的 Object Lambda 存取點，系統會自動指派別名以供使用。Object Lambda 存取點別名符合有效 Amazon S3 儲存貯體名稱的所有要求，並由下列部分組成：

Object Lambda Access Point name prefix-*metadata*--ol-s3

> **Note**
>
> Object Lambda 存取點別名會保留該 --ol-s3 尾碼，且該尾碼不能用於儲存貯體或 Object Lambda 存取點名稱。如需 Amazon S3 儲存貯體命名規則的詳細資訊，請參閱儲存貯體命名規則。

下列範例顯示了名稱為 *my-object-lambda-access-point* 之 Object Lambda 存取點的 ARN 和 Object Lambda 存取點別名。

- ARN – arn:aws:s3-object-lambda:*region*:*account-id*:accesspoint/*my-object-lambda-access-point*
- Object Lambda 存取點別名 – *my-object-lambda-acc-1a4n8yjrb3kda96f67zwrwiiuse1a*--ol-s3

使用 Object Lambda 存取點時，您可以使用 Object Lambda 存取點別名，而不需要大量變更程式碼。

刪除 Object Lambda 存取點時，Object Lambda 存取點別名會變成非作用中且未佈建狀態。

如何為 Object Lambda 存取點尋找別名

使用 S3 主控台

使用主控台為 Object Lambda 存取點尋找別名

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側的導覽窗格中，選擇 Object Lambda Access Points (Object Lambda 存取點)。
3. 針對您要使用的 Object Lambda 存取點，請複製 Object Lambda 存取點別名值。

使用 AWS CLI

建立 Object Lambda 存取點時，Amazon S3 會自動產生 Object Lambda 存取點別名，如下列範例命令所示。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。如需如何使用建立物件 Lambda 存取點的相關資訊 AWS CLI，請參閱若要使用建立物件 Lambda 存取點 AWS CLI。

```
aws s3control create-access-point-for-object-lambda --account-id 111122223333 --
name my-object-lambda-access-point --configuration file://my-olap-configuration.json
{
    "ObjectLambdaAccessPointArn": "arn:aws:s3:region:111122223333:accesspoint/my-
access-point",
    "Alias": {
        "Value": "my-object-lambda-acc-1a4n8yjrb3kda96f67zwrwiiuse1a--ol-s3",
        "Status": "READY"
    }
}
```

產生的 Object Lambda 存取點別名會有兩個欄位：

- `Value` 欄位是 Object Lambda 存取點的別名。
- `Status` 欄位是 Object Lambda 存取點別名的狀態。若狀態為 `PROVISIONING`，表示 Amazon S3 正在佈建 Object Lambda 存取點別名，且別名尚不可供使用。若狀態為 `READY`，則表示 Object Lambda 存取點別名已成功佈建並可供使用。

如需詳細了解 REST API 中的 `ObjectLambdaAccessPointAlias` 資料類型，請參閱《Amazon Simple Storage Service API 參考》中的 ObjectLambdaAccessPointAlias 和 CreateAccessPointForObjectLambda。

如何使用 Object Lambda 存取點別名

您可以針對 存取點與 AWS 服務的相容性 所列操作，使用 Object Lambda 存取點別名，而非 Amazon S3 儲存貯體名稱。

下列`get-bucket-location`命令 AWS CLI 範例會使用值區的存取點別名來傳回值區 AWS 區域 所在的位置。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3api get-bucket-location --bucket my-object-lambda-
acc-w7i37nq6xuzgax3jw3oqtifiusw2a--ol-s3

{
```

```
    "LocationConstraint": "us-west-2"
}
```

若請求中的 Object Lambda 存取點別名無效，則會傳回 `InvalidAccessPointAliasError` 錯誤碼。如需有關 `InvalidAccessPointAliasError` 的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的錯誤代碼清單。

Object Lambda 存取點別名的限制與存取點別名的限制相同。如需存取點別名限制的詳細資訊，請參閱 限制。

S3 Object Lambda 存取點的安全考量

使用 Amazon S3 物件 Lambda，您可以使用作為運算平台的規模和靈活性，在資料離開 Amazon S3 時對資料執行自訂轉換。 AWS Lambda S3 和 Lambda 依預設保持安全狀態，但若要維護安全，需要 Lambda 函數授權方的特別考量。S3 Object Lambda 要求所有存取都由經過驗證的主體 (無匿名訪問) 和透過 HTTPS 進行。

若要緩解安全風險，建議您採取下列操作：

- 將 Lambda 執行角色的範圍儘可能設定為最小的許可集。
- 儘可能確保您的 Lambda 函數透過提供的預先簽章 URL 存取 Amazon S3。

設定 IAM 政策

S3 存取點支援 AWS Identity and Access Management (IAM) 資源政策，可讓您依資源、使用者或其他條件控制存取點的使用。如需詳細資訊，請參閱 設定 Object Lambda 存取點的 IAM 政策。

加密行為

由於物件 Lambda 存取點同時使用 Amazon S3 AWS Lambda，因此加密行為存在差異。如需預設 S3 加密行為的詳細資訊，請參閱對 Amazon S3 儲存貯體設定預設伺服器端加密行為。

- 當您搭配 Object Lambda 存取點使用 S3 伺服器端加密時，物件會在傳送至 Lambda 之前進行解密。將物件傳送至 Lambda 之後，會以未加密方式處理物件 (若為 `GET` 或 `HEAD` 請求)。
- 為了防止金鑰遭到記錄，S3 會針對搭配客戶提供的金鑰 (SSE-C) 使用伺服器加密來加密的物件拒絕 `GET` 和 `HEAD` 請求。不過，如果 Lambda 函數可以存取用戶端提供的金鑰，則它仍然可能擷取這些物件。
- 搭配 Object Lambda 存取點使用 S3 用戶端加密時，請確定 Lambda 可以存取金鑰，以便其可以解密和重新加密物件。

存取點安全

S3 Object Lambda 使用兩個存取點，一個 Object Lambda 存取點和一個標準 S3 存取點，這些稱為支援存取點。當您對 Object Lambda 存取點提出請求時，S3 會代表您叫用 Lambda，或將請求委派給支援的存取點 (視 S3 Object Lambda 組態而定)。當針對請求叫用 Lambda 時，S3 會透過支援存取點代表您對物件產生預先簽章的 URL。叫用 Lambda 函數時，此函數會接收此 URL 作為輸入。

您可以將 Lambda 函數設定為使用此預先簽章 URL 來擷取原始物件，而不是直接叫用 S3。藉由使用此模型，您可以將更好的安全界限套用至您的物件。您可以透過 S3 儲存貯體或 S3 存取點，將直接物件存取限制為一組有限的 IAM 角色或使用者。此方法也可以保護 Lambda 函數免受混淆代理人問題的影響，其中具有與啟動程式不同許可的錯誤設定函數可能會允許或拒絕物件的存取。

Object Lambda 存取點公有存取點

S3 Object Lambda 不允許匿名或公有存取，因為 Amazon S3 必須授權您的身分，才能完成任何 S3 Object Lambda 請求。當透過 Object Lambda 存取點叫用請求時，您必須具有所設定 Lambda 函數的 `lambda:InvokeFunction` 許可。同樣，當透過 Object Lambda 存取點叫用其他 API 操作時，您必須具有必要的 `s3:*` 許可。

若沒有這些許可，叫用 Lambda 或委派給 S3 的請求將會失敗，並顯示 HTTP 403 (禁止) 錯誤。所有存取必須由經過身分驗證的委託人進行。如果需要公有存取，您可以使用 Lambda@Edge 作為可能的替代方案。如需詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的使用 Lambda @Edge 在邊緣進行自訂。

Object Lambda 存取點 IP 地址

這些`describe-managed-prefix-lists` 子網路支援閘道虛擬私有雲端 (VPC) 端點，並且與 VPC 端點的路由表相關。由於 Object Lambda 存取點不支援閘道 VPC，因此缺少其 IP 範圍。缺少的範圍屬於 Amazon S3，但閘道 VPC 端點不支援。如需有關DescribeManagedPrefixLists的AWS 詳 Amazon EC2 資訊*`describe-managed-prefix-lists`*，請參閱 AWS 一般參考.

設定 Object Lambda 存取點的 IAM 政策

Amazon S3 存取點支援 AWS Identity and Access Management (IAM) 資源政策，您可以使用這些政策按資源、使用者或其他條件控制存取點的使用。您可以透過 Object Lambda 存取點上的選用資源政策或支援存取點的資源政策來控制存取。如需 step-by-step 範例，請參閱教學課程：使用 S3 Object Lambda 轉換應用程式的資料和教學課程：使用 S3 Object Lambda 和 Amazon Comprehend 來偵測和編輯 PII 資料。

下列四個資源必須具有使用 Object Lambda 存取點的許可：

- IAM 身分，例如使用者或角色。如需 IAM 身分與最佳實務的詳細資訊，請參閱《IAM 使用者指南》中的 IAM 身分 (使用者、群組和角色)。
- 儲存貯體及其相關聯的標準存取點。當您使用 Object Lambda 存取點時，此標準存取點稱為支援存取點。
- Object Lambda 存取點。
- 該 AWS Lambda 功能。

> **Important**
>
> 儲存原則之前，請務必先解決安全性警告、錯誤、一般警告和建議 AWS Identity and Access Management Access Analyzer。IAM Access Analyzer 會比對 IAM 政策文法和最佳實務來執行政策檢查，以驗證您的政策。這些檢查會產生問題清單並提供可行的建議，協助您撰寫具有功能性且符合安全最佳實務的政策。
>
> 若要進一步了解如何使用 IAM Access Analyzer 驗證政策，請參閱《IAM 使用者指南》中的 IAM Access Analyzer 政策驗證。若要檢視 IAM Access Analyzer 傳回的警告、錯誤和建議清單，請參閱 IAM Access Analyzer 政策檢查參考。

下列政策範例假設您有下列資源：

- 含有下列 Amazon Resource Name (ARN) 的 Amazon S3 儲存貯體：

  `arn:aws:s3:::DOC-EXAMPLE-BUCKET1`
- 具有下列 ARN 的此儲存貯體上的 Amazon S3 Standard 存取點：

  `arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point`
- 具有下列 ARN 的 Object Lambda 存取點：

  `arn:aws:s3-object-lambda:us-east-1:111122223333:accesspoint/my-object-lambda-ap`
- 具有以下 ARN 的 AWS Lambda 函數：

  `arn:aws:lambda:us-east-1:111122223333:function:MyObjectLambdaFunction`

> **Note**
>
> 如果您使用帳戶中的 Lambda 函數，則必須在政策聲明中包含特定函數版本。在下列範例 ARN 中，版本由 *1* 表示：
> `arn:aws:lambda:us-east-1:111122223333:function:MyObjectLambdaFunction:1`
> Lambda 不支援將 IAM 政策新增至版本$LATEST。如需 Lambda 函數版本的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 Lambda 函數版本。

Example - 將存取控制委派給標準存取點的儲存貯體政策

下列 S3 儲存貯體政策範例會將儲存貯體的存取控制委派給儲存貯體的標準存取點。此政策允許完整存取儲存貯體擁有者帳戶所擁有的所有存取點。因此，對此儲存貯體的所有存取皆由連接至其存取點的政策所控制。使用者只能透過存取點從儲存貯體讀取，這表示只能透過存取點叫用操作。如需詳細資訊，請參閱 將存取控制委派給存取點。

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS":"account-ARN"},
        "Action" : "*",
        "Resource" : [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET1",
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"
        ],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Bucket owner's account
 ID" }
        }
    }]
}
```

Example — 授予使用者使用物件 Lambda 存取點所需權限的 IAM 政策

下列 IAM 政策許可使用者使用 Lambda 函數、標準存取點和 Object Lambda 存取點。

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "AllowLambdaInvocation",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:MyObjectLambdaFunction:1",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "s3-object-lambda.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "AllowStandardAccessPointAccess",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "s3-object-lambda.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "AllowObjectLambdaAccess",
      "Action": [
        "s3-object-lambda:Get*",
        "s3-object-lambda:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3-object-lambda:us-east-1:111122223333:accesspoint/my-
object-lambda-ap"
    }
  ]
```

```
}
```

啟用 Lambda 執行角色的許可

向物件 Lambda 存取點發出GET請求時，您的 Lambda 函數需要將資料傳送至 S3 物件 Lambda 存取點的權限。此許可是透過對您的 Lambda 函數的執行角色啟用 `s3-object-lambda:WriteGetObjectResponse` 許可來提供的。您可建立新的執行角色，或使用現有的角色進行更新。

> **Note**
>
> 只有當您提出 GET 請求時，您的函數才需要 `s3-object-lambda:WriteGetObjectResponse` 許可。

若要在 IAM 主控台中建立執行角色

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 選擇建立角色。
4. 在 Common use cases (一般使用案例) 下，選擇 Lambda。
5. 選擇下一步。
6. 在 [新增權限] 頁面上，搜尋受 AWS 管理的原則 `AmazonS3ObjectLambdaExecutionRolePolicy`，然後選取原則名稱旁邊的核取方塊。

   此政策應包含 `s3-object-lambda:WriteGetObjectResponse` 動作。
7. 選擇下一步。
8. 在 Name, review, and create (命名、檢閱和建立) 頁面上，針對 Role name (角色名稱) 輸入 **`s3-object-lambda-role`**。
9. (選用) 新增此角色的描述和標籤。
10. 選擇建立角色。
11. 套用新建立的 **`s3-object-lambda-role`** 作為 Lambda 函數的執行角色。此動作可以在 Lambda 主控台中建立 Lambda 函數期間或之後完成。

如需執行角色的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 Lambda 執行角色。

搭配 Object Lambda 存取點使用內容索引鍵

S3 Object Lambda 將評估內容索引鍵，例如，與連線相或請求簽署相關的 `s3-object-lambda:TlsVersion` 或 `s3-object-lambda:AuthType`。所有其他內容索引鍵，例如 `s3:prefix`，都會由 Amazon S3 評估。

Object Lambda 存取點 CORS 支援

當 S3 Object Lambda 從瀏覽器接收到請求，或請求包含 `Origin` 標頭時，S3 Object Lambda 一律會新增 `“AllowedOrigins":"*"` 標頭欄位。

如需詳細資訊，請參閱 使用跨來源資源分享 (CORS)。

撰寫 S3 Object Lambda 存取點的 Lambda 函數

本節詳細說明如何撰寫與 Amazon S3 物件 Lambda 存取點搭配使用的 AWS Lambda 函數。

若要了解某些 S3 物件 Lambda 任務的完整 end-to-end 程序，請參閱下列內容：

- 教學課程：使用 S3 Object Lambda 轉換應用程式的資料
- 教學課程：使用 S3 Object Lambda 和 Amazon Comprehend 來偵測和編輯 PII 資料
- 教學課程：使用 S3 Object Lambda 在擷取影像時動態加上浮水印

主題

- 使用 Lambda 中的 GetObject 請求
- 使用 Lambda 中的 HeadObject 請求
- 使用 Lambda 中的 ListObjects 請求
- 使用 Lambda 中的 ListObjectsV2 請求
- 事件內容格式和用量
- 使用 Range 和 partNumber 標頭

使用 Lambda 中的 **`GetObject`** 請求

本節假設您的 Object Lambda 存取點已設定為呼叫 `GetObject` 的 Lambda 函數。S3 Object Lambda 包含 Amazon S3 API 操作 (`WriteGetObjectResponse`)，其可讓 Lambda 函數為 `GetObject` 呼叫者提供自訂的資料和回應標頭。

WriteGetObjectResponse 可讓您根據處理需求，全面控制狀態碼、回應標頭和回應本文。您可以使用 WriteGetObjectResponse，以整個轉換的物件、轉換物件的部分，或是基於應用程式內容的其他回應進行回應。下一節展示了使用 WriteGetObjectResponse API 操作的唯一範例。

- 範例 1：使用 HTTP 狀態代碼 403 (禁止) 回應
- 範例 2：以轉換後的映像回應
- 範例 3：串流壓縮內容

範例 1：使用 HTTP 狀態代碼 403 (禁止) 回應

您可以根據物件內容使用 WriteGetObjectResponse 回應 HTTP 狀態代碼 403 (禁止)。

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;

import java.io.ByteArrayInputStream;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example1 {

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();

        // Check to see if the request contains all of the necessary information.
        // If it does not, send a 4XX response and a custom error code and message.
        // Otherwise, retrieve the object from S3 and stream it
        // to the client unchanged.
        var tokenIsNotPresent = !
event.getUserRequest().getHeaders().containsKey("requiredToken");
        if (tokenIsNotPresent) {
```

```
            s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                    .withRequestRoute(event.outputRoute())
                    .withRequestToken(event.outputToken())
                    .withStatusCode(403)
                    .withContentLength(0L).withInputStream(new
 ByteArrayInputStream(new byte[0]))
                    .withErrorCode("MissingRequiredToken")
                    .withErrorMessage("The required token was not present in the
 request."));
            return;
        }

        // Prepare the presigned URL for use and make the request to S3.
        HttpClient httpClient = HttpClient.newBuilder().build();
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // Stream the original bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(presignedResponse.body()));
    }
}
```

Python

```
import boto3
import requests

def handler(event, context):
    s3 = boto3.client('s3')

    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and contains a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to
    S3 Object Lambda.
    """
```

```
    get_context = event["getObjectContext"]
    user_request_headers = event["userRequest"]["headers"]

    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    # Check for the presence of a 'CustomHeader' header and deny or allow based on
 that header.
    is_token_present = "SuperSecretToken" in user_request_headers

    if is_token_present:
        # If the user presented our custom 'SuperSecretToken' header, we send the
 requested object back to the user.
        response = requests.get(s3_url)
        s3.write_get_object_response(RequestRoute=route, RequestToken=token,
 Body=response.content)
    else:
        # If the token is not present, we send an error back to the user.
        s3.write_get_object_response(RequestRoute=route, RequestToken=token,
 StatusCode=403,
        ErrorCode="NoSuperSecretTokenFound", ErrorMessage="The request was not
 secret enough.")

    # Gracefully exit the Lambda function.
    return { 'status_code': 200 }
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and contains a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    // The 'userRequest' object has information related to the user who made this
 'GetObject' request to S3 Object Lambda.
    const { userRequest, getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;
```

```
    // Check for the presence of a 'CustomHeader' header and deny or allow based on
 that header.
    const isTokenPresent = Object
        .keys(userRequest.headers)
        .includes("SuperSecretToken");

    if (!isTokenPresent) {
        // If the token is not present, we send an error back to the user. The
 'await' in front of the request
        // indicates that we want to wait for this request to finish sending before
 moving on.
        await s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            StatusCode: 403,
            ErrorCode: "NoSuperSecretTokenFound",
            ErrorMessage: "The request was not secret enough.",
        }).promise();
    } else {
        // If the user presented our custom 'SuperSecretToken' header, we send the
 requested object back to the user.
        // Again, note the presence of 'await'.
        const presignedResponse = await axios.get(inputS3Url);
        await s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            Body: presignedResponse.data,
        }).promise();
    }

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

範例 2：以轉換後的映像回應

執行映像轉換時，您可能會發現您需要來源物件的所有位元組，才能開始對其進行處理。在本例中，`WriteGetObjectResponse` 請求會在一次呼叫中，將整個物件傳回給請求的應用程式。

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.awt.Image;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example2 {

    private static final int HEIGHT = 250;
    private static final int WIDTH = 250;

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();
        HttpClient httpClient = HttpClient.newBuilder().build();

        // Prepare the presigned URL for use and make the request to S3.
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // The entire image is loaded into memory here so that we can resize it.
        // Once the resizing is completed, we write the bytes into the body
        // of the WriteGetObjectResponse request.
        var originalImage = ImageIO.read(presignedResponse.body());
        var resizingImage = originalImage.getScaledInstance(WIDTH, HEIGHT,
 Image.SCALE_DEFAULT);
        var resizedImage = new BufferedImage(WIDTH, HEIGHT,
 BufferedImage.TYPE_INT_RGB);
        resizedImage.createGraphics().drawImage(resizingImage, 0, 0, WIDTH, HEIGHT,
 null);
```

```
        var baos = new ByteArrayOutputStream();
        ImageIO.write(resizedImage, "png", baos);

        // Stream the bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(new ByteArrayInputStream(baos.toByteArray())));
    }
}
```

Python

```
import boto3
import requests
import io
from PIL import Image

def handler(event, context):
    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to
    S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    """
    In this case, we're resizing .png images that are stored in S3 and are
 accessible through the presigned URL
    'inputS3Url'.
    """
    image_request = requests.get(s3_url)
    image = Image.open(io.BytesIO(image_request.content))
    image.thumbnail((256,256), Image.ANTIALIAS)
```

```
    transformed = io.BytesIO()
    image.save(transformed, "png")

    # Send the resized image back to the client.
    s3 = boto3.client('s3')
    s3.write_get_object_response(Body=transformed.getvalue(), RequestRoute=route,
 RequestToken=token)

    # Gracefully exit the Lambda function.
    return { 'status_code': 200 }
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;
const sharp = require('sharp');

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    const { getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // In this case, we're resizing .png images that are stored in S3 and are
 accessible through the presigned URL
    // 'inputS3Url'.
    const { data } = await axios.get(inputS3Url, { responseType: 'arraybuffer' });

    // Resize the image.
    const resized = await sharp(data)
        .resize({ width: 256, height: 256 })
        .toBuffer();

    // Send the resized image back to the client.
    await s3.writeGetObjectResponse({
        RequestRoute: outputRoute,
        RequestToken: outputToken,
        Body: resized,
    }).promise();
```

```
    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

範例 3：串流壓縮內容

壓縮物件時，壓縮的資料會以增量方式產生。因此，您可以使用 WriteGetObjectResponse 請求，在壓縮資料就緒時將其傳回。如本範例所示，您不需要知道完成轉換的長度。

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;

import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example3 {

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();
        HttpClient httpClient = HttpClient.newBuilder().build();

        // Request the original object from S3.
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // Consume the incoming response body from the presigned request,
        // apply our transformation on that data, and emit the transformed bytes
        // into the body of the WriteGetObjectResponse request as soon as they're
 ready.
        // This example compresses the data from S3, but any processing pertinent
```

```
        // to your application can be performed here.
        var bodyStream = new GZIPCompressingInputStream(presignedResponse.body());

        // Stream the bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(bodyStream));
    }

}
```

Python

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


"""
A helper class to work with content iterators. Takes an interator and compresses the
 bytes that come from it. It
implements 'read' and '__iter__' so that the SDK can stream the response.
"""
class Compress:
    def __init__(self, content_iter):
        self.content = content_iter
        self.compressed_obj = zlib.compressobj()

    def read(self, _size):
        for data in self.__iter__()
            return data

    def __iter__(self):
        while True:
            data = next(self.content)
            chunk = self.compressed_obj.compress(data)
            if not chunk:
                break

            yield chunk
```

```
        yield self.compressed_obj.flush()


def handler(event, context):
    """
    Setting the 'payload_signing_enabled' property to False allows us to send a
 streamed response back to the client.
    in this scenario, a streamed response means that the bytes are not buffered into
 memory as we're compressing them,
    but instead are sent straight to the user.
    """
    my_config = Config(
        region_name='eu-west-1',
        signature_version='s3v4',
        s3={
            "payload_signing_enabled": False
        }
    )
    s3 = boto3.client('s3', config=my_config)

    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    # Compress the 'get' request stream.
    with requests.get(s3_url, stream=True) as r:
        compressed = Compress(r.iter_content())

        # Send the stream back to the client.
        s3.write_get_object_response(Body=compressed, RequestRoute=route,
 RequestToken=token, ContentType="text/plain",
                                     ContentEncoding="gzip")

    # Gracefully exit the Lambda function.
```

```
    return {'status_code': 200}
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;
const zlib = require('zlib');

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    const { getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // Download the object from S3 and process it as a stream, because it might be a
 huge object and we don't want to
    // buffer it in memory. Note the use of 'await' because we want to wait for
 'writeGetObjectResponse' to finish
    // before we can exit the Lambda function.
    await axios({
        method: 'GET',
        url: inputS3Url,
        responseType: 'stream',
    }).then(
        // Gzip the stream.
        response => response.data.pipe(zlib.createGzip())
    ).then(
        // Finally send the gzip-ed stream back to the client.
        stream => s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            Body: stream,
            ContentType: "text/plain",
            ContentEncoding: "gzip",
        }).promise()
    );

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
```

```
}
```

> **Note**
>
> 雖然 S3 Object Lambda 允許最多 60 秒，透過 `WriteGetObjectResponse` 請求將完整的回應傳送給呼叫者，但實際可用時間可能會更少。例如，Lambda 函數逾時可能小於 60 秒。在其他情況下，呼叫者可能會有更嚴格的逾時要求。

對於接收非 HTTP 狀態碼 500 (內部伺服器錯誤) 的原始呼叫者，必須完成 `WriteGetObjectResponse` 呼叫。異常或其他情形下，如果 Lambda 函數在呼叫 `WriteGetObjectResponse` API 操作之前傳回，則原始呼叫者將會收到 500 (內部伺服器錯誤) 回應。在完成回應所需的時間內擲出的異常會導致對呼叫者的回應截斷。如果 Lambda 函數從 `WriteGetObjectResponse` API 呼叫收到 HTTP 狀態碼 200 (OK) 回應，則原始呼叫者已傳送完整的請求。無論是否擲出異常，S3 Object Lambda 都會忽略 Lambda 函數的回應。

呼叫 `WriteGetObjectResponse` API 操作時，Amazon S3 需要來自事件內容的路由和請求字符。如需詳細資訊，請參閱 事件內容格式和用量。

路由和請求字符參數是必要項目，用來將 `WriteGetObjectResult` 回應與原始呼叫者連線。即使重試 500 (內部伺服器錯誤) 回應始終適用，仍請注意，請求字符是單次使用字符，後續嘗試使用該字符可能會導致 HTTP 狀態碼 400 (錯誤請求) 回應。雖然使用路由和請求字符的 `WriteGetObjectResponse` 呼叫不需要從叫用的 Lambda 函數建立，但必須由同一帳戶中的身分進行呼叫。在 Lambda 函數完成執行之前，還必須完成呼叫。

使用 Lambda 中的 **`HeadObject`** 請求

本節假設您的 Object Lambda 存取點已設定為呼叫 `HeadObject` 的 Lambda 函數。Lambda 將收到一個 JSON 承載，其中包含一個稱為 `headObjectContext` 的金鑰。在內容中，有一個稱為 `inputS3Url` 的單一屬性，這是 `HeadObject` 支援存取點的預先簽章 URL。

如果已指定預先簽章 URL，則此 URL 將包含下列屬性：

- `versionId` (在查詢參數中)
- `requestPayer` (在 `x-amz-request-payer` 標頭中)
- `expectedBucketOwner` (在 `x-amz-expected-bucket-owner` 標頭中)

其他屬性不會預先簽章，因此不會包含在內。呼叫 userRequest 標頭中找到的預先簽章 URL 時，可以將以標頭形式傳送的未簽章選項手動新增至請求。HeadObject 不支援伺服器端加密選項。

如需請求語法 URI 參數，請參閱《Amazon Simple Storage Service API 參考》中的 HeadObject。

下列範例顯示 HeadObject 的 Lambda JSON 輸入承載。

```
{
  "xAmzRequestId": "requestId",
  "**headObjectContext**": {
    "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/example?X-Amz-Security-Token=<snip>"
  },
  "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
  },
  "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
   },
   "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
             "attributes": {
             "mfaAuthenticated": "false",
             "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
        },
        "sessionIssuer": {
             "type": "Role",
             "principalId": "principalId",
```

```
            "arn": "arn:aws:iam::111122223333:role/Admin",
            "accountId": "111122223333",
            "userName": "Admin"
            }
       }
    },
  "protocolVersion": "1.00"
}
```

您的 Lambda 函數應該傳回一個 JSON 物件，其中包含將為 `HeadObject` 呼叫傳回的標頭和值。

下列範例顯示 `HeadObject` 的 Lambda 回應 JSON 結構。

```
{
    "statusCode": <number>; // Required
    "errorCode": <string>;
    "errorMessage": <string>;
    "headers": {
        "Accept-Ranges": <string>,
        "x-amz-archive-status": <string>,
        "x-amz-server-side-encryption-bucket-key-enabled": <boolean>,
        "Cache-Control": <string>,
        "Content-Disposition": <string>,
        "Content-Encoding": <string>,
        "Content-Language": <string>,
        "Content-Length": <number>, // Required
        "Content-Type": <string>,
        "x-amz-delete-marker": <boolean>,
        "ETag": <string>,
        "Expires": <string>,
        "x-amz-expiration": <string>,
        "Last-Modified": <string>,
        "x-amz-missing-meta": <number>,
        "x-amz-object-lock-mode": <string>,
        "x-amz-object-lock-legal-hold": <string>,
        "x-amz-object-lock-retain-until-date": <string>,
        "x-amz-mp-parts-count": <number>,
        "x-amz-replication-status": <string>,
        "x-amz-request-charged": <string>,
        "x-amz-restore": <string>,
        "x-amz-server-side-encryption": <string>,
        "x-amz-server-side-encryption-customer-algorithm": <string>,
        "x-amz-server-side-encryption-aws-kms-key-id": <string>,
        "x-amz-server-side-encryption-customer-key-MD5": <string>,
```

```
        "x-amz-storage-class": <string>,
        "x-amz-tagging-count": <number>,
        "x-amz-version-id": <string>,
        <x-amz-meta-headers>: <string>, // user-defined metadata
        "x-amz-meta-meta1": <string>, // example of the user-defined metadata header,
 it will need the x-amz-meta prefix
        "x-amz-meta-meta2": <string>
        ...
    };
}
```

下列範例會示範如何使用預先簽章的 URL 填入回應，方法是在傳回 JSON 之前視需要修改標頭值。

Python

```
import requests

def lambda_handler(event, context):
    print(event)

    # Extract the presigned URL from the input.
    s3_url = event["headObjectContext"]["inputS3Url"]

    # Get the head of the object from S3.
    response = requests.head(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        return {
            "statusCode": response.status_code,
            "errorCode": "RequestFailure",
            "errorMessage": "Request to S3 failed"
    }

    # Store the headers in a dictionary.
    response_headers = dict(response.headers)

    # This obscures Content-Type in a transformation, it is optional to add
    response_headers["Content-Type"] = ""

    # Return the headers to S3 Object Lambda.
    return {
        "statusCode": response.status_code,
```

```
        "headers": response_headers
        }
```

使用 Lambda 中的 **ListObjects** 請求

本節假設您的 Object Lambda 存取點已設定為呼叫 `ListObjects` 的 Lambda 函數。Lambda 將接收 JSON 承載，其中包含名為 `listObjectsContext` 的新物件。`listObjectsContext` 包含單一屬性 `inputS3Url`，這是 `ListObjects` 支援存取點的預先簽章 URL。

與 `GetObject` 和 `HeadObject` 不同，如果已指定預先簽章 URL，則此 URL 將包含下列屬性：

- 所有查詢參數
- `requestPayer` (在 `x-amz-request-payer` 標頭中)
- `expectedBucketOwner` (在 `x-amz-expected-bucket-owner` 標頭中)

如需請求語法 URI 參數，請參閱《Amazon Simple Storage Service API 參考》中的 `ListObjects`。

> **Important**
>
> 我們建議您在開發應用程式時使用較新版本 ListObjectsV2。為了回溯相容性，Amazon S3 繼續支援 `ListObjects`。

下列範例顯示 `ListObjects` 的 Lambda JSON 輸入承載。

```
{
    "xAmzRequestId": "requestId",
     "**listObjectsContext**": {
     "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/?X-Amz-Security-Token=<snip>",
     },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
```

```
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
  "protocolVersion": "1.00"
}
```

您的 Lambda 函數應該傳回 JSON 物件，其中包含狀態碼、清單 XML 結果或將從 S3 Object Lambda 傳回的錯誤資訊。

S3 Object Lambda 不會處理或驗證 `listResultXml`，而是將其轉送給 `ListObjects` 呼叫者。對於 `listBucketResult`，S3 Object Lambda 預期某些屬性為特定類型，而且如果無法剖析它們，則會擲回例外狀況。不能同時提供 `listResultXml` 和 `listBucketResult`。

下列範例示範如何使用預先簽章 URL 呼叫 Amazon S3，並使用結果填入回應，包括錯誤檢查。

Python

```
import requests
import xmltodict

def lambda_handler(event, context):
    # Extract the presigned URL from the input.
    s3_url = event["listObjectsContext"]["inputS3Url"]


    # Get the head of the object from Amazon S3.
    response = requests.get(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        error = xmltodict.parse(response.content)
        return {
            "statusCode": response.status_code,
            "errorCode": error["Error"]["Code"],
            "errorMessage": error["Error"]["Message"]
        }

    # Store the XML result in a dict.
    response_dict = xmltodict.parse(response.content)

    # This obscures StorageClass in a transformation, it is optional to add
    for item in response_dict['ListBucketResult']['Contents']:
        item['StorageClass'] = ""

    # Convert back to XML.
    listResultXml = xmltodict.unparse(response_dict)

    # Create response with listResultXml.
    response_with_list_result_xml = {
        'statusCode': 200,
        'listResultXml': listResultXml
    }

    # Create response with listBucketResult.
    response_dict['ListBucketResult'] =
 sanitize_response_dict(response_dict['ListBucketResult'])
    response_with_list_bucket_result = {
        'statusCode': 200,
        'listBucketResult': response_dict['ListBucketResult']
```

```
    }

    # Return the list to S3 Object Lambda.
    # Can return response_with_list_result_xml or response_with_list_bucket_result
    return response_with_list_result_xml

# Converting the response_dict's key to correct casing
def sanitize_response_dict(response_dict: dict):
    new_response_dict = dict()
    for key, value in response_dict.items():
        new_key = key[0].lower() + key[1:] if key != "ID" else 'id'
        if type(value) == list:
            newlist = []
            for element in value:
                if type(element) == type(dict()):
                    element = sanitize_response_dict(element)
                newlist.append(element)
            value = newlist
        elif type(value) == dict:
            value = sanitize_response_dict(value)
        new_response_dict[new_key] = value
    return new_response_dict
```

下列範例顯示 `ListObjects` 的 Lambda 回應 JSON 結構。

```
{
  "statusCode": <number>; // Required
  "errorCode": <string>;
  "errorMessage": <string>;
  "listResultXml": <string>; // This can also be Error XML string in case S3 returned
 error response when calling the pre-signed URL

  "listBucketResult": {  // listBucketResult can be provided instead of listResultXml,
 however they can not both be provided in the JSON response
        "name": <string>,  // Required for 'listBucketResult'
        "prefix": <string>,
        "marker": <string>,
        "nextMarker": <string>,
        "maxKeys": <int>,   // Required for 'listBucketResult'
        "delimiter": <string>,
        "encodingType": <string>
        "isTruncated": <boolean>,  // Required for 'listBucketResult'
```

```
        "contents": [  {
            "key": <string>,  // Required for 'content'
            "lastModified": <string>,
            "eTag": <string>,
            "checksumAlgorithm": <string>,   // CRC32,  CRC32C,  SHA1,  SHA256
            "size": <int>,   // Required for 'content'
            "owner": {
                "displayName": <string>,  // Required for 'owner'
                "id": <string>,  // Required for 'owner'
            },
            "storageClass": <string>
            },
        ...
        ],
        "commonPrefixes": [  {
            "prefix": <string>   // Required for 'commonPrefix'
        },
        ...
        ],
    }
}
```

使用 Lambda 中的 **ListObjectsV2** 請求

本節假設您的 Object Lambda 存取點已設定為呼叫 `ListObjectsV2` 的 Lambda 函數。Lambda 將接收 JSON 承載，其中包含名為 `listObjectsV2Context` 的新物件。`listObjectsV2Context` 包含單一屬性 `inputS3Url`，這是 `ListObjectsV2` 支援存取點的預先簽章 URL。

與 `GetObject` 和 `HeadObject` 不同，如果已指定預先簽章 URL，則此 URL 將包含下列屬性：

- 所有查詢參數
- `requestPayer` (在 `x-amz-request-payer` 標頭中)
- `expectedBucketOwner` (在 `x-amz-expected-bucket-owner` 標頭中)

如需請求語法 URI 參數，請參閱《Amazon Simple Storage Service API 參考》中的 `ListObjectsV2`。

下列範例顯示 `ListObjectsV2` 的 Lambda JSON 輸入承載。

```
{
    "xAmzRequestId": "requestId",
```

```
     "**listObjectsV2Context**": {
     "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/?list-type=2&X-Amz-Security-Token=<snip>",
     },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
  "protocolVersion": "1.00"
}
```

您的 Lambda 函數應該傳回 JSON 物件，其中包含狀態碼、清單 XML 結果或將從 S3 Object Lambda 傳回的錯誤資訊。

S3 Object Lambda 不會處理或驗證 `listResultXml`，而是將其轉送給 `ListObjectsV2` 呼叫者。對於 `listBucketResult`，S3 Object Lambda 預期某些屬性為特定類型，而且如果無法剖析它們，則會擲回例外狀況。不能同時提供 `listResultXml` 和 `listBucketResult`。

下列範例示範如何使用預先簽章 URL 呼叫 Amazon S3，並使用結果填入回應，包括錯誤檢查。

Python

```
import requests
import xmltodict

def lambda_handler(event, context):
    # Extract the presigned URL from the input.
    s3_url = event["listObjectsV2Context"]["inputS3Url"]


    # Get the head of the object from Amazon S3.
    response = requests.get(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        error = xmltodict.parse(response.content)
        return {
            "statusCode": response.status_code,
            "errorCode": error["Error"]["Code"],
            "errorMessage": error["Error"]["Message"]
        }

    # Store the XML result in a dict.
    response_dict = xmltodict.parse(response.content)

    # This obscures StorageClass in a transformation, it is optional to add
    for item in response_dict['ListBucketResult']['Contents']:
        item['StorageClass'] = ""

    # Convert back to XML.
    listResultXml = xmltodict.unparse(response_dict)

    # Create response with listResultXml.
    response_with_list_result_xml = {
```

```
        'statusCode': 200,
        'listResultXml': listResultXml
    }

    # Create response with listBucketResult.
    response_dict['ListBucketResult'] =
 sanitize_response_dict(response_dict['ListBucketResult'])
    response_with_list_bucket_result = {
        'statusCode': 200,
        'listBucketResult': response_dict['ListBucketResult']
    }

    # Return the list to S3 Object Lambda.
    # Can return response_with_list_result_xml or response_with_list_bucket_result
    return response_with_list_result_xml

# Converting the response_dict's key to correct casing
def sanitize_response_dict(response_dict: dict):
    new_response_dict = dict()
    for key, value in response_dict.items():
        new_key = key[0].lower() + key[1:] if key != "ID" else 'id'
        if type(value) == list:
            newlist = []
            for element in value:
                if type(element) == type(dict()):
                    element = sanitize_response_dict(element)
                newlist.append(element)
            value = newlist
        elif type(value) == dict:
            value = sanitize_response_dict(value)
        new_response_dict[new_key] = value
    return new_response_dict
```

下列範例顯示 `ListObjectsV2` 的 Lambda 回應 JSON 結構。

```
{
    "statusCode": <number>; // Required
    "errorCode": <string>;
    "errorMessage": <string>;
    "listResultXml": <string>; // This can also be Error XML string in case S3 returned
 error response when calling the pre-signed URL
```

```
    "listBucketResult": {  // listBucketResult can be provided instead of
 listResultXml, however they can not both be provided in the JSON response
        "name": <string>, // Required for 'listBucketResult'
        "prefix": <string>,
        "startAfter": <string>,
        "continuationToken": <string>,
        "nextContinuationToken": <string>,
        "keyCount": <int>, // Required for 'listBucketResult'
        "maxKeys": <int>, // Required for 'listBucketResult'
        "delimiter": <string>,
        "encodingType": <string>
        "isTruncated": <boolean>, // Required for 'listBucketResult'
        "contents": [ {
            "key": <string>, // Required for 'content'
            "lastModified": <string>,
            "eTag": <string>,
            "checksumAlgorithm": <string>, // CRC32, CRC32C, SHA1, SHA256
            "size": <int>, // Required for 'content'
            "owner": {
                "displayName": <string>, // Required for 'owner'
                "id": <string>, // Required for 'owner'
            },
            "storageClass": <string>
            },
            ...
        ],
        "commonPrefixes": [ {
            "prefix": <string> // Required for 'commonPrefix'
            },
        ...
        ],
    }
}
```

事件內容格式和用量

Amazon S3 物件 Lambda 提供有關在傳遞至 AWS Lambda 函數的事件中發出的請求的內容。請求範例如下所示。範例後面包括欄位的描述。

```
{
    "xAmzRequestId": "requestId",
    "getObjectContext": {
```

```
        "inputS3Url": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/example?X-Amz-Security-Token=<snip>",
        "outputRoute": "io-use1-001",
        "outputToken": "OutputToken"
    },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
    "protocolVersion": "1.00"
```

```
}
```

請求中包括下列欄位：

- `xAmzRequestId`：此請求的 Amazon S3 請求 ID。我們建議您記錄此值以協助進行偵錯。
- `getObjectContext`：連線至 Amazon S3 和 S3 Object Lambda 的輸入和輸出詳細資訊。
  - `inputS3Url`：預先簽章 URL，可用於從 Amazon S3 擷取原始物件。URL 是使用原始呼叫者的身分進行簽署，而且該使用者的許可將在使用 URL 時套用。如果 URL 中有簽署的標頭，Lambda 函數必須在對 Amazon S3 的呼叫中包含這些標頭，除了 `Host` 標頭之外。
  - `outputRoute` – 當 Lambda 函數呼叫 `WriteGetObjectResponse` 時，會新增至 S3 Object Lambda URL 的路由字符。
  - `outputToken` - S3 Object Lambda 用來將 `WriteGetObjectResponse` 呼叫與原始呼叫者進行比對的不透明字符。
- `configuration`：有關 Object Lambda 存取點的組態資訊。
  - `accessPointArn`：接收此請求之 Object Lambda 存取點的 Amazon Resource Name (ARN)。
  - `supportingAccessPointArn`：Object Lambda 存取點組態中指定的支援存取點的 ARN。
  - `payload`：套用至 Object Lambda 存取點組態的自訂資料。S3 Object Lambda 將此資料視為不透明字串，因此可能需要在使用之前進行解碼。
- `userRequest`：原始呼叫 S3 Object Lambda 的相關資訊。
  - `url`：S3 Object Lambda 接收之請求的解碼 URL，不包括任何授權相關的查詢參數。
  - `headers`：從原始呼叫中包含 HTTP 標頭及其值之字串到字串的映射，不包括任何授權相關的標頭。如果相同的標頭出現多次，來自相同標頭的每個執行個體的值會合併成逗號分隔的清單。原始標頭的情形保留在此映射中。
- `userIdentity`：有關呼叫 S3 Object Lambda 身分的詳細資訊。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的記錄資料事件。
  - `type`：身分的類型。
  - `accountId`— 身 AWS 帳戶 份所屬的。
  - `userName`：發出呼叫之身分的易記名稱。
  - `principalId`：發出呼叫之身分的唯一識別符。
  - `arn`：發出呼叫之主體的 ARN。ARN 的最後一個部分包含發出呼叫的使用者或角色。
  - `sessionContext`：如果使用臨時安全憑證提出請求，此元素會提供為這些憑證所建立之工作階段的相關資訊。

  - `invokedBy`— 提出請求 AWS 服務 的名稱，例如 Amazon EC2 Auto Scaling 或 AWS Elastic Beanstalk.
  - `sessionIssuer`：如果使用臨時安全憑證提出請求，此元素會提供憑證取得方式的相關資訊。
- `protocolVersion`：提供之內容的版本 ID。此欄位的格式為 `{Major Version}.{Minor Version}`。次要版本號碼永遠是兩位數。對欄位的語意進行任何移除或變更都需要提升主要版本，並且需要主動選擇加入。Amazon S3 可以隨時新增欄位，此時您可能會遇到次要版本凸起。由於軟體發行的性質，您可能會同時看到多個次要版本正在使用中。

使用 Range 和 partNumber 標頭

在 Amazon S3 Object Lambda 中使用大型物件時，您可以使用 `Range` HTTP 標頭，從物件下載指定的位元組範圍。若要從同一物件內擷取不同的位元組範圍，您可以對 Amazon S3 使用並行連線。您還可以指定 `partNumber` 參數 (1 到 10,000 之間的整數)，其會針對物件的指定部分執行範圍請求。

因為您可能需要多種方法來處理包含 `Range` 或者 `partNumber` 參數的請求，而 S3 Object Lambda 不會將這些參數套用至轉換的物件。相反，您的 AWS Lambda 函數必須根據應用程序的需要實現此功能。

若要搭配 S3 Object Lambda 使用 `Range` 和 `partNumber` 參數，請執行下列動作：

- 在 Object Lambda 存取點組態中啟用這些參數。
- 撰寫一個 Lambda 函數，其可以處理包含這些參數的請求。

下列步驟說明如何完成這項操作。

步驟 1：設定 Object Lambda 存取點

根據預設，Object Lambda 存取點會以 HTTP 狀態碼 501 (未實作) 錯誤回應任何 `GetObject` 或 `HeadObject` 請求，該請求包含在標頭或查詢參數中的 `Range` 或 `partNumber` 參數。

若要啟用 Object Lambda 存取點以接受此類請求，您必須在 Object Lambda 存取點組態的 `AllowedFeatures` 區段中包括 `GetObject-Range`、`GetObject-PartNumber`、`HeadObject-Range` 或 `HeadObject-PartNumber`。如需更新 Object Lambda 存取點組態的詳細資訊，請參閱 建立 Object Lambda 存取點。

步驟 2：在 Lambda 函數中實作 **Range** 或 **partNumber** 處理

當 Object Lambda 存取點使用範圍 GetObject 或 HeadObject 請求叫用 Lambda 函數時，Range 或 partNumber 參數會包含在事件內容中。如下表所述，參數在事件內容中的位置，取決於使用的參數以及將其包含在對 Object Lambda 存取點的原始請求中的方式。

參數	事件內容位置
Range (標頭)	userRequest.headers.Range
Range (查詢參數)	userRequest.url (查詢參數 Range)
partNumber	userRequest.url (查詢參數 partNumber)

> **Important**
>
> 針對您的 Object Lambda 存取點所提供的預先簽章 URL 不包含來自原始請求的 Range 或 partNumber 參數。請參閱以下有關如何在 AWS Lambda 函數中處理這些參數的選項。

在擷取 Range 或 partNumber 值後，您可以根據應用程式的需求採取下列其中一種方法：

A. 將請求的 **Range** 或 **partNumber** 映射到轉換的物件 (建議)。

處理 Range 或 partNumber 請求的最可靠方式是執行以下動作：

- 從 Amazon S3 擷取完整物件。
- 轉換物件。
- 將請求的 Range 或 partNumber 參數套用至轉換的物件。

為此，請使用提供的預先簽章 URL 從 Amazon S3 擷取整個物件，然後根據需要處理物件。如需以這種方式處理Range參數的 Lambda 函數範例，請參閱範例 GitHub 儲存庫中 AWS 的此範例。

B. 將請求的 **Range** 映射到預先簽署的 URL。

在某些情況下，Lambda 函數可以將請求的 Range 直接映射到預先簽署的 URL，以便從 Amazon S3 中僅擷取部分物件。只有當轉換符合以下兩個條件時，此方法才適用：

1. 轉換函數可套用於部分物件範圍。
2. 在轉換函數之前或之後套用 Range 參數，將會產生相同的轉換物件。

例如，將 ASCII 編碼物件中的所有字元轉換為大寫的轉換函數滿足上述兩個條件。轉換可套用至物件的一部分，而且在轉換前與轉換後套用 Range 參數的結果相同。

相比之下，將 ASCII 編碼物件中的字元反轉的函數不符合這些條件。這類函數滿足標準 1，因為其可以套用於部分物件範圍。但不符合標準 2，因為在轉換前套用 Range 參數與轉換後套用參數的結果不同。

請考慮以下請求：將函數套用至包含內容 abcdefg 之物件的前三個字元。在轉換前套用 Range 參數會僅擷取 abc，接著反轉資料，傳回 cba。但是，如果在轉換之後套用參數，函數將擷取整個物件，將其反轉，然後套用 Range 參數，最終傳回 gfe。因為這些結果都不同，所以此函數不應在從 Amazon S3 擷取物件時套用 Range 參數。其反而應該擷取整個物件、執行轉換，然後才套用 Range 參數。

> **Warning**
>
> 在許多情況下，將 Range 參數套用至預先簽署的 URL，將導致 Lambda 函數或請求用戶端的發生意外行為。如前文方法 A 中所述，除非確定應用程式在從 Amazon S3 中僅擷取部分物件時能夠正常工作，否則建議您擷取和轉換完整物件。

如果您的應用程式符合方法 B 先前描述的準則，您可以透過僅擷取要求的物件範圍，然後在該範圍上執行轉換，以簡化 AWS Lambda 函數。

下列 Java 程式碼範例會示範如何執行下列動作：

- 從 GetObject 請求中擷取 Range 標頭。
- 將 Range 標頭新增至 Lambda 可以用來從 Amazon S3 擷取請求範圍的預先簽章 URL。

```
private HttpRequest.Builder applyRangeHeader(ObjectLambdaEvent event,
 HttpRequest.Builder presignedRequest) {
    var header = event.getUserRequest().getHeaders().entrySet().stream()
            .filter(e -> e.getKey().toLowerCase(Locale.ROOT).equals("range"))
            .findFirst();

    // Add check in the query string itself.
    header.ifPresent(entry -> presignedRequest.header(entry.getKey(),
 entry.getValue()));
    return presignedRequest;
}
```

使用 AWS 內 Lambda 函數

AWS 提供一些預先建置的 AWS Lambda 函數，您可以與 Amazon S3 物件 Lambda 搭配使用，以偵測和編輯個人識別資訊 (PII) 和解壓縮 S3 物件。這些 Lambda 函數可在 AWS Serverless Application Repository中使用。建立 Object Lambda 存取點時，您可以透過 AWS Management Console 選取這些函數。

如需有關如何部署無伺服器應用程式的詳細資訊 AWS Serverless Application Repository，請參閱AWS Serverless Application Repository 開發人員指南中的部署應用程式。

> **Note**
>
> 下列範例只能與 `GetObject` 請求搭配使用。

範例 1：PII 存取控制

此 Lambda 函數使用了 Amazon Comprehend，這是一種自然語言處理 (NLP) 服務，使用機器學習在文字中尋找洞見和關係。此函數會在 Amazon S3 儲存貯體的文件中自動偵測個人身分識別資訊 (PII)，例如姓名、地址、日期、信用卡號碼和社會安全號碼。如果儲存貯體中有包含 PII 的文件，您可以設定 PII 存取控制函數來偵測這些 PII 實體類型，並限制未經授權的使用者存取。

若要開始使用，請在您的帳戶中部署下列 Lambda 函數，然後將此函數的 Amazon Resource Name (ARN) 新增至 Object Lambda 存取點組態。

以下是此函數的範例 ARN：

```
arn:aws:serverlessrepo:us-east-1:111122223333:applications/
ComprehendPiiAccessControlS3ObjectLambda
```

您可以使用下列 AWS Serverless Application Repository 連結在上 AWS Management Console 新增或檢視此函數：ComprehendPiiAccessControlS3 ObjectLambda。

若要檢視此函數 GitHub，請參閱 Amazon Comprehend S3 物件 Lambda。

範例 2：PII 修改

此 Lambda 函數使用了 Amazon Comprehend，這是一種自然語言處理 (NLP) 服務，使用機器學習在文字中尋找洞見和關係。此函數會從 Amazon S3 儲存貯體中的文件自動修改個人身分識別資訊 (PII)，例如姓名、地址、日期、信用卡號碼和社會安全號碼。

如果儲存貯體中有包含信用卡號碼或銀行帳戶資訊等資訊的文件，則您可以設定 PII Redaction S3 Object Lambda 函數來偵測 PII，然後傳回此類已修改 PII 實體類型的文件複本。

若要開始使用，請在您的帳戶中部署下列 Lambda 函數，然後將此函數的 ARN 新增至 Object Lambda 存取點組態。

以下是此函數的範例 ARN：

```
arn:aws:serverlessrepo:us-east-1:111122223333::applications/
ComprehendPiiRedactionS3ObjectLambda
```

您可以使用下列 AWS Serverless Application Repository 連結在上 AWS Management Console 新增或檢視此函數：ComprehendPiiRedactionS3 ObjectLambda。

若要檢視此函數 GitHub，請參閱 Amazon Comprehend S3 物件 Lambda。

若要了解 PII 編修中某些 S3 物件 Lambda 任務的完整 end-to-end 程序，請參閱。教學課程：使用 S3 Object Lambda 和 Amazon Comprehend 來偵測和編輯 PII 資料

範例 3：解壓縮

Lambda 函數 `S3ObjectLambdaDecompression` 可以將以六種壓縮檔案格式之一存放在 Amazon S3 中的物件解壓縮：`bzip2`、`gzip`、`snappy`、`zlib`、`zstandard` 和 `ZIP`。

若要開始使用，請在您的帳戶中部署下列 Lambda 函數，然後將此函數的 ARN 新增至 Object Lambda 存取點組態。

以下是此函數的範例 ARN：

```
arn:aws:serverlessrepo:us-east-1:111122223333::applications/S3ObjectLambdaDecompression
```

您可以使用下列 AWS Serverless Application Repository 連結在上 AWS Management Console 新增或檢視此函數：S3 ObjectLambdaDecompression。

若要檢視此函數 GitHub，請參閱 S3 物件 Lambda 解壓縮。

S3 Object Lambda 的最佳實務和指導方針

使用 S3 Object Lambda 時，請遵循下列最佳實務和指導方針，以優化作業和效能。

主題

- 使用 S3 Object Lambda
- AWS 服務 用於與 S3 對象 Lambda 連接
- Range 和 partNumber 標頭
- 轉換 expiry-date
- 使用 AWS CLI 和 AWS 軟體開發套件

使用 S3 Object Lambda

S3 Object Lambda 僅支援處理 GET、LIST 和 HEAD 請求。任何其他請求都不會調用， AWS Lambda 而是返回標準的非轉換 API 響應。您可以在每 AWS 帳戶 個區域建立最多 1,000 個物件 Lambda 存取點。您使用的 AWS Lambda 函數必須 AWS 帳戶 與物件 Lambda 存取點位於相同的區域。

S3 Object Lambda 允許最多 60 秒，將完整回應串流至其發起人。您的功能也受到 AWS Lambda 預設配額的限制。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 Lambda 配額。

當 S3 Object Lambda 叫用您指定的 Lambda 函數時，您必須負責確保您指定的 Lambda 函數或應用程式從 Amazon S3 覆寫或刪除的任何資料是預期且正確的。

您只能使用 S3 Object Lambda 對物件執行操作。您無法使用 S3 Object Lambda 來執行其他 Amazon S3 操作，例如修改或刪除儲存貯體。如需支援存取點的 S3 操作完整清單，請參閱存取點與 S3 操作的相容性。

除了此清單之外，Object Lambda 存取點不支援 POST Object、CopyObject (做為來源) 和 SelectObjectContent API 操作。

AWS 服務 用於與 S3 對象 Lambda 連接

S3 物件 Lambda 可連接 Amazon S3 AWS Lambda，並選擇性地連接您選擇交付與請求應用程式相關 AWS 服務 的物件。所有與 S3 物件 Lambda 搭配 AWS 服務 使用，均受其各自的服務等級協定 (SLA) 管理。例如，如果有任何 AWS 服務 不符合其「服務承諾」，則您有資格獲得服務積分，如服務 SLA 中所述。

Range 和 **partNumber** 標頭

使用大型物件時，您可以使用 Range HTTP 標頭，從物件下載指定的位元組範圍。當您使用標 Range 頭時，您的請求只會擷取物件的指定部分。您也可以使用 partNumber 標頭，針對物件中的指定部分執行範圍請求。

如需詳細資訊，請參閱 使用 Range 和 partNumber 標頭。

轉換 `expiry-date`

您可以從上的物件 Lambda 存取點開啟或下載轉換物件 AWS Management Console。這些物件必須未過期。如果您的 Lambda 函數會轉換物件的 `expiry-date`，您可能會看到無法開啟或下載的過期物件。此行為僅適用於 S3 Glacier Flexible Retrive 和 S3 Glacier Deep Archive 還原物件。

使用 AWS CLI 和 AWS 軟體開發套件

AWS Command Line Interface (AWS CLI) S3 子命令 (`cpmv`、和`sync`) 和 AWS SDK for Java `TransferManager`類別的使用不支援搭配 S3 物件 Lambda 使用。

S3 Object Lambda 教學課程

下列教學課 end-to-end 程提供部分 S3 物件 Lambda 工作的完整程序。

- 教學課程：使用 S3 Object Lambda 轉換應用程式的資料
- 教學課程：使用 S3 Object Lambda 和 Amazon Comprehend 來偵測和編輯 PII 資料
- 教學課程：使用 S3 Object Lambda 在擷取影像時動態加上浮水印

偵錯 S3 Object Lambda

當 Lambda 函數叫用或執行發生問題時，對 Amazon S3 Object Lambda 存取點的請求可能會導致新的錯誤回應。這些錯誤的格式與標準 Amazon S3 錯誤的格式相同。如需有關 S3 Object Lambda 錯誤的資訊，請參閱《Amazon Simple Storage Service API 參考》中的 S3 Object Lambda 錯誤代碼清單。

如需有關一般 Lambda 函數偵錯的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 Lambda 應用程式的監控和疑難排解。

有關標準 Amazon S3 錯誤的資訊，請參閱《Amazon Simple Storage Service API 參考》中的錯誤回應。

您可以在 Amazon 中 CloudWatch 為對象 Lambda 存取點啟用請求指標。這些指標會協助您監控存取點的操作效能。您可以在 Object Lambda 存取點建立期間或之後啟用請求指標。如需詳細資訊，請參閱 S3 物件 Lambda 請求指標 CloudWatch。

您可以啟用 AWS CloudTrail 資料事件，以取得對 Object Lambda 存取點所提出請求的更精細日誌記錄。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的記錄資料事件。

如需 S3 Object Lambda 的教學課程，請參閱下列各項：

- 教學課程：使用 S3 Object Lambda 轉換應用程式的資料
- 教學課程：使用 S3 Object Lambda 和 Amazon Comprehend 來偵測和編輯 PII 資料
- 教學課程：使用 S3 Object Lambda 在擷取影像時動態加上浮水印

如需標準存取點的詳細資訊，請參閱 使用 Amazon S3 存取點管理資料存取。

如需有關使用儲存貯體的詳細資訊，請參閱 儲存貯體概觀。如需使用物件的資訊，請參閱「Amazon S3 物件概觀」。

什麼是 S3 Express One Zone？

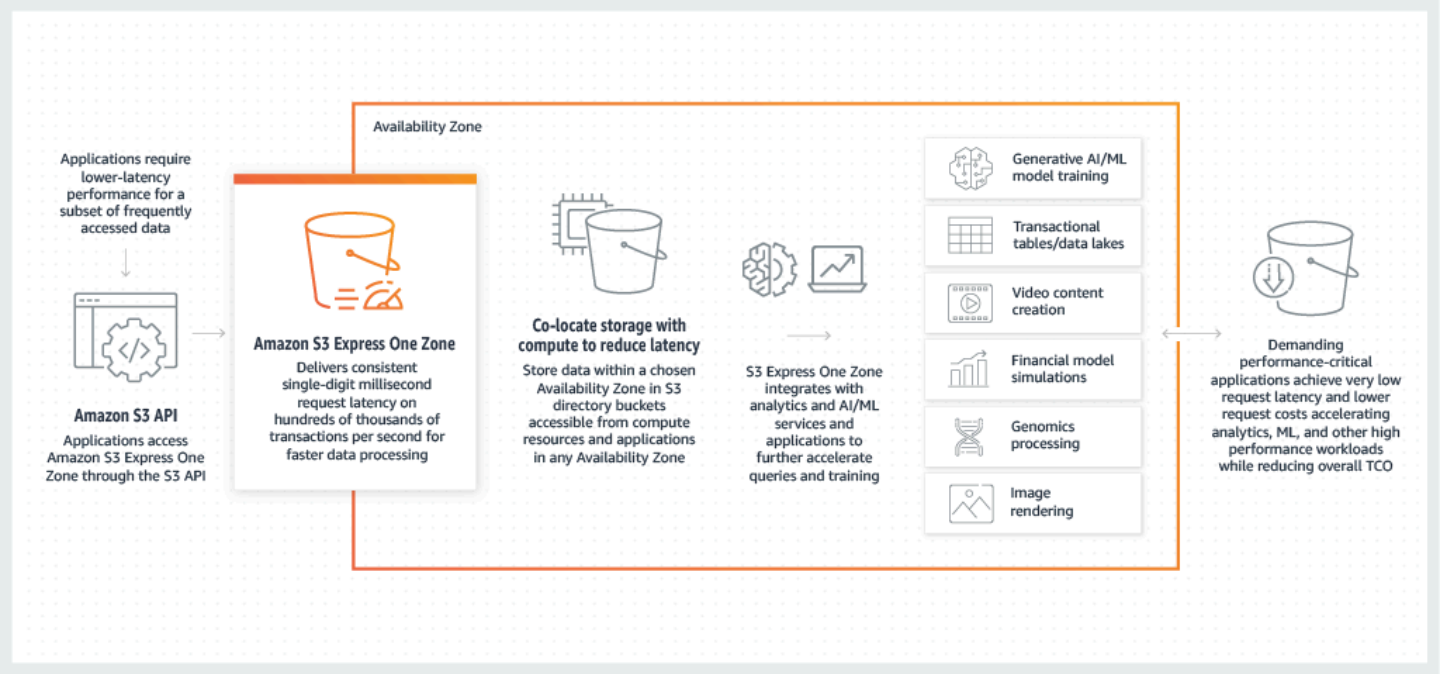
Amazon S3 Express One Zone 是一種高效能的單一區域 Amazon S3 儲存類別，專門為對延遲最敏感的應用程式提供一致的個位數毫秒資料存取。S3 Express One Zone 是目前可用的最低延遲雲端物件儲存類別，資料存取速度快上 10 倍，而且請求成本比 S3 標準低 50%。應用程序可以立即受益於請求的完成速度，速度提高了一個數量級。S3 快速單一區域提供與其他 S3 儲存類別類似的效能彈性。

與其他 Amazon S3 儲存類別一樣，您不需要事先規劃或佈建容量或輸送量需求。您可以根據需要擴展或縮減儲存規模，並透過 Amazon S3 API 存取資料。

S3 Express One Zone 是第一款可讓您選取單一可用區域的 S3 儲存類別，還可選擇將物件儲存體與運算資源共置，藉此盡可能提供最高存取速度。此外，為了進一步提高存取速度並支援每秒數十萬個請求，S3 Express One Zone Zone Zone 儲存類別中的資料會存放在新的儲存貯體類型中：Amazon S3 目錄儲存貯體。每個目錄儲存貯體每秒可支援數十萬筆交易 (TPS)，無論索引鍵名稱或存取模式為何。

Amazon S3 快速單區域儲存類別的設計可在單一可用區域內達到 99.95% 的可用性，並受到 Amazon S3 服務水準協議的支援。使用 S3 Express One Zone，您的資料會以備援方式儲存在單一可用區域內的多部裝置上。S3 Express One Zone 的設計在於透過快速偵測並修復任何遺失的備援來處理並行裝置故障。如果現有裝置發生故障，S3 Express One Zone 會自動將請求轉送到可用區域內的新裝置。這種備援有助於確保存取可用區域內的資料不間斷。

S3 Express One Zone 非常適合需要盡可能降低存取物件所需延遲的任何應用程式。這類應用程式可以是人類互動式工作流程，例如視訊編輯，創意專業人員需要從使用者介面回應存取內容。S3 Express One Zone 也有利於資料具有類似回應能力需求的分析和機器學習工作負載，特別是需進行許多小量存取或大量隨機存取的工作負載。S3 Express 單一區域可與其他區域搭配使用， AWS 服務 以支援分析和人工智慧以及機器學習 (AI/ML) 工作負載，例如 Amazon EMR SageMaker、Amazon 和 Amazon Athena。



使用 S3 Express One Zone 時，您可以使用閘道 VPC 端點與虛擬私有雲 (VPC) 中的目錄儲存貯體互動。使用閘道端點，您可以從 VPC 存取 S3 Express One Zone 目錄儲存貯體，而無需為虛擬私人雲端使用網際網路閘道或 NAT 裝置，而且無需支付額外費用。

您可以將許多相同的 Amazon S3 API 操作和功能與一般用途儲存貯體和其他儲存類別搭配使用的目錄儲存貯體使用。這些包括適用於 Amazon S3 的掛載點、使用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密、S3 Batch Operations，以及 S3 封鎖公開存取。您可以使用 Amazon S3 主控台 AWS Command Line Interface (AWS CLI)、 AWS 開發套件和 Amazon S3 REST API 存取 S3 快速單一區域。

如需有關 S3 Express One Zone 的詳細資訊，請參閱下方主題。

- 概觀
- S3 Express One Zone 的功能
- 相關服務
- 後續步驟

概觀

為了最佳化效能並降低延遲，S3 Express One Zone 導入了下列新概念。

單一可用區域

Amazon S3 快速單區域儲存類別的設計可在單一可用區域內達到 99.95% 的可用性，並受到 Amazon S3 服務水準協議的支援。使用 S3 Express One Zone，您的資料會以備援方式儲存在單一可用區域內的多部裝置上。S3 Express One Zone 的設計在於透過快速偵測並修復任何遺失的備援來處理並行裝置故障。如果現有裝置發生故障，S3 Express One Zone 會自動將請求轉送到可用區域內的新裝置。這種備援有助於確保存取可用區域內的資料不間斷。

可用區域是一個或多個獨立的資料中心，具備 AWS 區域中的備援電源、聯網和連線能力。建立目錄值區時，請選擇可用區域 AWS 區域 以及值區的所在位置。

目錄儲存貯體

Amazon S3 儲存貯體有兩種類型：S3 一般用途儲存貯體和 S3 目錄儲存貯體。一般用途儲存貯體是預設 Amazon S3 儲存貯體類型，適用於絕大多數的 S3 使用案例。目錄儲存貯體僅使用 S3 Express One Zone 儲存類別，這是專為需要一致的個位數毫秒延遲的工作負載或效能關鍵應用程式所設計的類別。選擇最適合您應用和效能需求的儲存貯體類型。

目錄值區會以階層方式將資料組織到目錄中，而不是一般用途值區的平面儲存結構。目錄儲存貯體沒有字首限制，個別目錄可以水平擴展。

目錄儲存貯體使用 S3 Express One Zone 儲存類別，這是專供效能敏感應用程式使用所打造的類別。使用 S3 Express One Zone，您可以選取單一可用區域，並選擇將物件儲存與運算資源共置，以提供最高的存取速度。這與通用值區不同，通用值區會以冗餘方式將物件儲存在. AWS 區域

如需有關目錄儲存貯體的詳細資訊，請參閱 目錄值區。如需有關一般用途儲存貯體的詳細資訊，請參閱 儲存貯體概觀。

端點和閘道 VPC 端點

目錄儲存貯體的儲存貯體管理 API 作業可透過區域端點取得，稱為區域端點 API 作業。區域端點 API 操作的範例包括 `CreateBucket` 和 `DeleteBucket`。建立目錄儲存貯體後，您可以使用區域 (Zone) 端點 API 操作來上傳和管理目錄儲存貯體中的物件。區域端點 API 操作可透過區域端點提供使用。區域端點 API 操作的範例包括 `PutObject` 和 `CopyObject`。

您可以使用閘道虛擬私人雲端端點，從 VPC 存取 S3 Express 單一區域。建立閘道端點後，您可以將其新增為路由表中的目標，用於從 VPC 到 S3 Express One Zone 的流量。如同使用 Amazon S3，使用閘道端點不需額外付費。如需如何設定閘道 VPC 端點的詳細資訊，請參閱 S3 Express One Zone 的網路功能

工作階段型授權

使用 S3 Express One Zone，您可以透過新的工作階段型機制來驗證和授權請求，該機制經過最佳化以提供最低延遲。您可以使用 `CreateSession` 請求臨時憑證來提供低延遲的儲存貯體存取。這些臨時憑證的範圍會設為特定 S3 目錄儲存貯體。工作階段權杖只能與區域 (物件層級) 作業搭配使用 (除了)。CopyObject如需詳細資訊，請參閱 `CreateSession` 授權。

S3 Express One 區域支援的 AWS SDK 會代表您處理工作階段建立和重新整理。為保護您的工作階段，臨時安全憑證會在 5 分鐘後過期。下載並安裝 AWS SDK 並設定必要 AWS Identity and Access Management (IAM) 許可後，您可以立即開始使用 API 操作。

S3 Express One Zone 的功能

下列 S3 功能適用於 S3 Express One Zone。如需支援 API 作業和不受支援功能的完整清單，請參閱。S3 Express One Zone 有什麼不同？

存取管理與安全性

使用目錄儲存貯體時，您可以使用下列功能來稽核和管理存取。根據預設，目錄儲存貯體為私有，只有明確獲得存取權的使用者才能存取。與可在儲存貯體、字首或物件標籤層級設定存取控制界限的一般用途儲存貯體不同的是，目錄儲存貯體的存取控制界限只會在儲存貯體層級設定。如需詳細資訊，請參閱 AWS Identity and Access Management 適用於 S3 快遞單一區域的 (IAM)。

- S3 區塊公開存取 — 依預設會在儲存貯體層級啟用所有 S3 區塊公開存取設定。此預設設定無法修改。
- S3 物件擁有權 (預設強制執行儲存貯體擁有者) — 目錄儲存貯體不支援存取控制清單 (ACL)。目錄儲存貯體會自動使用 S3 物件擁有權的儲存貯體擁有者強制設定 強制執行值區擁有者表示 ACL 已停用，而值區擁有者會自動擁有並完全控制值區中的每個物件。此預設設定無法修改。
- AWS Identity and Access Management (IAM) — IAM 可協助您安全地控制目錄值區的存取。您可以透過`s3express:CreateSession`動作使用 IAM 授予儲存貯體管理 (區域) API 作業和物件管理 (區域) API 作業的存取權。如需詳細資訊，請參閱 AWS Identity and Access Management 適用於 S3 快遞單一區域的 (IAM)。與物件管理動作不同的是，儲存貯體管理動作不可跨帳戶。只有儲存貯體擁有者可執行這些動作。
- 儲存貯體政策：使用 IAM 型政策語言，為目錄儲存貯體設定以資源為基礎的許可。您也可以使用 IAM 來控制 `CreateSession` API 作業的存取，讓您可以使用區域或物件管理 API 作業。您可以授予相同帳戶或跨帳戶存取區域 API 作業。如需 S3 Express 單一區域許可和政策的相關資訊，請參閱AWS Identity and Access Management 適用於 S3 快遞單一區域的 (IAM)。

- 適用於 S3 的 IAM 存取分析器 — 評估和監控您的存取政策，以確保政策僅提供對 S3 資源的預期存取權。

日誌記錄和監控

S3 Express One Zone 使用下列 S3 記錄和監控工具，您可以使用這些工具來監控和控制資源的使用方式：

- Amazon CloudWatch 指標 — 使用收集和追蹤指標 CloudWatch 來監控您的 AWS 資源和應用程式。S3 Express 單一區域使用與其他 Amazon S3 儲存類別 (`AWS/S3`) 相同的 CloudWatch 命名空間，並支援目錄儲存貯體的每日儲存指標：`BucketSizeBytes`和`NumberOfObjects`。如需詳細資訊，請參閱 使用 Amazon 監控指標 CloudWatch。
- AWS CloudTrail 日誌 — AWS CloudTrail AWS 服務 是一種可幫助您 AWS 帳戶 通過記錄用戶，角色或 AWS 服務. 對於 S3 Express 單一區域， CloudTrail擷取區域端點 API 操作 (例如`CreateBucket`和`PutBucketPolicy`) 做為管理事件。這些事件包括在 AWS Management Console、 AWS Command Line Interface (AWS CLI)、 AWS SDK 和 AWS API 作業中採取的動作。`eventsource`適用於 S3 快速單一區域的 CloudTrail 管理事件是`s3express.amazonaws.com`。如需詳細資訊，請參閱 Amazon S3 CloudTrail 活動。

> **Note**
>
> S3 快速單一區域不支援 Amazon S3 伺服器存取日誌。

物件管理

建立目錄儲存貯體後，您可以使用 Amazon S3 主控台、 AWS 開發套件和 AWS CLI. 下列功能適用於 S3 Express 單區域的物件管理：

- S3 Batch 操作 — 使用 Batch 操作對目錄儲存貯體中的物件執行批次作業，例如複製和叫用 AWS Lambda 函數。例如，您可以使用 Batch Operations 在目錄儲存貯體和一般用途儲存貯體之間複製物件。透過 Batch 操作，您可以使用 AWS 開發套件 AWS CLI 或在 Amazon S3 主控台按幾下，透過單一 S3 請求大規模管理數十億個物件。
- 匯入：建立目錄儲存貯體之後，您可以使用 Amazon S3 主控台中的匯入功能將物件填入儲存貯體。匯入是用來建立 Batch Operations 任務的簡化方法，可將物件從一般用途儲存貯體複製到目錄儲存貯體。

AWS SDK 和用戶端程式庫

建立目錄值區並將物件上傳至值區後，您可以使用下列指令來管理物件儲存空間。

- 適用於 Amazon S3 的掛載點 — Amazon S3 的掛載點是開放原始碼檔案用戶端，可提供高輸送量存取，並降低 Amazon S3 上資料湖的運算成本。Amazon S3 的掛載點會將本機檔案系統 API 呼叫轉換為 S3 物件 API 呼叫 (例如和)。GET LIST對於處理數 PB 資料且需要 Amazon S3 提供的高彈性輸送量來擴展和縮減數千個執行個體，這些資料湖工作負載非常適合讀取密集型資料湖工作負載。
- S3A— S3A 是用於存取 Amazon S3 中資料存放區的建議Hadoop相容界面。 S3A取代S3NHadoop檔案系統用戶端。
- PyTorchon AWS— PyTorch on AWS 是開放原始碼深度學習架構，可讓您更輕鬆地開發機器學習模型並將其部署至生產環境。
- AWS 開發套件 — 您可以在使用 Amazon S3 開發應用程式時使用開發 AWS 套件。這些 AWS 開發套件透過包裝基礎的 Amazon S3 REST API 來簡化您的程式設計任務。如需將 AWS 開發套件與 S3 快速單區域搭配使用的詳細資訊，請參閱the section called “AWS 開發套件”。

加密和資料保護

存放在目錄儲存貯體中的物件會使用伺服器端加密搭配 Amazon S3 受管金鑰 (SSE-S3) 自動加密。目錄儲存貯體不支援使用 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS) 進行伺服器端加密、使用客戶提供的加密金鑰 (SSE-C) 進行伺服器端加密，或使用 (DSSE-KMS) 的雙層伺服器端加密。 AWS KMS keys 如需詳細資訊，請參閱 資料保護和加密 及 使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密。

S3 Express One Zone 可讓您選擇用於在上傳或下載過程中驗證資料的檢查總和演算法選項。
您可以選擇以下 Secure Hash 演算法 (SHA) 或循環冗餘檢查 (CRC) 資料完整性演算法之一：
CRC32、CRC32C、SHA-1 和 SHA-256。S3 快速單區域儲存類別不支援以 MD5 為基礎的總和檢查碼。

如需詳細資訊，請參閱 S3 其他檢查總和最佳實務。

AWS 簽名版本 4 (SigV4)

S3 快速單區使用 AWS 簽名版本 4 (SigV4)。 SigV4這是一種簽署通訊協定，用來驗證透過 HTTPS 傳送給 Amazon S3 的請求。S3 快遞一個區域使用簽署請求 AWS Sigv4。如需詳細資訊，請參閱 Amazon 簡單儲存服務 API 參考中的驗證請求 (AWS 簽名版本 4)。

高度的一致性

S3 Express One Zone 區域為所有目錄儲存貯體中的物件提供了強大的一 read-after-write 致性PUT和DELETE請求 AWS 區域。如需詳細資訊，請參閱 Amazon S3 資料一致性模式。

相關服務

您可以將下列項目 AWS 服務 與 S3 Express 單區儲存類別搭配使用，以支援特定的低延遲使用案例。

- Amazon Elastic Compute Cloud (Amazon EC2) — 亞馬遜 EC2 在 AWS 雲端. 使用 Amazon EC2 可減少前期所需的硬體投資，讓您更快速開發並部署應用程式。您可使用 Amazon EC2 按需要啟動任意數量的虛擬伺服器，設定安全性和聯網功能以及管理儲存。
- AWS Lambda：Lambda 是一項運算服務，可讓您執行程式碼，而無需佈建或管理伺服器。您在儲存貯體上設定通知設定，並授予 Amazon S3 許可以在函數以資源為基礎的許可政策上叫用函數。
- Amazon Elastic Kubernetes Service (Amazon EKS) — Amazon EKS 是一種受管服務，無需安裝、操作和維護自己的控制平面。Kubernetes AWSKubernetes是一種開放原始碼系統，可自動化容器化應用程式的管理、擴展和部署。
- Amazon Elastic Container Service (Amazon ECS)：Amazon ECS 為全受管容器協同運作服務，可讓您輕鬆部署、管理和擴展容器化應用程式。
- Amazon Athena：Athena 是一種互動式查詢服務，可讓您使用標準 SQL 直接輕鬆分析 Amazon S3 中的資料。您也可以使用 Athena 以互動方式執行資料分析，Apache Spark無需規劃、設定或管理資源。當您在 Athena 上執行Apache Spark應用程式時，您可以提交程式Spark碼進行處理，並直接接收結果。
- Amazon SageMaker 執行階段模型訓練 — Amazon SageMaker 執行階段是全受管的機器學習服務。透過 SageMaker Runtime，資料科學家和開發人員可以快速輕鬆地建置和訓練機器學習模型，然後將它們直接部署到生產就緒的託管環境中。
- AWS Glue— AWS Glue 是一項無伺服器資料整合服務，可讓分析使用者輕鬆探索、準備、移動和整合來自多個來源的資料。您可以用 AWS Glue 於分析、機器學習和應用程式開發。 AWS Glue 還包括用於編寫、執行工作和實作業務工作流程的額外生產力和資料作業工具。
- Amazon EMR — Amazon EMR 是一個受管叢集平台，可簡化執行大數據架構 (例如Apache Hadoop和)Apache Spark， AWS 以處理和分析大量資料。

後續步驟

如需有關使用 S3 Express One Zone 儲存類別與目錄儲存貯體的詳細資訊，請參閱下方主題：

S3 Express One Zone 有什麼不同？

Amazon S3 Express One Zone 是一種高效能的單一區域 Amazon S3 儲存類別，專門為對延遲最敏感的應用程式提供一致的個位數毫秒資料存取。S3 Express One Zone 是第一款可讓您選取單一可用區域的 S3 儲存類別，還可選擇將物件儲存體與運算資源共置，藉此盡可能提供最高存取速度。此外，為了進一步提高存取速度並支援每秒數十萬個請求，S3 Express One Zone 資料會以新的儲存貯體類型儲存：Amazon S3 目錄儲存貯體。

如需詳細資訊，請參閱 什麼是 S3 Express One Zone？ 及 目錄值區。

您可以使用 Amazon S3 API 建立目錄儲存貯體，並存取 S3 Express One Zone 中的資料。Amazon S3 API 與 S3 Express One Zone 和目錄儲存貯體相容，但有幾項顯著的差異。如需有關 S3 Express One Zone 差異之處的詳細資訊，請參閱下方主題。

主題

S3 Express One Zone 差異之處

- 支援的儲存貯體類型：僅限在目錄儲存貯體中儲存 S3 Express One Zone 儲存類別中的物件。如需詳細資訊，請參閱 目錄值區。
- 耐久性：使用 S3 Express One Zone，您的資料會以備援方式儲存在單一可用區域內的多部裝置上。S3 Express One Zone 的設計可在單一可用區域內達到 99.95% 的可用性，並依照 Amazon S3 服務水準協議提供支援。如需詳細資訊，請參閱 單一可用區域。

- **`ListObjectsV2`行為**
  - 對於目錄值區，`ListObjectsV2`不會以字母順序傳回物件。此外，字首的結尾必須是分隔符號，並且只能指定「/」作為分隔符號。
  - 對於目錄值區，`ListObjectsV2`回應會包含僅與進行中多部分上傳相關的前置詞。
- 刪除行為：當您刪除目錄儲存貯體中的物件時，Amazon S3 會週期性地刪除物件路徑中的所有空目錄。例如，如果您刪除物件金鑰`dir1/dir2/file1.txt`，Amazon S3 就會刪除`file1.txt`。如果 `dir1/` 和 `dir2/` 目錄是空的且未包含任何其他物件，則 Amazon S3 也會刪除這些目錄。
- ETag 和檢查總和：S3 Express One Zone 的實體標籤 (ETag) 是隨機英數字串，而非 MD5 檢查總和。如需有關在 S3 Express One Zone 使用其他檢查總和的詳細資訊，請參閱 S3 其他檢查總和最佳實務。
- **`DeleteObjects`** 請求中的物件索引鍵
  - `DeleteObjects` 請求中的物件索引鍵必須至少包含一個非空格字元。`DeleteObjects` 請求中不支援只包含空格字元的字串。
  - `DeleteObjects` 請求中的物件索引鍵不得包含 Unicode 控制字元，但新增一行 (\n)、tab 鍵 (\t) 和換行 (\r) 字元除外。
- 區域 (Region) 和區域 (Zone) 端點：使用 S3 Express One Zone 時，您必須在所有用戶端請求中指定區域。對於區域 (Region) 端點，您可以指定「區域」，例如 `s3express-control.us-west-2.amazonaws.com`。對於區域 (Zone) 端點，您可以指定「區域」和「可用區域」，例如 `s3express-usw2-az1.us-west-2.amazonaws.com`。如需詳細資訊，請參閱 區域和區域端點。
- 分段上傳：如同儲存在 Amazon S3 中的其他物件，您可以使用分段上傳程序來上傳和複製儲存在 S3 Express One Zone 儲存類別中的大型物件。然而，以下是對儲存在 S3 Express One Zone 中的物件使用分段上傳程序時的一些差異。如需詳細資訊，請參閱 the section called “搭配目錄儲存貯體使用多部分上傳”。
  - 物件建立日期是指分段上傳的完成日期。
  - 分段的各段編號必須使用連續的分段編號。如果您嘗試使用非連續分段編號完成分段上傳請求，則 Amazon S3 會產生 HTTP `400 (Bad Request)` 錯誤。
  - 分段上傳的啟動器只有在透過 `s3express:CreateSession` 許可授予明確允許存取 `AbortMultipartUpload` 的情況下，才能中止分段上傳請求。如需詳細資訊，請參閱 AWS Identity and Access Management 適用於 S3 快遞單一區域的 (IAM)。
- 清空目錄值區 — 透過 AWS Command Line Interface (CLI) 執行的`s3 rm`命令、透過掛載點執行的`delete`作業，以及透過的 [清空值區] 選項按鈕，無法刪除 AWS Management Console 目錄值區中進行中的多部分上傳。若要刪除這些進行中的分段上傳，請使用此`ListMultipartUploads`作業

在值區中列出進行中的多部分上傳，並使用此`AbortMultupartUpload`作業中止所有進行中的分段上傳。

S3 Express One Zone 支援的 API 操作

Amazon S3 Express One Zone 儲存類別會支援區域 (Region) 端點 API 操作 (儲存貯體層級，或控制平面) 和區域 (Zone) 端點 API 操作 (物件層級，或資料平面)。如需詳細資訊，請參閱 S3 Express One Zone 的網路功能 及 端點和閘道 VPC 端點。

區域 (Region) 端點 API 操作

S3 Express One Zone 支援下列區域 (Region) 端點 API 操作：

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy
- ListDirectoryBuckets
- PutBucketPolicy

區域 (Zone) 端點 API 操作

S3 Express One Zone 支援下列區域 (Zone) 端點 API 操作：

- CreateSession
- CopyObject
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2
- PutObject

- AbortMultipartUpload
- CompleteMultiPartUpload
- CreateMultipartUpload
- ListMultipartUploads
- ListParts
- UploadPart
- UploadPartCopy

S3 Express One Zone 不支援的 Amazon S3 功能

S3 Express One Zone 不支援下列 Amazon S3 功能：

- AWS CloudTrail 資料平面事件
- AWS 受管政策
- AWS PrivateLink 適用於 S3
- MD5 檢查總和
- 多重要素驗證 (MFA) 刪除
- S3 物件鎖定
- 請求者付款
- S3 存取授權
- S3 存取點
- 儲存貯體標籤
- Amazon CloudWatch 請求指標
- S3 事件通知
- S3 生命週期
- S3 多區域存取點
- S3 Object Lambda 存取點
- S3 版本控制
- S3 庫存
- S3 複寫

- 物件標籤
- S3 Select
- 伺服器存取日誌
- 靜態網站託管
- S3 Storage Lens
- S3 Storage Lens 群組
- S3 Transfer Acceleration
- 使用 AWS Key Management Service (AWS KMS) 金鑰 (DSSE-KMS) 進行雙層伺服器端加密
- 使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 進行伺服器端加密
- 使用客戶提供金鑰 (SSE-C) 的伺服器端加密
- 在中建立新值區時複製現有值區設定的選項 AWS Management Console。

開始使用 S3 Express One Zone

下節說明如何開始使用 Amazon S3 Express One Zone 儲存類別和目錄儲存貯體。如需詳細資訊，請參閱 什麼是 S3 Express One Zone？。

主題

- 使用 S3 快速單一區域設定 AWS Identity and Access Management (IAM)
- 設定閘道 VPC 端點
- 使用 S3 主控台和 AWS 開發套件使用 S3 快速單區域 AWS CLI

使用 S3 快速單一區域設定 AWS Identity and Access Management (IAM)

AWS Identity and Access Management (IAM) 可協助管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員可控制哪些人員可進行身分驗證 (登入) 並獲得授權 (具有許可)，以使用 S3 Express One Zone 中的 Amazon S3 資源。您可以免費使用 IAM。

根據預設，使用者未具備執行目錄儲存貯體和 S3 Express One Zone 操作的許可。若要授予存取目錄儲存貯體和 S3 Express One Zone 操作的許可，您可以使用 IAM 建立使用者或角色，並將許可附加至這些身分。

若要開始使用 IAM，請參閱 AWS Identity and Access Management 適用於 S3 快遞單一區域的 (IAM) 和 S3 Express One Zone 的 IAM 身分型政策。

設定閘道 VPC 端點

若要存取 S3 Express One Zone，可使用與標準 Amazon S3 端點不同的區域 (Region) 和區域 (Zone) 端點。需要區域 (Zone) 或區域 (Region) 端點，取決於您使用的 Amazon S3 API 操作。如需依端點類型的受支援 API 操作完整清單，請參閱 S3 Express One Zone 支援的 API 操作。您必須透過閘道虛擬私有雲端 (VPC) 端點存取區域 (Zone) 和區域 (Region) 端點。若要設定閘道端點，請參閱 S3 Express One Zone 的網路功能。

使用 S3 主控台和 AWS 開發套件使用 S3 快速單區域 AWS CLI

您可以使用 AWS 開發套件、Amazon S3 主控台、 AWS Command Line Interface (AWS CLI) 和 Amazon S3 REST API，使用 S3 快速單區域儲存類別和目錄儲存貯體。

S3 主控台

請依照下列步驟開始使用 S3 主控台：

- 建立目錄儲存貯體
- 清空目錄儲存貯體
- 刪除目錄儲存貯體

AWS 開發套件

S3 快速單一區域支援下列 AWS 開發套件：

- AWS SDK for C++
- AWS SDK for Go V2
- AWS SDK for Java 2.x
- AWS SDK for JavaScript V3
- AWS SDK for .NET
- AWS SDK for PHP
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby
- 適用於 Kotlin 的 AWS SDK
- 適用於 Rust 的 AWS SDK

使用 S3 Express One Zone 時，建議您使用最新版的 AWS SDK。S3 Express One Zone 支援的 AWS SDK 會代表您處理工作階段建立、更新和終止。這表示您可以在下載並安裝 AWS SDK 並設定必要的 IAM 許可後立即開始使用 API 作業。如需詳細資訊，請參閱 AWS Identity and Access Management 適用於 S3 快遞單一區域的 (IAM)。

如需 AWS SDK 的相關資訊，包括如何下載和安裝它們，請參閱建置的 AWS工具。

如需 AWS SDK 範例，請參閱下列內容：

- 建立目錄儲存貯體
- 清空目錄儲存貯體
- 刪除目錄儲存貯體

AWS Command Line Interface (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 建立目錄儲存貯體，並針對 S3 Express 單一區域使用支援的區域和區域端點 API 操作。

若要開始使用 AWS CLI，請參閱《AWS CLI 指令參考》AWS CLI中的〈開始使用〉。

> **Note**
>
> 若要搭配高階aws s3指令使用目錄值區，請 AWS CLI 將您的目錄值區更新為最新版本。如需有關如何安裝和規劃的詳細資訊 AWS CLI，請參閱《AWS CLI 命令參考》 AWS CLI中的安裝或更新最新版本的。

如需 AWS CLI 範例，請參閱下列內容：

- 建立目錄儲存貯體
- 清空目錄儲存貯體
- 刪除目錄儲存貯體

S3 Express One Zone 的網路功能

若要存取 Amazon S3 Express One Zone 儲存類別物件和目錄儲存貯體，可使用與標準 Amazon S3 端點不同的區域 (Region) 和區域 (Zone) API 端點。需要區域 (Zone) 或區域 (Region) 端點，取決於您

使用的 S3 API 操作。如需依端點類型列出的完整 API 操作清單，請參閱 S3 Express One Zone 支援的 API 操作。

您可以透過閘道虛擬私有雲端 (VPC) 端點存取區域 (Zone) 和區域 (Region) API 操作。若要設定閘道 VPC 端點，請參閱 the section called “設定 VPC 閘道端點”。

下列主題說明使用閘道 VPC 端點存取 S3 Express One Zone 的網路功能需求。

主題

端點

您可以使用閘道 VPC 端點從 VPC 存取 Amazon S3 Express One Zone 儲存類別物件和目錄儲存貯體。S3 Express One Zone 使用區域 (Region) 和區域 (Zone) API 端點。需要區域 (Region) 或區域 (Zone) 端點，取決於您使用的 Amazon S3 API 操作。使用閘道端點不需額外付費。

儲存貯體層級 (也就是控制平面) API 操作可透過區域 (Region) 端點提供使用，並且稱為區域端點 API 操作。區域端點 API 操作的範例包括 `CreateBucket` 和 `DeleteBucket`。當您建立目錄儲存貯體時，可以選擇要在其中建立目錄儲存貯體的單一可用區域。建立目錄儲存貯體後，您可以使用區域 (Zone) 端點 API 操作來上傳和管理目錄儲存貯體中的物件。

物件層級 (也就是資料平面) API 操作可透過區域 (Zone) 端點提供使用，並且稱為區域端點 API 操作。區域端點 API 操作的範例包括 `CreateSession` 和 `PutObject`。

下表顯示每個區域和可用區域可用的區域 (Region) 和區域 (Zone) API 端點。

設定 VPC 閘道端點

下列程序可用來建立連線至 Amazon S3 Express One Zone 儲存類別物件和目錄儲存貯體的閘道端點。

設定閘道 VPC 端點

1. 開啟 Amazon VPC 主控台，位於 https://console.aws.amazon.com/vpc/。
2. 在導覽窗格中選擇 Endpoints (端點)。
3. 選擇 建立端點。
4. 建立端點的名稱。

5. 對於 Service category (服務類別)，選擇 AWS 服務。
6. 針對服務，新增篩選條件 Type=Gateway，然後選擇 com.amazonaws.***region***.s3express 旁邊的選項按鈕。
7. 針對 VPC，選擇要在其中建立端點的 VPC。
8. 針對 Route tables (路由表)，選取要供端點使用的路由表。Amazon VPC 會自動新增路由，將以服務為目標的流量指向端點網路介面。
9. 針對政策，選擇完整存取，以允許 VPC 端點上所有資源的所有主體進行所有操作。否則，選擇自訂以連接 VPC 端點政策，該政策會控制主體在 VPC 端點上對資源執行操作所擁有的許可。
10. (選用) 若要新增標籤，請選擇新增標籤，然後輸入標籤金鑰和標籤值。
11. 選擇 建立端點。

建立閘道端點後，您可以使用區域 (Region) API 端點和區域 (Zone) API 端點存取 Amazon S3 Express One Zone 儲存類別物件和目錄儲存貯體。

目錄值區

有兩種類型的 Amazon S3 儲存貯體：一般用途儲存貯體和目錄儲存貯體。請選擇最適合您的應用程式和效能需求的儲存貯體類型。

- 一般用途儲存貯體是原始 S3 儲存貯體類型，建議用於大多數使用案例和存取模式。一般用途儲存貯體也允許跨所有儲存類別儲存的物件，但 S3 Express One Zone 除外。
- 目錄儲存貯體使用 S3 Express One Zone 儲存類別，如果您的應用程式對於效能很敏感，且受益於個位數毫秒的 PUT 和 GET 延遲，則建議使用此類別。

目錄儲存貯體用於需要一致的個位數毫秒延遲的工作負載或效能關鍵應用程式。目錄儲存貯體會以階層方式將資料組織到目錄中，不同於一般用途儲存貯體的平面儲存結構。目錄儲存貯體沒有字首限制，個別目錄可以水平擴展。

目錄儲存貯體使用 S3 Express One Zone 儲存類別，該類別可將資料存放在單一可用區域內的多個裝置，但不會以備援方式跨可用區域存放資料。建立目錄儲存貯體時，建議您指定 Amazon EC2 AWS 區域、Amazon 彈性 Kubernetes 服務或 Amazon 彈性容器服務 (Amazon ECS) 運算執行個體的本機可用區域，以優化效能。

您可以在每個目錄值區中建立最多 10 個目錄值區 AWS 帳戶，而且值區中可以儲存的物件數目沒有限制。您的儲存貯體配額會套用至 AWS 帳戶中的每個區域。如果您的應用程序需要提高此限制，請聯繫 AWS Support。如需詳細資訊，請造訪 Ser vice Quotas 主控台。

> **Important**
>
> 至少 90 天沒有要求活動的目錄值區，會轉換為非作用中狀態。處於非作用中狀態時，會暫時無法存取目錄儲存貯體來進行讀取和寫入。非作用中儲存貯體會保留所有儲存、物件中繼資料和儲存貯體中繼資料。現有儲存體費用適用於非作用中值區。如果您對非作用中值區發出存取要求，值區會在幾分鐘內轉換為使用中狀態。在此轉換期間，讀取和寫入會傳回 HTTP 503 (Service Unavailable) 錯誤碼。

下列主題提供目錄儲存貯體的相關資訊。如需有關一般用途儲存貯體的詳細資訊，請參閱 儲存貯體概觀。

主題

可用區域

當您建立目錄儲存貯體時，可選擇可用區域和 AWS 區域。

目錄儲存貯體使用 S3 Express One Zone 儲存類別，這是專供效能敏感應用程式使用所打造的類別。S3 Express One Zone 是第一款可讓您選取單一可用區域的 S3 儲存類別，還可選擇將物件儲存體與運算資源共置，藉此盡可能提供最高存取速度。

使用 S3 Express One Zone，您的資料會以備援方式儲存在單一可用區域內的多部裝置上。S3 快速單一區域的設計可在單一可用區域內達到 99.95% 的可用性，並受到 Amazon S3 服務等級協議的支援。如需更多資訊，請參閱單一可用區域

目錄儲存貯體的名稱

目錄儲存貯體名稱包含您提供的基本名稱，以及包含儲存貯體所在之可用區域 ID 的字尾。目錄儲存貯體名稱必須使用下列格式，並遵循目錄儲存貯體的命名規則：

```
bucket-base-name--azid--x-s3
```

例如，下列目錄儲存貯體名稱包含可用區域 ID `usw2-az1`：

```
bucket-base-name--usw2-az1--x-s3
```

如需詳細資訊，請參閱 目錄儲存貯體命名規則。

目錄

目錄儲存貯體會以階層方式將資料組織到目錄中，不同於一般用途儲存貯體的平面儲存結構。每個 S3 目錄儲存貯體每秒可支援數十萬筆交易 (TPS)，不受儲存貯體內的目錄數目影響。

使用階層式命名空間時，物件索引鍵中的分隔符號非常重要。唯一支援的分隔符號為正斜線 (`/`)。目錄是以分隔符號邊界決定。例如，物件索引鍵 `dir1/dir2/file1.txt` 會產生目錄 `dir1/` 並自動建立 `dir2/`，以及將物件 `file1.txt` 新增至路徑 `dir1/dir2/file1.txt` 中的 `/dir2` 目錄。

目錄儲存貯體索引模型會針對 `ListObjectsV2` API 操作傳回未排序的結果。如果您需要將結果限於儲存貯體的某個子區段，您可以在 `prefix` 參數中指定子目錄路徑，例如 `prefix=dir1/`。

鍵值名稱

對於目錄儲存貯體，多個物件索引鍵常用的子目錄會使用第一個物件索引鍵來建立。相同子目錄的其他物件索引鍵會使用先前建立的子目錄。此模型可讓您彈性選擇最適合應用程式的物件索引鍵，同時支援稀疏和密集目錄。

存取管理

目錄儲存貯體預設會在儲存貯體層級啟用所有 S3 封鎖公開存取設定。S3 物件擁有權會設定為儲存貯體擁有者強制執行，且存取控制清單 (ACL) 會停用。這些設定無法修改。

根據預設，使用者未具備執行目錄儲存貯體和 S3 Express One Zone 操作的許可。若要授予存取目錄儲存貯體的許可，您可以使用 IAM 建立使用者、群組或角色，並將許可附加至這些身分。如需詳細資訊，請參閱 S3 快速單一區域的AWS Identity and Access Management (IAM)。

使用目錄儲存貯體

如需使用目錄儲存貯體的詳細資訊，請參閱下方主題。

主題

- 目錄儲存貯體命名規則
- 建立目錄儲存貯體
- 檢視目錄儲存貯體屬性
- 管理目錄儲存貯體的儲存貯體政策
- 清空目錄儲存貯體
- 刪除目錄儲存貯體
- 列出目錄儲存貯體
- HeadBucket搭配目錄值區使用

目錄儲存貯體命名規則

當您在 Amazon S3 中建立目錄儲存貯體時，須遵循下列儲存貯體命名規則。若要了解一般用途儲存貯體命名規則，請參閱 儲存貯體命名規則。

目錄值區名稱是由您提供的基礎名稱，以及包含值區所在之 AWS 可用區域 ID 的尾碼所組成`--x-s3`。

```
base-name--azid--x-s3
```

例如，下列目錄儲存貯體名稱包含可用區域 ID `usw2-az1`：

```
bucket-base-name--usw2-az1--x-s3
```

> **Note**
>
> 當您使用主控台建立目錄值區時，字尾會自動新增至您提供的基本名稱。此字尾包含您所選擇可用區域的可用區域 ID。
> 使用 API 建立目錄儲存貯體時，必須在要求中提供完整尾碼，包括可用區域 ID。如需可用區域 ID 的清單，請參閱S3 Express One Zone 可用區域和區域。

目錄儲存貯體名稱必須：

- 在所選區域 AWS 區域 和可用區域內是唯一的。
- 名稱長度必須介於 3 (最小) 到 63 個 (最多) 個字元之間，包括後置字元。
- 只能由小寫字母、數字和連字號 (-) 組成。
- 開頭和結尾為字母或數字。
- 必須包含以下後綴：`--azid--x-s3`。

建立目錄儲存貯體

若要開始使用 Amazon S3 Express One Zone 儲存類別，請建立目錄儲存貯體。S3 Express One Zone 儲存類別只能搭配目錄儲存貯體使用。S3 Express One Zone 儲存類別支援低延遲使用案例，並可在單一可用區域內提供更快速的資料處理。如果您的應用程式對效能很敏感，且受益於個位數毫秒的 PUT 和 GET 延遲，則建議您建立目錄儲存貯體，以便使用 S3 Express One Zone 儲存類別。

有兩種類型的 Amazon S3 儲存貯體：一般用途儲存貯體和目錄儲存貯體。您應該選擇最適合您的應用程式和效能需求的儲存貯體類型。一般用途儲存貯體是原始 S3 儲存貯體類型。建議大多數使用案例和存取模式使用一般用途儲存貯體，並允許物件存放在所有儲存類別 (S3 Express One Zone 除外)。如需有關一般用途儲存貯體的詳細資訊，請參閱 儲存貯體概觀。

目錄儲存貯體使用 S3 Express One Zone 儲存類別，這是專供需要一致的個位數毫秒延遲的工作負載或效能關鍵應用程式使用所設計的類別。S3 Express One Zone 是第一款可讓您選取單一可用區域的 S3 儲存類別，還可選擇將物件儲存體與運算資源共置，藉此盡可能提供最高存取速度。建立目錄儲存貯體時，您可以選擇性地指定 AWS 區域 和可用區域，該區域位於 Amazon EC2、Amazon 彈性 Kubernetes 服務或 Amazon 彈性容器服務 (Amazon ECS) 運算執行個體的本機執行個體，以優化效能。

使用 S3 Express One Zone，您的資料會以備援方式儲存在單一可用區域內的多部裝置上。S3 快速單一區域的設計可在單一可用區域內達到 99.95% 的可用性，並受到 Amazon S3 服務等級協議的支援。如需更多資訊，請參閱單一可用區域

目錄值區會以階層方式將資料組織到目錄中，而不是一般用途值區的平面儲存結構。目錄儲存貯體沒有字首限制，個別目錄可以水平擴展。

如需有關目錄儲存貯體的詳細資訊，請參閱 目錄值區。

目錄儲存貯體的名稱

目錄儲存貯體名稱必須遵循此格式，並符合目錄儲存貯體的命名規則：

```
bucket-base-name--azid--x-s3
```

例如，下列目錄儲存貯體名稱包含可用區域 ID `usw2-az1`：

```
bucket-base-name--usw2-az1--x-s3
```

如需有關儲存貯體命名規則的詳細資訊，請參閱 目錄儲存貯體命名規則。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在頁面頂端的導覽列中，選擇目前顯示的名稱 AWS 區域。接下來，選擇您要在其中建立值區的「區域」。

   > **Note**
   >
   > 請選擇接近您的區域，以充分降低延遲及成本，並因應法規要求。除非您明確地將存放在區域中的物件傳輸到其他區域，否則物件絕對不會離開該區域。如需 Amazon S3 的清單 AWS 區域，AWS 服務 請參閱 Amazon Web Services 一般參考.

3. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
4. 選擇 Create bucket (建立儲存貯體)。

   Create bucket (建立儲存貯體) 頁面隨即開啟。
5. 在 [一般設定] 下方，檢視 AWS 區域 儲存貯體的建立位置。
6. 在「值區類型」下選擇「目錄」。

> **Note**
>
> - 如果您選擇的區域不支援目錄值區，則「值區」型態選項會消失，且值區型態會預設為一般用途值區。若要建立目錄值區，您必須選擇支援的「地區」。如需支援目錄儲存貯體和 Amazon S3 快速單區域儲存類別的區域清單，請參閱the section called “S3 Express One Zone 可用區域和區域”。
> - 在您建立儲存貯體之後，便無法變更儲存貯體類型。

針對可用區域，選擇運算服務本機上的可用區域。如需支援目錄儲存貯體和 S3 Express 單區儲存類別的可用區域清單，請參閱the section called “S3 Express One Zone 可用區域和區域”。

> **Note**
>
> 儲存貯體建立之後，就無法變更可用區域。

7. 在可用區域下，選取核取方塊，表示您確實了解在可用區域發生中斷時，您的資料可能無法使用或遺失。

> **Important**
>
> 雖然目錄值區儲存在單一可用區域內的多個裝置上，但目錄儲存貯體不會跨可用區域以冗餘方式儲存資料。

8. 針對儲存貯體名稱，輸入您的目錄儲存貯體的名稱。

目錄儲存貯體名稱必須：

- 在所選區域 AWS 區域 和可用區域內是唯一的。
- 名稱長度必須介於 3 (最小) 到 63 個 (最多) 個字元之間，包括後置字元。
- 只能由小寫字母、數字和連字號 (-) 組成。
- 開頭和結尾為字母或數字。
- 必須包含以下後綴：`--azid--x-s3`。

當您使用主控台建立目錄值區時，字尾會自動新增至您提供的基本名稱。此字尾包含您所選擇可用區域的可用區域 ID。

建立儲存貯體後，便無法變更其名稱。如需儲存貯體命名的詳細資訊，請參閱 儲存貯體命名規則。

> **Important**
>
> 請勿在值區名稱中包含敏感資訊，例如帳號。在指向儲存貯體中之物件的 URL 中，會顯示儲存貯體名稱。

9. 在 [物件擁有權] 底下，會自動啟用 [值區擁有者強制執行] 設定，且會停用所有存取控制清單 (ACL)。您無法針對目錄儲存貯體啟用 ACL。

   已停用 ACL

   - 儲存貯體擁有者強制執行 (預設) - 停用 ACL，儲存貯體擁有者會自動擁有並完全控制儲存貯體中的每個物件。ACL 不再影響 S3 儲存貯體中資料的存取許可。儲存貯體單獨使用政策來定義存取控制。

     Amazon S3 中的大多數新式使用案例不再需要使用 ACL。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

10. 在此值區的「封鎖公用存取」設定下，會自動啟用目錄值區的所有「封鎖公用存取」設定。目錄值區的這些設定無法修改。如需封鎖公開存取的詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。
11. 在伺服器端加密設定下，Amazon S3 會使用 Amazon S3 受管金鑰 (SSE-S3) 套用伺服器端加密，做為所有 S3 儲存貯體的加密基礎層級。所有上傳至目錄儲存貯體的物件都會使用 SSE-S3 加密。對於目錄值區，無法修改加密類型。如需 SSE-S3 的詳細資訊，請參閱 the section called “Amazon S3 受管加密金鑰 (SSE-S3)”。
12. 選擇建立儲存貯體。

    建立儲存貯體之後，您可以新增檔案和資料夾至儲存貯體。如需詳細資訊，請參閱 the section called “使用目錄儲存貯體中的物件”。

使用 AWS 軟體開發套件

SDK for Go

此範例顯示如何使用建立目錄值區 AWS SDK for Go。

Example

```
var bucket = "..."

func runCreateBucket(c *s3.Client) {
    resp, err := c.CreateBucket(context.Background(), &s3.CreateBucketInput{
        Bucket: &bucket,
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            Location: &types.LocationInfo{
                Name: aws.String("usw2-az1"),
                Type: types.LocationTypeAvailabilityZone,
            },
            Bucket: &types.BucketInfo{
                DataRedundancy: types.DataRedundancySingleAvailabilityZone,
                Type:           types.BucketTypeDirectory,
            },
        },
    })
    var terr *types.BucketAlreadyOwnedByYou
    if errors.As(err, &terr) {
        fmt.Printf("BucketAlreadyOwnedByYou: %s\n", aws.ToString(terr.Message))
        fmt.Printf("noop...\n")
        return
    }
    if err != nil {
        log.Fatal(err)
    }

    fmt.Printf("bucket created at %s\n", aws.ToString(resp.Location))
}
```

SDK for Java 2.x

此範例顯示如何使用建立目錄值區 AWS SDK for Java 2.x。

Example

```
public static void createBucket(S3Client s3Client, String bucketName) {

    //Bucket name format is {base-bucket-name}--{az-id}--x-s3
    //example: doc-example-bucket--usw2-az1--x-s3 is a valid name for a directory
 bucket created in
    //Region us-west-2, Availability Zone 2
```

```
   CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
            .location(LocationInfo.builder()
                    .type(LocationType.AVAILABILITY_ZONE)
                    .name("usw2-az1").build()) //this must match the Region and
Availability Zone in your bucket name
            .bucket(BucketInfo.builder()
                   .type(BucketType.DIRECTORY)
                   .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                   .build()).build();
   try {

            CreateBucketRequest bucketRequest =
CreateBucketRequest.builder().bucket(bucketName).createBucketConfiguration(bucketConfigurat
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            System.out.println(response);
   }

   catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
        }
   }
```

AWS SDK for JavaScript

此範例顯示如何使用建立目錄值區 AWS SDK for JavaScript。

Example

```
// file.mjs, run with Node.js v16 or higher
// To use with the preview build, place this in a folder
// inside the preview build directory, such as /aws-sdk-js-v3/workspace/

import { S3 } from "@aws-sdk/client-s3";

const region = "us-east-1";
const zone = "use1-az4";
const suffix = `${zone}--x-s3`;
```

```
const s3 = new S3({ region });

const bucketName = `...--${suffix}`;

const createResponse = await s3.createBucket(
    { Bucket: bucketName,
      CreateBucketConfiguration: {Location: {Type: "AvailabilityZone", Name: zone},
      Bucket: { Type: "Directory", DataRedundancy: "SingleAvailabilityZone" }}
    }
   );
```

AWS SDK for .NET

此範例顯示如何使用建立目錄值區 AWS SDK for .NET。

Example

```
using (var amazonS3Client = new AmazonS3Client())
{
    var putBucketResponse = await amazonS3Client.PutBucketAsync(new PutBucketRequest
    {

       BucketName = "DOC-EXAMPLE-BUCKET--usw2-az1--x-s3",
       PutBucketConfiguration = new PutBucketConfiguration
       {
         BucketInfo = new BucketInfo { DataRedundancy =
 DataRedundancy.SingleAvailabilityZone, Type = BucketType.Directory },
         Location = new LocationInfo { Name = "usw2-az1", Type =
 LocationType.AvailabilityZone }
       }
     }).ConfigureAwait(false);
}
```

SDK for PHP

此範例顯示如何使用建立目錄值區 AWS SDK for PHP。

Example

```
require 'vendor/autoload.php';

$s3Client = new S3Client([
```

```
    'region'      => 'us-east-1',
]);


$result = $s3Client->createBucket([
    'Bucket' => 'doc-example-bucket--use1-az4--x-s3',
    'CreateBucketConfiguration' => [
        'Location' => ['Name'=> 'use1-az4', 'Type'=> 'AvailabilityZone'],
        'Bucket' => ["DataRedundancy" => "SingleAvailabilityZone" ,"Type" =>
 "Directory"]   ],
]);
```

SDK for Python

此範例顯示如何使用建立目錄值區 AWS SDK for Python (Boto3)。

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def create_bucket(s3_client, bucket_name, availability_zone):
    '''
    Create a directory bucket in a specified Availability Zone

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to create; for example, 'doc-example-bucket--usw2-
az1--x-s3'
    :param availability_zone: String; Availability Zone ID to create the bucket in,
 for example, 'usw2-az1'
    :return: True if bucket is created, else False
    '''

    try:
        bucket_config = {
                'Location': {
                    'Type': 'AvailabilityZone',
                    'Name': availability_zone
                },
                'Bucket': {
                    'Type': 'Directory',
```

```
                    'DataRedundancy': 'SingleAvailabilityZone'
                }
            }
        s3_client.create_bucket(
            Bucket = bucket_name,
            CreateBucketConfiguration = bucket_config
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True


if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    availability_zone = 'usw2-az1'
    s3_client = boto3.client('s3', region_name = region)
    create_bucket(s3_client, bucket_name, availability_zone)
```

SDK for Ruby

此範例顯示如何使用建立目錄值區 AWS SDK for Ruby。

Example

```
s3 = Aws::S3::Client.new(region:'us-west-2')
s3.create_bucket(
  bucket: "bucket_base_name--az_id--x-s3",
  create_bucket_configuration: {
    location: { name: 'usw2-az1', type: 'AvailabilityZone' },
    bucket: { data_redundancy: 'SingleAvailabilityZone', type: 'Directory' }
  }
)
```

使用 AWS CLI

此範例顯示如何使用建立目錄值區 AWS CLI。若要使用指令，請以您自己的資訊取代使用*#######*。

建立目錄值區時，您必須提供組態詳細資訊，並使用下列命名慣例：*bucket-base-name*--*azid*--x-s3

```
aws s3api create-bucket
--bucket bucket-base-name--azid--x-s3
--create-bucket-configuration 'Location={Type=AvailabilityZone,Name=usw2-
az1},Bucket={DataRedundancy=SingleAvailabilityZone,Type=Directory}'
--region us-west-2
```

如需詳細資訊，請參閱 AWS Command Line Interface

檢視目錄儲存貯體屬性

您可以使用 Amazon S3 主控台檢視和設定 Amazon S3 目錄儲存貯體的屬性。如需詳細資訊，請參閱目錄值區 及 什麼是 S3 Express One Zone？。

使用 S3 主控台

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇目錄儲存貯體索引標籤。
4. 在目錄儲存貯體清單中，選擇要檢視其屬性的儲存貯體名稱。
5. 選擇屬性索引標籤。
6. 在 [屬性] 索引標籤上，您可以檢視值區的下列屬性：

   - 目錄儲存貯體概觀 — 您可以查看值區的可用區域、Amazon 資源名稱 (ARN) 和建立日期。AWS 區域
   - 預設加密：Amazon S3 會套用使用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密作為所有 S3 儲存貯體的基本加密層級。目錄儲存貯體的此設定無法修改。Amazon S3 會在將物件儲存到磁碟之前加密，並在下載物件時解密。如需詳細資訊，請參閱 對 Amazon S3 儲存貯體設定預設伺服器端加密行為。

     如需目錄儲存貯體所支援功能的詳細資訊，請參閱 S3 Express One Zone 的功能。

管理目錄儲存貯體的儲存貯體政策

您可以使用 Amazon S3 主控台和 AWS 開發套件新增、刪除、更新和檢視 Amazon S3 目錄儲存貯體的儲存貯體政策。如需詳細資訊，請參閱下列主題。如需 S3 Express 單一區域支援 AWS Identity and Access Management (IAM) 動作和條件金鑰的詳細資訊，請參閱AWS Identity and Access

Management 適用於 S3 快遞單一區域的 (IAM)。如需目錄儲存貯體的儲存貯體政策範例，請參閱 S3 Express One Zone 的目錄儲存貯體政策範例。

主題

新增儲存貯體政策

若要將儲存貯體政策新增至目錄儲存貯體，您可以使用 Amazon S3 主控台、 AWS 開發套件 AWS CLI 或.

使用 S3 主控台

建立或編輯儲存貯體政策

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇目錄儲存貯體索引標籤。
4. 在目錄儲存貯體清單中，選擇要將資料夾或檔案上傳至其中的儲存貯體名稱。
5. 選擇許可索引標籤標籤。
6. 在 Bucket policy (儲存貯體政策) 下方，選擇 Edit (編輯)。Edit bucket policy (編輯儲存貯體政策) 頁面隨即出現。
7. 若要自動產生策略，請選擇「策略產生器」。

   如果您選擇「策略產生器」，則「 AWS 策略產生器」會在新視窗中開啟。

   如果您不想使用 AWS 原則產生器，您可以在 [原則] 區段中新增或編輯 JSON 陳述式。

   a. 在 AWS Policy Generator (AWS 政策產生器) 頁面上，針對 Select Type of Policy (選取政策類型)，選擇 S3 Bucket Policy (S3 儲存貯體政策)。
   b. 在提供的欄位中輸入資訊，以新增陳述式，然後選擇 Add Statement (新增陳述式)。針對您要新增的敘述句數量重複此步驟。如需這些欄位的詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

> **Note**
>
> 為了方便起見，[編輯儲存貯體政策] 頁面會在 [政策] 文字欄位上方顯示目前儲存貯體的儲存貯體 ARN (Amazon 資源名稱)。您可以複製此 ARN，以在 AWS Policy Generator (政策產生器) 頁面上的陳述式中使用。

c. 完成新增陳述式後，選擇 Generate Policy (產生政策)。

d. 複製產生的政策文字，選擇 Close (關閉)，然後退回 Amazon S3 主控台中的 Edit bucket policy (編輯儲存貯體政策) 頁面。

8. 在 [原則] 方塊中，編輯現有政策，或從 [原則產生器] 貼上值區 AWS 原則。請務必先處理安全性警告、錯誤、一般警告，以及建議，然後再儲存政策。

> **Note**
>
> 儲存貯體政策的大小限制為 20 KB。

9. 選擇 Save changes (儲存變更)，這會讓您回到 Permissions (許可) 索引標籤。

使用 AWS 軟體開發套件

SDK for Java 2.x

Example

PutBucketPolicy AWS SDK for Java 2.x

```
public static void setBucketPolicy(S3Client s3Client, String bucketName, String
 policyText) {

      //sample policy text
      /**
       * policy_statement = {
        *         'Version': '2012-10-17',
        *         'Statement': [
        *             {
        *                 'Sid': 'AdminPolicy',
        *                 'Effect': 'Allow',
        *                 'Principal': {
        *                     "AWS": "111122223333"
```

```
         *                 },
         *                 'Action': 's3express:*',
         *                 'Resource':
 'arn:aws:s3express:region:111122223333:bucket/bucket-base-name--azid--x-s3'
         *             }
         *         ]
         *     }
         */
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();
            s3Client.putBucketPolicy(policyReq);
            System.out.println("Done!");
        }

        catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

使用 AWS CLI

此範例顯示如何使用將值區政策新增至目錄值區 AWS CLI。若要使用指令，請將使用*######*取代為您自己的資訊。

```
aws s3api put-bucket-policy --bucket bucket-base-name--azid--x-s3 --policy file://
bucket_policy.json
```

垃圾桶政策. JSON：

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AdminPolicy",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3express*",
            "Resource": "arn:aws:s3express:us-west-2:111122223333:bucket/"
        }
    ]
}
```

若要取得更多資訊，請參閱put-bucket-policy中的 AWS Command Line Interface。

檢視儲存貯體政策

若要檢視目錄值區的值區政策，請使用下列範例。

使用 AWS CLI

此範例顯示如何使用檢視附加至目錄值區的值區政策 AWS CLI。若要使用指令，請將使用*#######*取代為您自己的資訊。

```
aws s3api get-bucket-policy --bucket bucket-base-name--azid--x-s3
```

若要取得更多資訊，請參閱get-bucket-policy中的 AWS Command Line Interface。

刪除儲存貯體政策

若要刪除目錄值區的值區政策，請使用下列範例。

使用 AWS 軟體開發套件

SDK for Java 2.x

Example

`DeleteBucketPolicy` AWS SDK for Java 2.x

```
public static void deleteBucketPolicy(S3Client s3Client, String bucketName) {
```

```
        try {
            DeleteBucketPolicyRequest deleteBucketPolicyRequest =
DeleteBucketPolicyRequest
                    .builder()
                    .bucket(bucketName)
                    .build()
            s3Client.deleteBucketPolicy(deleteBucketPolicyRequest);
            System.out.println("Successfully deleted bucket policy");
        }

        catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

使用 AWS CLI

此範例顯示如何使用刪除目錄值區的值區政策 AWS CLI。若要使用指令，請將使用*#######*取代為您自己的資訊。

```
aws s3api delete-bucket-policy --bucket bucket-base-name--azid--x-s3
```

若要取得更多資訊，請參閱delete-bucket-policy中的 AWS Command Line Interface。

清空目錄儲存貯體

您可以使用 Amazon S3 主控台清空 Amazon S3 目錄儲存貯體。如需有關目錄儲存貯體的詳細資訊，請參閱 目錄值區。

清空目錄儲存貯體之前，請注意下列事項：

- 當您清空目錄儲存貯體時，將會刪除內含的所有物件，但會保留目錄儲存貯體本身。
- 清空目錄值區之後，空白動作就無法復原。
- 在空值區動作進行中時新增至目錄值區的物件可能會遭到刪除。

如果您也想刪除值區，請注意下列事項：

- 您必須先刪除目錄儲存貯體中的所有物件，才能刪除儲存貯體本身。
- 您必須先中止目錄儲存貯體中正在進行的分段上傳，才能刪除儲存貯體本身。

> **Note**
>
> 透過 AWS Command Line Interface (CLI) 執行的`s3 rm`命令、透過 Mountpoint 執行的`delete`作業，以及透過的 [清空值區] 選項按鈕無法刪除 AWS Management Console 目錄值區中進行中的多部分上傳。若要刪除這些進行中的分段上傳，請使用此`ListMultipartUploads`作業在值區中列出進行中的多部分上傳，並使用此`AbortMultupartUpload`作業中止所有進行中的分段上傳。

若要刪除目錄儲存貯體，請參閱 刪除目錄儲存貯體。若要中止進行中的分段上傳，請參閱。the section called “中止分段上傳”

若要清空一般用途儲存貯體，請參閱 清空儲存貯體。

使用 S3 主控台

若要清空目錄值區

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇目錄儲存貯體索引標籤。
4. 選擇您要清空的值區名稱旁邊的選項按鈕，然後選擇 [清空]。
5. 在 Empty bucket (清空儲存貯體) 頁面上，在文字欄位中輸入 **`permanently delete`** 以確認您要清空儲存貯體，然後選擇 Empty (清空)。
6. 在「空值區：狀態」頁面上監控值區清空程序的進度。

刪除目錄儲存貯體

您只能刪除空的 Amazon S3 目錄儲存貯體。刪除目錄值區之前，您必須刪除值區中的所有物件，並中止所有進行中的多部分上傳。

若要清空目錄儲存貯體，請參閱 清空目錄儲存貯體。若要中止進行中的分段上傳，請參閱。the section called “中止分段上傳”

若要刪除一般用途儲存貯體，請參閱 刪除儲存貯體。

使用 S3 主控台

清空目錄值區並中止所有進行中的分段上傳後，您可以刪除值區。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇目錄儲存貯體索引標籤。
4. 在 [目錄值區] 清單中，選擇要刪除之值區旁邊的選項按鈕。
5. 選擇刪除。
6. 在「刪除值區」頁面的文字欄位中輸入值區的名稱，以確認刪除值區。

   > **Important**
   >
   > 目錄儲存貯體刪除後，即無法復原。

7. 若要刪除目錄儲存貯體，請選擇刪除儲存貯體。

使用 AWS 軟體開發套件

下列範例使用 AWS SDK for Java 2.x 和刪除目錄值區 AWS SDK for Python (Boto3)。

SDK for Java 2.x

Example

```
public static void deleteBucket(S3Client s3Client, String bucketName) {

    try {
        DeleteBucketRequest del = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();
        s3Client.deleteBucket(del);
        System.out.println("Bucket " + bucketName + " has been deleted");
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

SDK for Python

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def delete_bucket(s3_client, bucket_name):
    '''
    Delete a directory bucket in a specified Region

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to delete; for example, 'doc-example-bucket--usw2-
az1--x-s3'
    :return: True if bucket is deleted, else False
    '''

    try:
        s3_client.delete_bucket(Bucket = bucket_name)
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    s3_client = boto3.client('s3', region_name = region)
```

使用 AWS CLI

此範例顯示如何使用刪除目錄值區 AWS CLI。若要使用指令，請以您自己的資訊取代使用*#######*。

```
aws s3api delete-bucket --bucket bucket-base-name--azid--x-s3 --region us-west-2
```

如需詳細資訊，請參閱. AWS Command Line Interface

列出目錄儲存貯體

下列範例顯示如何使用 AWS SDK 和 AWS CLI 列出目錄值區。

使用 AWS 軟體開發套件

SDK for Java 2.x

Example

下列範例會使用列出目錄值區 AWS SDK for Java 2.x。

```
public static void listBuckets(S3Client s3Client) {
       try {
           ListDirectoryBucketsRequest listDirectoryBucketsRequest =
ListDirectoryBucketsRequest.builder().build();
           ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);
           if (response.hasBuckets()) {
               for (Bucket bucket: response.buckets()) {
                   System.out.println(bucket.name());
                   System.out.println(bucket.creationDate());
               }
           }
       }

       catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

SDK for Python

Example

下列範例會使用列出目錄值區 AWS SDK for Python (Boto3)。

```
import logging
import boto3
from botocore.exceptions import ClientError

def list_directory_buckets(s3_client):
'''
Prints a list of all directory buckets in a Region
```

```
:param s3_client: boto3 S3 client
:return: True if there are buckets in the Region, else False
'''
try:
    response = s3_client.list_directory_buckets()
    for bucket in response['Buckets']:
        print (bucket['Name'])
except ClientError as e:
    logging.error(e)
    return False
return True


if __name__ == '__main__':
    region = 'us-east-1'
    s3_client = boto3.client('s3', region_name = region)
    list_directory_buckets(s3_client)
```

AWS SDK for .NET

Example

下列範例會使用列出目錄值區 AWS SDK for .NET。

```
var listDirectoryBuckets = await amazonS3Client.ListDirectoryBucketsAsync(new
 ListDirectoryBucketsRequest
{
  MaxDirectoryBuckets = 10
  }).ConfigureAwait(false);
```

SDK for PHP

Example

下列範例會使用列出目錄值區 AWS SDK for PHP。

```
require 'vendor/autoload.php';

$s3Client = new S3Client([
```

```
    'region'      => 'us-east-1',
]);
$result = $s3Client->listDirectoryBuckets();
```

SDK for Ruby

Example

下列範例會使用列出目錄值區 AWS SDK for Ruby。

```
s3 = Aws::S3::Client.new(region:'us-west-1')
s3.list_directory_buckets
```

使用 AWS CLI

下列`list-directory-buckets`範例命令顯示如何使用列 AWS CLI 出 *us-east-* 1 區域中的目錄值區。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3api list-directory-buckets --region us-east-1
```

如需詳細資訊，請參閱 AWS CLI 命令參考中的 list-directory-buckets。

HeadBucket搭配目錄值區使用

下列 AWS SDK 範例說明如何使用 HeadBucket API 操作來判斷 Amazon S3 目錄儲存貯體是否存在，以及您是否有存取該儲存貯體的權限。

使用 AWS 軟體開發套件

下列 AWS SDK for Java 2.x 範例說明如何判斷值區是否存在，以及您是否有存取該值區的權限。

SDK for Java 2.x

Example

AWS SDK for Java 2.x

```
public static void headBucket(S3Client s3Client, String bucketName) {
```

```
    try {
         HeadBucketRequest headBucketRequest = HeadBucketRequest
                 .builder()
                 .bucket(bucketName)
                 .build();
         s3Client.headBucket(headBucketRequest);
         System.out.format("Amazon S3 bucket: \"%s\" found.", bucketName);
    }

    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

使用 AWS CLI

下列head-bucket範例命令顯示如何使用 AWS CLI 來判斷目錄值區是否存在，以及您是否有存取該目錄值區的權限。若要執行此命令，請以您自己的資訊取代使用者輸入預留位置。

```
aws s3api head-bucket --bucket bucket-base-name--azid--x-s3
```

如需詳細資訊，請參閱 AWS CLI 命令參考中的 head-bucket。

使用目錄值區中的物件

建立 Amazon S3 目錄儲存貯體之後，您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI) 和 AWS 開發套件來處理物件。

如需使用 S3 Express 單區儲存類別中存放的物件進行大量物件操作的相關資訊，請參閱物件管理。如需匯入、上載、複製、刪除和下載物件，以及從目錄值區中的物件讀取中繼資料的詳細資訊，請參閱下列主題。

主題

- 將物件匯入目錄儲存貯體
- 使用 Batch Operations 搭配 S3 Express One Zone
- 將物件上傳至目錄儲存貯體
- 搭配目錄儲存貯體使用多部分上傳

- 將物件複製到目錄儲存貯體
- 刪除目錄儲存貯體中的物件
- 下載目錄值區中的物件
- HeadObject搭配目錄值區使用

將物件匯入目錄儲存貯體

在 Amazon S3 中建立目錄儲存貯體之後，您可以使用匯入動作將資料填入新的儲存貯體。匯入是用來建立 S3 Batch Operations 任務的簡化方法，可將物件從一般用途儲存貯體複製到目錄儲存貯體。

> **Note**
>
> 下列限制適用於匯入任務：
>
> - 來源儲存貯體和目的地儲存貯體必須於相同 AWS 區域 和帳戶中。
> - 來源儲存貯體不可以是目錄儲存貯體。
> - 不支援大於 5GB 的物件，而且會從複製操作中省略。
> - Glacier Flexible Retrieval、Glacier Deep Archive、Intelligent-Tiering Archive Access 層和 Intelligent-Tiering Deep Archive 層儲存類別中的物件必須先還原才能匯入。
> - 使用 MD5 檢查總和演算法匯入的物件會轉換為使用 CRC32 檢查總和。
> - 匯入的物件會使用搭配 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密。
> - 匯入的物件會使用 Express One Zone 儲存類別，此類別的定價結構與一般用途值區所使用的儲存類別不同。匯入大量物件時，請考慮這項成本上的差異。

當您設定匯入任務時，可以指定要從中複製現有物件的來源儲存貯體或字首。另外也會提供具有存取來源物件之許可的 AWS Identity and Access Management (IAM) 角色。接著 Amazon S3 就會啟動 Batch Operations 任務來複製物件，並自動套用適當的儲存類別和檢查總和設定。

您可以使用 Amazon S3 主控台來設定匯入任務。

使用 Amazon S3 主控台

將物件匯入目錄儲存貯體

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

2. 在左側導覽窗格中，選擇儲存貯體，然後選擇目錄儲存貯體索引標籤。選擇要將物件匯入其中的目錄儲存貯體旁的選項按鈕。
3. 選擇匯入。
4. 針對來源，輸入包含要匯入之物件的一般用途儲存貯體 (或包含字首的儲存貯體路徑)。若要從清單中選擇現有的一般用途儲存貯體，請選擇瀏覽 S3。
5. 針對存取和複製來源物件的許可，執行下列其中一項操作，以指定具有匯入來源物件所需許可的 IAM 角色：

   - 若要允許 Amazon S3 代表您建立新的 IAM 角色，請選擇建立新的 IAM 角色。

     > **Note**
     >
     > 如果來源物件是採用使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 的伺服器端加密進行加密，請不要選擇建立新的 IAM 角色選項。請改為指定具有 `kms:Decrypt` 許可的現有 IAM 角色。
     > Amazon S3 將使用此許可來解密您的物件。在匯入過程中，Amazon S3 將使用伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 重新加密這些物件。

   - 若要從清單中選擇現有的 IAM 角色，請選擇從現有的 IAM 角色中選擇。
   - 若要透過輸入 Amazon Resource Name (ARN) 來指定現有的 IAM 角色，請選擇輸入 IAM 角色 ARN，然後在對應的欄位中輸入 ARN。
6. 檢閱目的地和複製物件設定區段中顯示的資訊。如果目的地區段中的資訊正確無誤，請選擇匯入以開始複製任務。

   Amazon S3 主控台會在 Batch Operations 頁面上顯示新任務的狀態。如需有關任務的詳細資訊，請選擇任務名稱旁的選項按鈕，然後在動作選單上選擇檢視詳細資訊。若要開啟要將物件匯入其中的目錄儲存貯體，請選擇檢視匯入目的地。

使用 Batch Operations 搭配 S3 Express One Zone

您可以使用 Amazon S3 Batch Operations 對存放在 S3 儲存貯體中的物件執行操作。若要深入了解 S3 Batch Operations，請參閱在 Amazon S3 物件上執行大規模批次操作。

下列主題討論對目錄儲存貯體中 S3 Express 單區儲存類別中存放的物件執行批次作業。

主題

- 使用 Batch Operations 搭配目錄儲存貯體

- 主要差異

使用 Batch Operations 搭配目錄儲存貯體

您可以對儲存在目錄值區中的物件執行「複製」作業和叫用AWS Lambda函數作業。使用 Copy，您可以在相同類型的值區之間複製物件 (例如，從目錄值區到目錄值區)。您也可以在一般用途儲存貯體和目錄儲存貯體之間進行複製。若使用調用 AWS Lambda 函數，則可利用 Lambda 函數透過您定義的程式碼對目錄儲存貯體中的物件執行動作。

複製物件

您可以在相同的儲存貯體類型之間，或目錄儲存貯體與一般用途儲存貯體之間進行複製。複製到目錄儲存貯體時，必須針對此儲存貯體類型使用正確的 Amazon 資源名稱 (ARN) 格式。目錄儲存貯體的 ARN 格式為 `arn:aws:s3express:`*`region`*`:`*`account-id`*`:bucket/`*`bucket-base-name`*`--x-s3`。

您也可以使用 S3 主控台中的匯入動作，將資料填入目錄儲存貯體。匯入是用來建立 Batch Operations 任務的簡化方法，可將物件從一般用途儲存貯體複製到目錄儲存貯體。針對從一般用途儲存貯體到目錄儲存貯體的匯入複製任務，S3 會自動產生清單檔案。如需詳細資訊，請參閱將物件匯入目錄儲存貯體和指定資訊清單。

調用 Lambda 函數（ ）**`LambdaInvoke`**

使用 Batch Operations 調用對目錄儲存貯體執行動作的 Lambda 函數時，須遵循特殊需求。例如，您必須使用 v2 JSON 叫用結構描述來建構 Lambda 請求，並在建立工作InvocationSchemaVersion 2.0時指定。如需詳細資訊，請參閱叫用AWS Lambda函數。

主要差異

以下是當您使用 Batch 操作對使用 S3 Express One Zone Zone 儲存類別存放在目錄儲存貯體中的物件執行批次操作時，主要差異清單：

- Amazon S3 會自動加密上傳到 S3 儲存貯體的所有新物件。S3 儲存貯體的預設加密組態一律為啟用狀態，且最低限度設定為使用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密。對於目錄儲存貯體，僅支援 SSSE-S3。如果您使用客戶提供的金鑰 (SSE-C) 設定伺服器端加密的`CopyObject`要求，或在目錄儲存貯體 AWS Key Management Service (來源或目的地AWS KMS) 上使用 () 金鑰 (SSE-KMS) 進行伺服器端加密，則回應會傳回 HTTP 錯誤。`400 (Bad Request)`
- 目錄儲存貯體中的物件無法加上標籤。您只能指定空的標籤集。根據預設，Batch Operations 會複製標籤。如果您複製在一般用途值區與目錄值區之間有標籤的物件，就會收到`501 (Not Implemented)`回應。

- S3 Express One Zone 提供選項，讓您選擇用於在上傳或下載期間驗證資料的總和檢查碼演算法。您可以選擇以下 Secure Hash 演算法 (SHA) 或循環冗餘檢查 (CRC) 資料完整性演算法之一：CRC32、CRC32C、SHA-1 和 SHA-256。S3 快速單區域儲存類別不支援以 MD5 為基礎的總和檢查碼。
- 根據預設，所有 Amazon S3 儲存貯體都會將 S3 物件擁有權設定設定設定為強制執行儲存貯體擁有者，並停用存取控制清單 (ACL)。目錄儲存貯體的此設定無法修改。您也可以將物件從一般用途儲存貯體複製到目錄儲存貯體。不過，當您複製到目錄值區或從目錄儲存貯體複製時，您無法覆寫預設 ACL。
- 無論您如何指定清單檔案，此清單本身都必須儲存在一般用途儲存貯體中。Batch 作業無法從目錄儲存貯體匯入現有的資訊清單 (或將產生的資訊清單儲存至)。不過，清單檔案內描述的物件可以儲存在目錄儲存貯體中。
- Batch 操作無法將目錄儲存貯體指定為 S3 庫存報告中的位置。庫存報告不支援目錄儲存貯體。您可以使用 `ListObjectsV2` API 作業列出物件，為目錄值區內的物件建立資訊清單檔案。然後，您可以將清單插入 CSV 檔案中。

授與 存取權

若要執行複製任務，您必須具有下列許可：

- 若要在目錄儲存貯體之間複製物件，您必須具有 `s3express:CreateSession` 許可。
- 若要將物件從目錄儲存貯體複製到一般用途儲存貯體，您必須具有 `s3express:CreateSession` 許可和 `s3:PutObject` 許可，以將複製的物件寫入目的地儲存貯體。
- 若要將物件從一般用途值區複製到目錄值區，您必須擁有讀取要複製之來源物件的`s3:GetObject`權限和權限。`s3express:CreateSession`

  如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 CopyObject。
- 若要調用 Lambda 函數，您必須根據 Lambda 函數為資源授予許可。要確定需要哪些權限，請檢查相應的 API 操作。

將物件上傳至目錄儲存貯體

建立 Amazon S3 目錄儲存貯體之後，您可以將物件上傳到該儲存貯體。下列範例顯示如何使用 S3 主控台和 AWS SDK 將物件上傳至目錄儲存貯體。如需使用 S3 Express 單一區域進行大量物件上傳作業的相關資訊，請參閱物件管理。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇目錄儲存貯體索引標籤。
4. 選擇您要上傳資料夾或檔案的儲存貯體名稱。
5. 在「物件」清單中，選擇「上載」。
6. 在「上載」頁面上，執行下列任一項作業：
   - 將檔案和資料夾拖放至虛線上傳區域。
   - 選擇 [新增檔案] 或 [新增資料夾]，選擇要上傳的檔案或資料夾，然後選擇 [開啟] 或 [上傳]。
7. 在 [總和檢查碼] 底下，選擇您要使用的 [總和檢查碼] 功能。

   (選擇性) 如果您要上傳大小小於 16 MB 的單一物件，您也可以指定預先計算的總和檢查碼值。當您提供預先計算的值時，Amazon S3 會將其與使用選取的總和檢查碼函數所計算的值進行比較。如果這些值不相符，就不會開始上傳。
8. [權限] 和 [屬性] 區段中的選項會自動設定為預設設定，且無法修改。區塊公用存取會自動啟用，且無法為目錄儲存貯體啟用 S3 版本控制和 S3 物件鎖定。

   (選擇性) 如果您要將索引鍵值配對中的中繼資料新增至物件，請展開「屬性」區段，然後在「中繼資料」區段中選擇「新增中繼資料」。
9. 若要上傳列出的檔案和資料夾，請選擇 [上傳]。

   Amazon S3 會上傳您的物件和資料夾。上傳完成後，您可以在上傳：狀態頁面上看到成功訊息。

使用 AWS 軟體開發套件

SDK for Java 2.x

Example

```
public static void putObject(S3Client s3Client, String bucketName, String objectKey,
 Path filePath) {
       //Using File Path to avoid loading the whole file into memory
       try {
           PutObjectRequest putObj = PutObjectRequest.builder()
```

```
                    .bucket(bucketName)
                    .key(objectKey)
                    //.metadata(metadata)
                    .build();
            s3Client.putObject(putObj, filePath);
            System.out.println("Successfully placed " + objectKey +" into bucket
 "+bucketName);

        }

        catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
}
```

SDK for Python

Example

```
import boto3
import botocore
from botocore.exceptions import ClientError


def put_object(s3_client, bucket_name, key_name, object_bytes):
    """
    Upload data to a directory bucket.
    :param s3_client: The boto3 S3 client
    :param bucket_name: The bucket that will contain the object
    :param key_name: The key of the object to be uploaded
    :param object_bytes: The data to upload
    """
    try:
        response = s3_client.put_object(Bucket=bucket_name, Key=key_name,
                            Body=object_bytes)
        print(f"Upload object '{key_name}' to bucket '{bucket_name}'.")
        return response
    except ClientError:
        print(f"Couldn't upload object '{key_name}' to bucket '{bucket_name}'.")
        raise

def main():
```

```
    # Share the client session with functions and objects to benefit from S3 Express
 One Zone auth key
    s3_client = boto3.client('s3')
    # Directory bucket name must end with --azid--x-s3
    resp = put_object(s3_client, 'doc-bucket-example--use1-az5--x-s3', 'sample.txt',
 b'Hello, World!')
    print(resp)

if __name__ == "__main__":
    main()
```

使用 AWS CLI

下列put-object範例命令顯示如 AWS CLI 何使用從 Amazon S3 上傳物件。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3api put-object --bucket bucket-base-name--azid--x-s3 --key sampleinut/file001.bin
 --body bucket-seed/file001.bin
```

如需詳細資訊，請參閱 AWS CLI 命令參考中的 put-object。

搭配目錄儲存貯體使用多部分上傳

您可以使用分段上載流程將單一物件上載為一組零件。每個組件都是物件資料的接續部分。您可依任何順序分別上傳這些物件組件。若任何組件的傳輸失敗，您可再次傳輸該組件，而不會影響其他組件。當物件的所有組件都全部上傳完後，Amazon S3 會將這些組件組合起來建立該物件。一般而言，當物件大小達到 100 MB 時，應考慮使用分段上傳，而不是以單次操作上傳物件。

使用分段上傳具備下列優勢：

- 改善輸送量 - 您可平行上傳各組件以改進輸送量。
- 從任何網路問題中快速復原 — 較小的零件尺寸可將因網路錯誤而重新啟動失敗上傳的影響降到最低。
- 暫停及繼續上傳物件 - 您可在一段時間內上傳物件組件。啟動分段上傳之後，就沒有到期日。您必須明確完成或中止分段上傳。
- 在您知道最終物件大小前開始上傳 - 您可在建立物件的同時上傳物件。

我們建議您以下列方式使用分段上傳：

- 如果您要透過穩定的高頻寬網路上傳大型物件，請使用多部分上傳，以 parallel 上傳物件部分以獲得多執行緒效能，以充分利用可用頻寬。
- 如果您透過不穩定的網路上傳，請使用多部分上傳來避免上傳重新啟動，以提高網路錯誤的復原能力。使用分段上傳時，您只需要重新上載上傳期間中斷的部分。您不需要從頭開始重新上傳物件。

當您使用分段上傳將物件上傳到目錄儲存貯體中的 Amazon S3 Express One Zone Zone 儲存類別時，多部分上傳程序類似於使用多部分上傳將物件上傳到一般用途儲存貯體的程序。不過，還是有一些顯著的差異。

如需使用分段上傳將物件上傳到 S3 Express One 區域的相關資訊，請參閱下列主題。

主題

多部分上傳過程

多部分上傳是一個三個步驟的過程：

- 您啟動上傳。
- 您上傳物件的零件。
- 上傳完所有零件之後，即可完成多部分上傳。

收到完整的分段上傳請求後，Amazon S3 會從上傳的零件建構物件，然後您就可以像存取儲存貯體中任何其他物件一樣存取物件。

啟動分段上傳

當您傳送要求要啟動分段上傳時，Amazon S3 會傳回具有上傳 ID 的回應，其為分段上傳的唯一識別符。每次上傳分段各組件、列出各組件、完成上傳或中止上傳時，都必須納入此上傳 ID。

組件上傳

上傳某個分段組件時，除了上傳 ID 之外，還必須指定組件編號。當您透過 S3 Express One Zone 使用多部分上傳時，多部分零件編號必須是連續的零件編號。如果您嘗試完成具有非連續零件編號的多部分上傳請求，則會產生 `HTTP 400 Bad Request` (無效的零件訂單) 錯誤。

零件編號可唯一識別零件及其在您要上載之物件中的位置。如果您使用與先前上載零件相同的零件編號來上載新零件，則先前上載的零件會遭到覆寫。

每次上傳一個組件時，Amazon S3 會在其回應中傳回 企業標籤 (ETag) 標頭。您必須記錄每個上傳組件的組件編號與 ETag 值。所有物件零件上載的 ETag 值將保持不變，但每個零件都會被指派不同的零件編號。後續的要求中必須包含這些值，才能完成分段上傳。

Amazon S3 會自動加密上傳到 S3 儲存貯體的所有新物件。進行分段上傳時，若您未在請求中指定加密資訊，所上傳分段的加密設定會設為目的地儲存貯體的預設加密組態。Amazon S3 儲存貯體的預設加密組態一律為啟用狀態，且最低限度設定為使用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密。對於目錄值區，僅支援 SSE-S3。如需詳細資訊，請參閱 使用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密。

完成分段上傳

當您完成分段上傳時，Amazon S3 會根據零件編號以遞增順序串連零件來建立物件。成功完成請求之後，這些片段就不再存在。

您完整的分段上傳請求必須包含上傳 ID 以及零件編號及其對應 ETag 值的清單。Amazon S3 回應包含的 ETag 可識別獨特的物件資料組合。這個 ETag 不是物件資料的 MD5 雜湊。

分段上傳清單

您可列出特定分段上傳的組件或所有進行中之分段上傳。列出組件操作會傳回特定分段上傳之已上傳組件的資訊。Amazon S3 會為每項列出的組件要求，傳回指定分段上傳組件的資訊，上限為 1,000 個組件。若分段上傳中有超過 1,000 個分段，您必須使用分頁來擷取所有分段。

傳回的零件清單不包含尚未完成上傳的零件。使用列出分段上傳操作，即可取得正在進行中的分段上傳清單。

進行中的分段上傳是您已啟動但尚未完成或已中止的上傳。每個要求最多可傳回 1,000 個分段上傳。若正在進行超過 1,000 個的分段上傳，您必須另行傳送請求以擷取剩餘的分段上傳。傳回的清單僅用於進行驗證。傳送完成分段上傳請求時，請不要使用此清單的結果。而是在上傳 Amazon S3 傳回的組件與相對應之 ETag 值時，保有您自己的組件編號清單。

如需有關分段上傳清單的詳細資訊，請參閱 Amazon 簡單儲存服務 API 參考ListParts中的。

使用分段上傳操作的檢查總和

當您上傳物件時，可以指定檢查總和演算法來檢查物件完整性。目錄儲存貯體不支援 MD5。您可以指定下列其中一個安全雜湊演算法 (SHA) 或循環冗餘檢查 (CRC) 資料完整性檢查演算法：

- CRC32
- CRC32C
- SHA-1
- SHA-256

您可以使用 Amazon S3 REST API 或 AWS 開發套件，透過使`GetObject`用或擷取個別零件的總和檢查值。`HeadObject`如果想要擷取分段上傳 (仍在進行中) 之個別部分的總和檢查值，您可以使用`ListParts`。

> **Important**
>
> 使用上述總和檢查碼演算法時，多組件零件編號必須使用連續的零件編號。如果您嘗試使用不連續的零件編號完成多部分上傳請求，Amazon S3 會產生 `HTTP 400 Bad Request` (零件訂單無效) 錯誤。

如需如何使用分段物件檢查總和的詳細資訊，請參閱「檢查物件完整性」。

並行分段上傳操作

在分散式開發環境中，您的應用程式可以同時在同一物件上啟動多個更新。例如，您的應用程式可能會使用相同的物件索引鍵來啟動數個分段上傳。然後針對這些每一個上傳，應用程式會上傳各組件，並對 Amazon S3 傳送完成上傳要求，以建立物件。對於 S3 Express One Zone 來說，物件建立時間是指分段上傳的完成日期。

> **Important**
>
> 儲存在目錄值區中的物件不支援版本控制。

分段上傳和定價

啟動分段上傳之後，Amazon S3 就會保留所有分段，直到您完成或中止上傳為止。在其整個生命週期內，您都要支付此分段上傳及其相關組件的儲存體、頻寬與要求之費用。如果您中止分段上傳，Amazon S3 會刪除上傳成品和您上傳的任何零件，而且不會再支付這些成品的費用。無論指定的儲存空間類別為何，刪除不完整的分段上傳都不會收取提前刪除費用。如需定價的詳細資訊，請參閱 Amazon S3 定價。

> **Important**
>
> 如果完整的多部分上載請求未成功發送，則對象部件將被「組裝」，並且不會創建對象。系統會向您收取與已上傳部分相關的所有儲存空間費用。請務必完成多部分上載以建立物件，或中止多部分上載以移除所有上載的零件。
>
> 您必須完成或中止所有進行中的多部分上傳，才能刪除目錄值區。目錄儲存貯體不支援 S3 生命週期組態。如果需要，您可以列出活動中的分段上傳內容，然後中止上傳，然後刪除值區。

多部分上傳 API 操作和權限

若要允許存取目錄值區上的物件管理 API 作業，請在值區政策或 AWS Identity and Access Management (IAM) 身分型政策中授與`s3express:CreateSession`權限。

您必須要有必要許可，才可使用分段上傳操作。您可以使用儲存貯體政策或 IAM 身分型政策授與 IAM 主體許可以執行這些操作。下表列出各種分段上傳操作所需的許可。

您可以透過元素識別多部分上傳的初始器。`Initiator`如果初始器是 AWS 帳戶，則此元素會提供與元`Owner`素相同的資訊。若啟動者是 IAM 使用者，此元素會提供使用者 ARN 與顯示名稱。

動作	所需的許可
建立分段上傳	若要建立分段上傳，您必須被允許對目錄值區執行`s3express:CreateSession` 動作。
啟動多部分上傳	若要啟動分段上傳，您必須被允許對目錄值區執行`s3express:CreateSession` 動作。
上傳零件	若要上載零件，您必須被允許對目錄值區執行`s3express:CreateSession` 動作。

動作	所需的許可
	若要讓初始器上傳零件，值區擁有者必須允許初始器對目錄值區執行s3express:CreateSession 動作。
上傳零件 (副本)	若要上載零件，您必須被允許對目錄值區執行s3express:CreateSession 動作。 啟動者若要分段上傳該物件，儲存貯體擁有者必須允許啟動者對物件執行s3express:CreateSession 動作。
完成分段上傳	若要完成分段上傳，您必須被允許對目錄值區執行s3express:CreateSession 動作。 若要讓啟動者完成分段上傳，值區擁有者必須允許初始器對物件執行s3express:CreateSession 動作。
中止分段上傳	若要中止多部分上載，您必須被允許執行動作s3express:CreateSession 。 若要讓啟動器中止多部分上載，必須授與啟動器明確允許存取權，才能執行動作。s3express:CreateSession
清單零件	若要列出分段上傳中的零件，您必須被允許對目錄值區執行s3express:CreateSession 動作。
列出進行中的分段上傳	若要列出儲存貯體進行中的分段上傳作業，您必須被允許對該值區執行s3:ListBucketMultipartUploads 動作。

多部分上傳的 API 操作支持

Amazon 簡單儲存服務 API 參考中的以下各節說明用於多部分上傳的 Amazon S3 REST API 操作。

- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- AbortMultipartUpload
- ListParts

- ListMultipartUploads

範例

若要使用分段上傳將物件上傳至目錄儲存貯體中的 S3 Express One Zone 區域，請參閱下列範例。

主題

- 建立分段上傳
- 上傳分段上傳的部分
- 完成分段上傳
- 中止分段上傳
- 建立分段上傳複製操作
- 列出進行中的多部分上傳
- 列出分段上傳的部分

建立分段上傳

下列範例說明如何建立分段上傳。

使用 AWS 軟體開發套件

SDK for Java 2.x

Example

```
/**
 * This method creates a multipart upload request that generates a unique upload ID
 that is used to track
 * all the upload parts
 *
 * @param s3
 * @param bucketName - for example, 'doc-example-bucket--use1-az4--x-s3'
 * @param key
 * @return
 */
 private static String createMultipartUpload(S3Client s3, String bucketName, String
 key) {
```

```
    CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

    String uploadId = null;

    try {
        CreateMultipartUploadResponse response =
 s3.createMultipartUpload(createMultipartUploadRequest);
        uploadId = response.uploadId();
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return uploadId;
```

SDK for Python

Example

```
def create_multipart_upload(s3_client, bucket_name, key_name):
    '''
   Create a multipart upload to a directory bucket

   :param s3_client: boto3 S3 client
   :param bucket_name: The destination bucket for the multipart upload
   :param key_name: The key name for the object to be uploaded
   :return: The UploadId for the multipart upload if created successfully, else None
   '''

   try:
        mpu = s3_client.create_multipart_upload(Bucket = bucket_name, Key =
 key_name)
        return mpu['UploadId']
    except ClientError as e:
        logging.error(e)
        return None
```

使用 AWS CLI

此範例顯示如何使用建立多部分上傳至目錄儲存貯體。 AWS CLI*###### KEY_NAME ########## bucket-base-name--azid--x-s3#*若要使用指令，請將使用*#######*取代為您自己的資訊。

```
aws s3api create-multipart-upload --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
```

若要取得更多資訊，請參閱create-multipart-upload中的 AWS Command Line Interface。

上傳分段上傳的部分

下列範例顯示如何上傳分段上傳的部分內容。

使用 AWS 軟體開發套件

SDK for Java 2.x

下列範例會示範如何將單一物件分割成多個部分，然後使用 Java 2.x 的 SDK 將這些部分上傳至目錄儲存貯體。

Example

```
/**
 * This method creates part requests and uploads individual parts to S3 and then
 returns all the completed parts
 *
 * @param s3
 * @param bucketName
 * @param key
 * @param uploadId
 * @throws IOException
 */
 private static ListCompletedPartmultipartUpload(S3Client s3, String bucketName,
 String key, String uploadId, String filePath) throws IOException {

        int partNumber = 1;
        ListCompletedPart completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        // read the local file, breakdown into chunks and process
        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            int position = 0;
```

```
            while (position < fileSize) {
                file.seek(position);
                int read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        }

        catch (IOException e) {
            throw e;
        }
        return completedParts;
    }
```

SDK for Python

下列範例會示範如何將單一物件分割成多個部分，然後使用適用於 Python 的 SDK 將這些部分上傳至目錄儲存貯體。

Example

```
def multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_size):
```

```
    '''
    Break up a file into multiple parts and upload those parts to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name for object to be uploaded and for the local file
that's being uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_size: The size parts that the object will be broken into, in bytes.
                      Minimum 5 MiB, Maximum 5 GiB. There is no minimum size for the
last part of your multipart upload.
    :return: part_list for the multipart upload if all parts are uploaded
successfully, else None
    '''

    part_list = []
    try:
        with open(key_name, 'rb') as file:
            part_counter = 1
            while True:
                file_part = file.read(part_size)
                if not len(file_part):
                    break
                upload_part = s3_client.upload_part(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = mpu_id,
                    Body = file_part,
                    PartNumber = part_counter
                )
                part_list.append({'PartNumber': part_counter, 'ETag':
upload_part['ETag']})
                part_counter += 1
    except ClientError as e:
        logging.error(e)
        return None
    return part_list
```

使用 AWS CLI

此範例顯示如何將單一物件分割成多個部分，然後使用將這些零件上傳至目錄儲存貯體 AWS CLI。若要使用指令，請將使用#######取代為您自己的資訊。

```
aws s3api upload-part --bucket bucket-base-name--azid--x-s3 --
key KEY_NAME --part-number 1 --body LOCAL_FILE_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

如需詳細資訊，請參閱中的上傳部分。 AWS Command Line Interface

完成分段上傳

下列範例顯示如何完成分段上傳。

使用 AWS 軟體開發套件

SDK for Java 2.x

下列範例說明如何使用適用於 Java 2.x 的 SDK 來完成多部分上傳。

Example

```
/**
 * This method completes the multipart upload request by collating all the upload
 parts
 * @param s3
 * @param bucketName - for example, 'doc-example-bucket--usw2-az1--x-s3'
 * @param key
 * @param uploadId
 * @param uploadParts
 */
 private static void completeMultipartUpload(S3Client s3, String bucketName, String
 key, String uploadId, ListCompletedPart uploadParts) {
        CompletedMultipartUpload completedMultipartUpload =
 CompletedMultipartUpload.builder()
                .parts(uploadParts)
                .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
                CompleteMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .multipartUpload(completedMultipartUpload)
                        .build();
```

```
        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

    public static void multipartUploadTest(S3Client s3, String bucketName, String
 key, String localFilePath)  {
        System.out.println("Starting multipart upload for: " + key);
        try {
            String uploadId = createMultipartUpload(s3, bucketName, key);
            System.out.println(uploadId);
            ListCompletedPart parts = multipartUpload(s3, bucketName, key, uploadId,
 localFilePath);
            completeMultipartUpload(s3, bucketName, key, uploadId, parts);
            System.out.println("Multipart upload completed for: " + key);
        }

        catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python

下列範例說明如何使用適用於 Python 的 SDK 來完成多部分上傳作業。

Example

```
def complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_list):
    '''
    Completes a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: The destination bucket for the multipart upload
    :param key_name: The key name for the object to be uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_list: The list of uploaded part numbers with their associated ETags
    :return: True if the multipart upload was completed successfully, else False
    '''

    try:
        s3_client.complete_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
```

```
            UploadId = mpu_id,
            MultipartUpload = {
                'Parts': part_list
            }
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    MB = 1024 ** 2
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'OBJECT_NAME'
    part_size = 10 * MB
    s3_client = boto3.client('s3', region_name = region)
    mpu_id = create_multipart_upload(s3_client, bucket_name, key_name)
    if mpu_id is not None:
        part_list = multipart_upload(s3_client, bucket_name, key_name, mpu_id,
 part_size)
        if part_list is not None:
            if complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id,
 part_list):
                print (f'{key_name} successfully uploaded through a ultipart upload
 to {bucket_name}')
            else:
                print (f'Could not upload {key_name} hrough a multipart upload to
 {bucket_name}')
```

使用 AWS CLI

此範例顯示如何使用完成目錄值區的分段上傳。 AWS CLI若要使用指令，請將使用*#######*取代為您自己的資訊。

```
aws s3api complete-multipart-upload --bucket bucket-base-name--azid--x-s3 --
key KEY_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
 --multipart-upload file://parts.json
```

此範例採用 JSON 結構，描述應重新組合到完整檔案中的多部分上傳部分。在此範例中，`file://`前置詞用於從名為的本機資料夾中的檔案載入 JSON 結構`parts`。

部分：

```
parts.json
{
  "Parts": [
    {
      "ETag": "6b78c4a64dd641a58dac8d9258b88147",
      "PartNumber": 1
    }
  ]
}
```

若要取得更多資訊，請參閱complete-multipart-upload中的 AWS Command Line Interface。

中止分段上傳

下列範例顯示如何中止分段上傳。

使用 AWS 軟體開發套件

SDK for Java 2.x

下列範例會示範如何使用適用於 Java 2.x 的 SDK 中止多部分上傳。

Example

```
public static void abortMultiPartUploads( S3Client s3, String bucketName ) {

         try {
             ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                     .bucket(bucketName)
                     .build();

             ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
             ListMultipartUpload uploads = response.uploads();

             AbortMultipartUploadRequest abortMultipartUploadRequest;
             for (MultipartUpload upload: uploads) {
                 abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                         .bucket(bucketName)
                         .key(upload.key())
```

```
                        .uploadId(upload.uploadId())
                        .build();

                s3.abortMultipartUpload(abortMultipartUploadRequest);
            }

        }

        catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python

下列範例會示範如何使用適用於 Python 的 SDK 中止多部分上傳。

Example

```
import logging
import boto3
from botocore.exceptions import ClientError


def abort_multipart_upload(s3_client, bucket_name, key_name, upload_id):
    '''
    Aborts a partial multipart upload in a directory bucket.

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket where the multipart upload was initiated - for
 example, 'doc-example-bucket--usw2-az1--x-s3'
    :param key_name: Name of the object for which the multipart upload needs to be
 aborted
    :param upload_id: Multipart upload ID for the multipart upload to be aborted
    :return: True if the multipart upload was successfully aborted, False if not
    '''
    try:
        s3_client.abort_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = upload_id
        )
```

```
    except ClientError as e:
        logging.error(e)
        return False
    return True


if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'KEY_NAME'
        upload_id = 'UPLOAD_ID'
    s3_client = boto3.client('s3', region_name = region)
    if abort_multipart_upload(s3_client, bucket_name, key_name, upload_id):
        print (f'Multipart upload for object {key_name} in {bucket_name} bucket has
 been aborted')
    else:
        print (f'Unable to abort multipart upload for object {key_name} in
 {bucket_name} bucket')
```

使用 AWS CLI

下列範例示範如何使用中止分段上傳。 AWS CLI若要使用指令，請將使用*#######*取代為您自己的資訊。

```
aws s3api abort-multipart-upload --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
 --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
MAQAAAAB0OxUFeA7LTbWWFS8WYwhrxDxTIDN-pdEEq_agIHqsbg"
```

若要取得更多資訊，請參閱abort-multipart-upload中的 AWS Command Line Interface。

建立分段上傳複製操作

下列範例說明如何使用多部分上傳將 bject 從一個值區複製到另一個值區。

使用 AWS 軟體開發套件

SDK for Java 2.x

下列範例會示範如何使用多部分上傳，透過使用 Java 2.x 的 SDK，以程式設計方式將物件從一個值區複製到另一個值區。

Example

```
/**
 * This method creates a multipart upload request that generates a unique upload ID
 that is used to track
 * all the upload parts.
 *
 * @param s3
 * @param bucketName
 * @param key
 * @return
 */
 private static String createMultipartUpload(S3Client s3, String bucketName, String
 key) {
        CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();
        String uploadId = null;
        try {
            CreateMultipartUploadResponse response =
 s3.createMultipartUpload(createMultipartUploadRequest);
            uploadId = response.uploadId();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return uploadId;
  }

  /**
   * Creates copy parts based on source object size and copies over individual parts
   *
   * @param s3
   * @param sourceBucket
   * @param sourceKey
   * @param destnBucket
   * @param destnKey
   * @param uploadId
   * @return
   * @throws IOException
   */
```

```
   public static ListCompletedPart multipartUploadCopy(S3Client s3, String
sourceBucket, String sourceKey, String destnBucket, String destnKey, String
uploadId) throws IOException {

        // Get the object size to track the end of the copy operation.
        HeadObjectRequest headObjectRequest = HeadObjectRequest
                .builder()
                .bucket(sourceBucket)
                .key(sourceKey)
                .build();
        HeadObjectResponse response = s3.headObject(headObjectRequest);
        Long objectSize = response.contentLength();

        System.out.println("Source Object size: " + objectSize);

        // Copy the object using 20 MB parts.
        long partSize = 20 * 1024 * 1024;
        long bytePosition = 0;
        int partNum = 1;
        ListCompletedPart completedParts = new ArrayList<>();
        while (bytePosition < objectSize) {
            // The last part might be smaller than partSize, so check to make sure
            // that lastByte isn't beyond the end of the object.
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

            System.out.println("part no: " + partNum + ", bytePosition: " +
bytePosition + ", lastByte: " + lastByte);

            // Copy this part.
            UploadPartCopyRequest req = UploadPartCopyRequest.builder()
                    .uploadId(uploadId)
                    .sourceBucket(sourceBucket)
                    .sourceKey(sourceKey)
                    .destinationBucket(destnBucket)
                    .destinationKey(destnKey)
                    .copySourceRange("bytes="+bytePosition+"-"+lastByte)
                    .partNumber(partNum)
                    .build();
            UploadPartCopyResponse res = s3.uploadPartCopy(req);
            CompletedPart part = CompletedPart.builder()
                    .partNumber(partNum)
                    .eTag(res.copyPartResult().eTag())
                    .build();
            completedParts.add(part);
```

```
            partNum++;
            bytePosition += partSize;
        }
        return completedParts;
    }


    public static void multipartCopyUploadTest(S3Client s3, String srcBucket, String
 srcKey, String destnBucket, String destnKey)  {
        System.out.println("Starting multipart copy for: " + srcKey);
        try {
            String uploadId = createMultipartUpload(s3, destnBucket, destnKey);
            System.out.println(uploadId);
            ListCompletedPart parts = multipartUploadCopy(s3, srcBucket,
 srcKey,destnBucket,  destnKey, uploadId);
            completeMultipartUpload(s3, destnBucket, destnKey, uploadId, parts);
            System.out.println("Multipart copy completed for: " + srcKey);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python

下列範例會示範如何使用多部分上傳，透過使用 Python 的 SDK，以程式設計方式將物件從一個值區複製到另一個值區。

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def head_object(s3_client, bucket_name, key_name):
    '''
    Returns metadata for an object in a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that contains the object to query for metadata
    :param key_name: Key name to query for metadata
    :return: Metadata for the specified object if successful, else None
    '''
```

```
    try:
        response = s3_client.head_object(
            Bucket = bucket_name,
            Key = key_name
        )
        return response
    except ClientError as e:
        logging.error(e)
        return None

def create_multipart_upload(s3_client, bucket_name, key_name):
    '''
    Create a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name of the object to be uploaded
    :return: UploadId for the multipart upload if created successfully, else None
    '''

    try:
        mpu = s3_client.create_multipart_upload(Bucket = bucket_name, Key =
 key_name)
        return mpu['UploadId']
    except ClientError as e:
        logging.error(e)
        return None

def multipart_copy_upload(s3_client, source_bucket_name, key_name,
 target_bucket_name, mpu_id, part_size):
    '''
    Copy an object in a directory bucket to another bucket in multiple parts of a
 specified size

    :param s3_client: boto3 S3 client
    :param source_bucket_name: Bucket where the source object exists
    :param key_name: Key name of the object to be copied
    :param target_bucket_name: Destination bucket for copied object
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_size: The size parts that the object will be broken into, in bytes.
                      Minimum 5 MiB, Maximum 5 GiB. There is no minimum size for the
 last part of your multipart upload.
    :return: part_list for the multipart copy if all parts are copied successfully,
 else None
```

```
    '''

    part_list = []
    copy_source = {
        'Bucket': source_bucket_name,
        'Key': key_name
    }
    try:
        part_counter = 1
        object_size = head_object(s3_client, source_bucket_name, key_name)
        if object_size is not None:
            object_size = object_size['ContentLength']
        while (part_counter - 1) * part_size <object_size:
            bytes_start = (part_counter - 1) * part_size
            bytes_end = (part_counter * part_size) - 1
            upload_copy_part = s3_client.upload_part_copy (
                Bucket = target_bucket_name,
                CopySource = copy_source,
                CopySourceRange = f'bytes={bytes_start}-{bytes_end}',
                Key = key_name,
                PartNumber = part_counter,
                UploadId = mpu_id
            )
            part_list.append({'PartNumber': part_counter, 'ETag':
 upload_copy_part['CopyPartResult']['ETag']})
            part_counter += 1
    except ClientError as e:
        logging.error(e)
        return None
    return part_list

def complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_list):
    '''
    Completes a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name of the object to be uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_list: List of uploaded part numbers with associated ETags
    :return: True if the multipart upload was completed successfully, else False
    '''

    try:
```

```
        s3_client.complete_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = mpu_id,
            MultipartUpload = {
                'Parts': part_list
            }
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    MB = 1024 ** 2
    region = 'us-west-2'
    source_bucket_name = 'SOURCE_BUCKET_NAME'
    target_bucket_name = 'TARGET_BUCKET_NAME'
    key_name = 'KEY_NAME'
    part_size = 10 * MB
    s3_client = boto3.client('s3', region_name = region)
    mpu_id = create_multipart_upload(s3_client, target_bucket_name, key_name)
    if mpu_id is not None:
        part_list = multipart_copy_upload(s3_client, source_bucket_name, key_name,
 target_bucket_name, mpu_id, part_size)
        if part_list is not None:
            if complete_multipart_upload(s3_client, target_bucket_name, key_name,
 mpu_id, part_list):
                print (f'{key_name} successfully copied through multipart copy from
 {source_bucket_name} to {target_bucket_name}')
            else:
                print (f'Could not copy {key_name} through multipart copy from
 {source_bucket_name} to {target_bucket_name}')
```

使用 AWS CLI

下列範例示範如何使用多部分上傳，以程式設計方式使用 AWS CLI. 若要使用指令，請將使用#######取代為您自己的資訊。

```
aws s3api upload-part-copy --bucket bucket-base-name--azid--x-s3 --key TARGET_KEY_NAME
 --copy-source SOURCE_BUCKET_NAME/SOURCE_KEY_NAME --part-number 1 --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

若要取得更多資訊，請參閱upload-part-copy中的 AWS Command Line Interface。

列出進行中的多部分上傳

若要列出目錄儲存貯體的進行中分段上傳，您可以使用 AWS SDK 或. AWS CLI

使用 AWS 軟體開發套件

SDK for Java 2.x

下列範例說明如何使用適用於 Java 2.x 的 SDK 列出進行中 (不完整) 分段上傳。

Example

```
public static void listMultiPartUploads( S3Client s3, String bucketName) {
       try {
           ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

           ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
           List MultipartUpload uploads = response.uploads();
           for (MultipartUpload upload: uploads) {
               System.out.println("Upload in progress: Key = \"" + upload.key() +
"\", id = " + upload.uploadId());
           }
     }
     catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
     }
 }
```

SDK for Python

下列範例說明如何使用適用於 Python 的 SDK 列出進行中 (不完整) 分段上傳。

Example

```
import logging
import boto3
from botocore.exceptions import ClientError
```

```
def list_multipart_uploads(s3_client, bucket_name):
    '''
    List any incomplete multipart uploads in a directory bucket in e specified gion

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to check for incomplete multipart uploads
    :return: List of incomplete multipart uploads if there are any, None if not
    '''

    try:
        response = s3_client.list_multipart_uploads(Bucket = bucket_name)
        if 'Uploads' in response.keys():
            return response['Uploads']
        else:
            return None
    except ClientError as e:
        logging.error(e)

if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    s3_client = boto3.client('s3', region_name = region)
    multipart_uploads = list_multipart_uploads(s3_client, bucket_name)
    if multipart_uploads is not None:
        print (f'There are {len(multipart_uploads)} ncomplete multipart uploads for
 {bucket_name}')
    else:
        print (f'There are no incomplete multipart uploads for {bucket_name}')
```

使用 AWS CLI

下列範例說明如何使用列出進行中 (不完整) 分段上傳。 AWS CLI若要使用指令，請將使用#######取代為您自己的資訊。

```
aws s3api list-multipart-uploads --bucket bucket-base-name--azid--x-s3
```

若要取得更多資訊，請參閱list-multipart-uploads中的 AWS Command Line Interface。

列出分段上傳的部分

下列範例顯示如何列出多部分上傳至目錄儲存貯體的部分。

使用 AWS 軟體開發套件

SDK for Java 2.x

下列範例說明如何使用適用於 Java 2.x 的 SDK，列出多部分上傳至目錄儲存貯體的部分。

```
public static void listMultiPartUploadsParts( S3Client s3, String bucketName, String
 objKey, String uploadID) {

        try {
            ListPartsRequest listPartsRequest = ListPartsRequest.builder()
                .bucket(bucketName)
                .uploadId(uploadID)
                .key(objKey)
                .build();

            ListPartsResponse response = s3.listParts(listPartsRequest);
            ListPart parts = response.parts();
            for (Part part: parts) {
                System.out.println("Upload in progress: Part number = \"" +
 part.partNumber() + "\", etag = " + part.eTag());
            }

        }

        catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }


    }
```

SDK for Python

下列範例說明如何使用 SDK of Python，列出多部分上傳至目錄儲存貯體的部分。

```
import logging
import boto3
from botocore.exceptions import ClientError

def list_parts(s3_client, bucket_name, key_name, upload_id):
    '''
```

```
    Lists the parts that have been uploaded for a specific multipart upload to a
 directory bucket.

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that multipart uploads parts have been uploaded to
    :param key_name: Name of the object that has parts uploaded
    :param upload_id: Multipart upload ID that the parts are associated with
    :return: List of parts associated with the specified multipart upload, None if
 there are no parts
    '''
    parts_list = []
    next_part_marker = ''
    continuation_flag = True
    try:
        while continuation_flag:
            if next_part_marker == '':
                response = s3_client.list_parts(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = upload_id
                )
            else:
                response = s3_client.list_parts(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = upload_id,
                    NextPartMarker = next_part_marker
                )
            if 'Parts' in response:
                for part in response['Parts']:
                    parts_list.append(part)
                if response['IsTruncated']:
                    next_part_marker = response['NextPartNumberMarker']
                else:
                    continuation_flag = False
            else:
                continuation_flag = False
        return parts_list
    except ClientError as e:
        logging.error(e)
        return None

if __name__ == '__main__':
    region = 'us-west-2'
```

```
    bucket_name = 'BUCKET_NAME'
    key_name = 'KEY_NAME'
    upload_id = 'UPLOAD_ID'
    s3_client = boto3.client('s3', region_name = region)
    parts_list = list_parts(s3_client, bucket_name, key_name, upload_id)
    if parts_list is not None:
        print (f'{key_name} has {len(parts_list)} parts uploaded to {bucket_name}')
    else:
        print (f'There are no multipart uploads with that upload ID for
{bucket_name} bucket')
```

使用 AWS CLI

下列範例說明如何使用列出多部分上傳至目錄儲存貯體的部分。 AWS CLI若要使用指令，請將使用*####*取代為您自己的資訊。

```
aws s3api list-parts --bucket bucket-base-name--azid--x-s3 --key KEY_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

如需詳細資訊，請參閱中的清單零件。 AWS Command Line Interface

將物件複製到目錄儲存貯體

該複製操作會為已存放在 Amazon S3 中的物件建立複本。您可以在目錄儲存貯體和一般用途儲存貯體之間複製物件。您也可以在儲存貯體內以及相同類型的儲存貯體之間複製物件，例如從目錄儲存貯體複製到目錄儲存貯體。

您可以在單一原子作業中建立最多 5 GB 的物件副本。不過，若要複製大於 5 GB 的物件，您必須使用分段上傳 API 作業。如需詳細資訊，請參閱 搭配目錄儲存貯體使用多部分上傳。

許可

若要複製物件，您必須具有下列許可：

- 若要在目錄儲存貯體之間複製物件，您必須具有 `s3express:CreateSession` 許可。
- 若要將物件從目錄儲存貯體複製到一般用途儲存貯體，您必須具有 `s3express:CreateSession` 許可和 `s3:PutObject` 許可，以將複製的物件寫入目的地儲存貯體。
- 若要將物件從一般用途值區複製到目錄值區，您必須擁有讀取要複製之來源物件的`s3:GetObject`權限和權限。`s3express:CreateSession`

如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 CopyObject。

加密

Amazon S3 會自動加密上傳到 S3 儲存貯體的所有新物件。S3 儲存貯體的預設加密組態一律為啟用狀態，且最低限度設定為使用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密。

對於目錄值區，僅支援 SSE-S3。對於一般用途值區，您可以使用 SSE-S3 (預設值)、使用 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS) 的伺服器端加密、使用金鑰的雙層伺服器端加密 (DSSE-KMS)，或使用客戶提供的金 AWS KMS 鑰 (SSE-C) 進行伺服器端加密。

如果您提出複製要求，將目錄儲存貯體上的 SSE-C、SSE-KMS 或 DSSE-KMS 參數設定為來源或目的地，則回應會傳回錯誤訊息，

標籤

目錄儲存貯體不支援標籤。如果您將具有標籤的物件從一般用途值區複製到目錄儲存貯體，就會收到 HTTP `501 (Not Implemented)` 回應。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 CopyObject。

ETag

S3 快速單一區域的實體標籤 (ETag) 是隨機的英數字元字串，而不是 MD5 總和檢查碼。若要協助確保物件完整性，請使用其他總和檢查碼。

其他檢查總和

S3 Express One Zone 可讓您選擇用於在上傳或下載過程中驗證資料的檢查總和演算法選項。您可以選擇以下 Secure Hash 演算法 (SHA) 或循環冗餘檢查 (CRC) 資料完整性演算法之一：CRC32、CRC32C、SHA-1 和 SHA-256。S3 快速單區域儲存類別不支援以 MD5 為基礎的總和檢查碼。

如需詳細資訊，請參閱 S3 其他檢查總和最佳實務。

支援的功能

如需 S3 快速單區域支援哪些 Amazon S3 功能的詳細資訊，請參閱S3 Express One Zone 有什麼不同？。

使用 S3 主控台 (複製到目錄儲存貯體)

將物件從一般用途值區或目錄值區複製到目錄值區

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇您要從中複製物件的值區：
   - 若要從一般用途值區進行複製，請選擇「一般用途時段」頁標。
   - 若要從目錄值區進行複製，請選擇 [目錄值區] 索引標籤。
4. 選擇包含您要複製之物件的一般用途值區或目錄值區。
5. 選擇 Objects (物件) 索引標籤。在「物件」頁面上，選取您要複製之物件名稱左側的核取方塊。
6. 在 Actions (動作) 功能表上，選擇 Copy (複製)。

   便會顯示「複製」頁面。
7. 在「目的地」下，選擇目的地類型的「目錄值區」。若要指定目的地路徑，請選擇瀏覽 S3，導覽至目的地，然後選擇目的地左側的選項按鈕。選擇右下角的 Choose destination (選擇目的地)。

   或者，輸入目的地路徑。
8. 在「總和檢查」(Check sum) 底下，選擇是否要使用現有的總和檢查碼功能複製物件，或以新的總和檢查碼功能取代現有的總和檢查碼函數。上傳物件時，您可以選擇指定用於驗證資料完整性的檢查總和演算法。複製物件時，您可以選擇一個新函數。如果您最初沒有指定額外的校驗和，則可以使用 e 校驗和部分添加一個。

   > **Note**
   >
   > 即使您選擇使用相同的總和檢查碼函數，如果物件大小超過 16 MB，您的總和檢查碼值也可能會變更。檢查總和的值可能會因為計算分段上傳檢查總和而變更。如需複製物件時檢查總和會如何變化的詳細資訊，請參閱「對分段上傳使用部分檢查總和」。

   若要變更檢查總和函數，請選擇Replace with a new checksum function (替換為新的檢查總和函數)。從下拉列表中選擇新的校驗和功能。複製物件時，會使用指定的演算法來計算和儲存新的總和檢查碼。
9. 選擇右下角的 Copy (複製)。Amazon S3 會將您的物件複製到目的地。

使用 S3 主控台 (複製到一般用途儲存貯體)

將物件從目錄儲存貯體複製到一般用途儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇「目錄值區」頁標。
4. 選擇包含您要複製之物件的目錄值區。
5. 選擇 Objects (物件) 索引標籤。在「物件」頁面上，選取您要複製之物件名稱左側的核取方塊。
6. 在 Actions (動作) 功能表上，選擇 Copy (複製)。
7. 在「目的地」下，選擇目的地類型的「一般用途值區」。若要指定目的地路徑，請選擇瀏覽 S3，導覽至目的地，然後選擇目的地左側的選項按鈕。選擇右下角的 Choose destination (選擇目的地)。

   或者，輸入目的地路徑。
8. 在「總和檢查」(Check sum) 底下，選擇是否要使用現有的總和檢查碼功能複製物件，或以新的總和檢查碼功能取代現有的總和檢查碼函數。上傳物件時，您可以選擇指定用於驗證資料完整性的檢查總和演算法。複製物件時，您可以選擇一個新函數。如果您原本沒有指定額外的總和檢查碼，您可以使用 [總和檢查碼] 區段來新增一個總和檢查碼。

   > **Note**
   >
   > 即使您選擇使用相同的總和檢查碼函數，如果物件大小超過 16 MB，您的總和檢查碼值也可能會變更。檢查總和的值可能會因為計算分段上傳檢查總和而變更。如需複製物件時檢查總和會如何變化的詳細資訊，請參閱「對分段上傳使用部分檢查總和」。

   若要變更檢查總和函數，請選擇Replace with a new checksum function (替換為新的檢查總和函數)。從下拉列表中選擇新的校驗和功能。複製物件時，會使用指定的演算法來計算和儲存新的總和檢查碼。
9. 選擇右下角的 Copy (複製)。Amazon S3 會將您的物件複製到目的地。

使用 AWS 軟體開發套件

SDK for Java 2.x

Example

```
public static void copyBucketObject (S3Client s3, String sourceBucket, String
objectKey, String targetBucket) {
     CopyObjectRequest copyReq = CopyObjectRequest.builder()
         .sourceBucket(sourceBucket)
         .sourceKey(objectKey)
         .destinationBucket(targetBucket)
         .destinationKey(objectKey)
         .build();
      String temp = "";

      try {
          CopyObjectResponse copyRes = s3.copyObject(copyReq);
          System.out.println("Successfully copied " + objectKey +" from bucket " +
sourceBucket +" into bucket "+targetBucket);
      }

      catch (S3Exception e) {
          System.err.println(e.awsErrorDetails().errorMessage());
          System.exit(1);
      }
}
```

使用 AWS CLI

下列copy-object範例指令顯示如何使用將物件從一個值區複製到另一個值區。 AWS CLI 您可以在值區類型之間複製物件。若要執行此命令，請以您自己的資訊取代使用者輸入預留位置。

```
aws s3api copy-object --copy-source bucket SOURCE_BUCKET/SOURCE_KEY_NAME --
key TARGET_KEY_NAME --bucket TARGET_BUCKET_NAME
```

如需詳細資訊，請參閱 AWS CLI 命令參考中的 copy-object。

刪除目錄儲存貯體中的物件

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI) 或 AWS 開發套件，從 Amazon S3 目錄儲存貯體刪除物件。如需詳細資訊，請參閱 目錄值區 及 什麼是 S3 Express One Zone？。

Warning

- 刪除物件無法復原。
- 此動作會刪除所有指定的物件。刪除資料夾時，請等待刪除動作完成，然後再將新物件加入至資料夾。否則，系統也可能會刪除新物件。

Note

當您以程式設計方式從目錄值區刪除多個物件時，請注意下列事項：

- `DeleteObjects` 請求中的物件索引鍵必須至少包含一個非空格字元。不支援所有空格字元的字串。
- `DeleteObjects`要求中的物件索引鍵不能包含 Unicode 控制字元，除了換行符號 (\n)、tab (\t) 和歸位字元 (\r)。

使用 S3 主控台

刪除物件

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇目錄儲存貯體索引標籤。
4. 選擇包含您要刪除之物件的目錄值區。
5. 選擇 Objects (物件) 索引標籤。在「物件」清單中，選取要刪除之一或多個物件左側的核取方塊。
6. 選擇刪除。
7. 在 [刪除物件] 頁面上，於文字方塊**`permanently delete`**中輸入。

8. 選擇 Delete objects (刪除物件)。

使用 AWS 軟體開發套件

SDK for Java 2.x

Example

下列範例會使用刪除目錄值區中的物件 AWS SDK for Java 2.x。

```
static void deleteObject(S3Client s3Client, String bucketName, String objectKey) {



        try {

            DeleteObjectRequest del = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            s3Client.deleteObject(del);

            System.out.println("Object " + objectKey + " has been deleted");


        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

    }
```

SDK for Python

Example

下列範例會使用刪除目錄值區中的物件 AWS SDK for Python (Boto3)。

```

import logging
import boto3
from botocore.exceptions import ClientError
```

```
def delete_objects(s3_client, bucket_name, objects):
    '''
    Delete a list of objects in a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that contains objects to be deleted; for example,
 'doc-example-bucket--usw2-az1--x-s3'
    :param objects: List of dictionaries that specify the key names to delete
    :return: Response output, else False
    '''

    try:
        response = s3_client.delete_objects(
            Bucket = bucket_name,
            Delete = {
                'Objects': objects
            }
        )
        return response
    except ClientError as e:
        logging.error(e)
        return False


if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    objects = [
        {
            'Key': '0.txt'
        },
        {
            'Key': '1.txt'
        },
        {
            'Key': '2.txt'
        },
        {
            'Key': '3.txt'
        },
        {
            'Key': '4.txt'
        }
```

```
    ]

    s3_client = boto3.client('s3', region_name = region)
    results = delete_objects(s3_client, bucket_name, objects)
    if results is not None:
        if 'Deleted' in results:
            print (f'Deleted {len(results["Deleted"])} objects from {bucket_name}')
        if 'Errors' in results:
            print (f'Failed to delete {len(results["Errors"])} objects from
{bucket_name}')
```

使用 AWS CLI

下列delete-object範例命令顯示如何使用 AWS CLI 從目錄值區刪除物件。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3api delete-object --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
```

如需詳細資訊，請參閱 AWS CLI 命令參考中的 delete-object。

下載目錄值區中的物件

下列程式碼範例說明如何使用 GetObject API 操作從 Amazon S3 目錄儲存貯體中的物件讀取 (下載) 資料。

使用 AWS 軟體開發套件

SDK for Java 2.x

Example

下列程式碼範例示範如何使用 AWS SDK for Java 2.x.

```
public static void getObject(S3Client s3Client, String bucketName, String objectKey)
 {
     try {
         GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(objectKey)
            .bucket(bucketName)
```

```
            .build();

         ResponseBytes GetObjectResponse objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
         byte[] data = objectBytes.asByteArray();

         //Print object contents to console
         String s = new String(data, StandardCharsets.UTF_8);
         System.out.println(s);
    }

    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

SDK for Python

Example

下列程式碼範例示範如何使用 AWS SDK for Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from botocore.response import StreamingBody

def get_object(s3_client: boto3.client, bucket_name: str, key_name: str) ->
 StreamingBody:
    """
    Gets the object.
    :param s3_client:
    :param bucket_name: The bucket that contains the object.
    :param key_name: The key of the object to be downloaded.
    :return: The object data in bytes.
    """
    try:
        response = s3_client.get_object(Bucket=bucket_name, Key=key_name)
        body = response['Body'].read()
        print(f"Got object '{key_name}' from bucket '{bucket_name}'.")
    except ClientError:
        print(f"Couldn't get object '{key_name}' from bucket '{bucket_name}'.")
        raise
```

```
    else:
        return body

def main():
    s3_client = boto3.client('s3')
    resp = get_object(s3_client, 'doc-example-bucket--use1-az4--x-s3', 'sample.txt')
    print(resp)

if __name__ == "__main__":
     main()
```

使用 AWS CLI

以下 get-object 範例命令顯示如何使用 AWS CLI 從 Amazon S3 下載物件。此命令會*KEY_NAME*從目錄值區取得物件*bucket-base-name*--*azid*--x-s3。物件將下載到名為 *LOCAL_FILE_NAME* 的檔案中。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3api get-object --bucket bucket-base-name--azid--x-s3 --
key KEY_NAME LOCAL_FILE_NAME
```

如需詳細資訊，請參閱 AWS CLI 命令參考中的 get-object。

HeadObject搭配目錄值區使用

以下 AWS SDK 和 AWS CLI 範例說明如何使用 HeadObject API 操作從 Amazon S3 目錄儲存貯體中的物件擷取中繼資料，而無需傳回物件本身。

使用 AWS 軟體開發套件

SDK for Java 2.x

Example

```
public static void headObject(S3Client s3Client, String bucketName, String
 objectKey) {
     try {
         HeadObjectRequest headObjectRequest = HeadObjectRequest
                 .builder()
                 .bucket(bucketName)
```

```
                .key(objectKey)
                .build();
        HeadObjectResponse response = s3Client.headObject(headObjectRequest);
        System.out.format("Amazon S3 object: \"%s\" found in bucket: \"%s\" with
ETag: \"%s\"", objectKey, bucketName, response.eTag());
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
```

使用 AWS CLI

下列head-object範例命令顯示如何使用從物 AWS CLI 件擷取中繼資料。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3api head-object --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
```

如需詳細資訊，請參閱 AWS CLI 命令參考中的 head-object。

S3 Express One Zone 的安全

雲端安全是 AWS 最重視的一環。身為 AWS 的客戶，您將能從資料中心和網路架構中獲益，這些都是專為最重視安全的組織而設計的。安全性是 AWS 與您共同肩負的責任。共同的責任模型 將此描述為雲端 本身 的安全和雲端內部的安全：

- 雲端本身的安全 – AWS 負責保護在 AWS 雲端 中執行 AWS 服務 的基礎設施。AWS 也提供您可安全使用的服務。在 AWS Compliance Programs 中，第三方稽核員會定期測試並驗證我們的安全成效。

  若要了解適用於 Amazon S3 Express One Zone 的合規計劃，請參閱 AWS 服務 in Scope by Compliance Program。
- 雲端內部的安全：您的責任取決於所使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件將協助您了解如何在使用 S3 Express One Zone 時套用共同責任模式。下列主題將示範如何設定 S3 Express One Zone 以達到您的安全和合規目標。另外也將帶您了解如何使用其他 AWS 服務，協助您在使用 S3 Express One Zone 時監控並保護您的資源。

主題

資料保護和加密

如需有關 S3 Express One Zone 如何加密和保護資料的詳細資訊，請參閱下方主題。

主題

使用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密

存放在目錄儲存貯體中的所有物件都預設為使用伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 進行加密。不允許在未加密的情況下上傳至目錄儲存貯體。如需詳細資訊，請參閱 使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密 及 使用加密來保護資料。

目錄儲存貯體不支援使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 的伺服器端加密、使用 AWS Key Management Service (AWS KMS) 金鑰 (DSSE-KMS) 的雙層伺服器端加密，或使用客戶所提供加密金鑰 (SSE-C) 的伺服器端加密。

傳輸中加密

S3 Express One Zone 只能透過 HTTPS (TLS) 存取。

S3 Express One Zone 使用區域 (Region) 和區域 (Zone) API 端點。需要區域 (Region) 或區域 (Zone) 端點，取決於您使用的 Amazon S3 API 操作。您可以透過閘道虛擬私有雲端 (VPC) 端點存取區域 (Zone) 和區域 (Region) 端點。使用閘道端點不需額外付費。若要進一步了解區域 (Region) 和區域 (Zone) API 端點，請參閱 S3 Express One Zone 的網路功能。

其他檢查總和

S3 Express One Zone 可讓您選擇用於在上傳或下載過程中驗證資料的檢查總和演算法選項。您可以選擇以下 Secure Hash 演算法 (SHA) 或循環冗餘檢查 (CRC) 資料完整性演算法之一：CRC32、CRC32C、SHA-1 和 SHA-256。S3 快速單區域儲存類別不支援以 MD5 為基礎的總和檢查碼。

如需詳細資訊，請參閱 S3 其他檢查總和最佳實務。

資料刪除

您可以使用 Amazon S3 主控台、AWS SDK、AWS Command Line Interface (AWS CLI) 或 Amazon S3 REST API，直接從 S3 Express One Zone 刪除一或多個物件。由於目錄儲存貯體中的所有物件都會產生儲存成本，因此建議您刪除不再需要的物件。

刪除存放在目錄儲存貯體中的物件也會週期性地刪除任何父目錄，如果這些父目錄未包含所要刪除物件以外的任何物件。

> **Note**
>
> S3 Express One Zone 不支援多重要素驗證 (MFA) 刪除和 S3 版本控制。

AWS Identity and Access Management 適用於 S3 快遞單一區域的 (IAM)

AWS Identity and Access Management (IAM) 可協助管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員可控制哪些人員可進行身分驗證 (登入) 並獲得授權 (具有許可)，以使用 S3 Express One Zone 中的 Amazon S3 資源。您可以免費使用 IAM。

根據預設，使用者未具備執行目錄儲存貯體和 S3 Express One Zone 操作的許可。若要授予存取目錄儲存貯體的許可，您可以使用 IAM 建立使用者、群組或角色，並將許可附加至這些身分。如需有關 IAM 的詳細資訊，請參閱《IAM 使用者指南》中的 IAM 安全最佳實務。

若要提供存取權，您可以透過下列方式新增許可至您的使用者、群組或角色：

- 中的使用者和群組 AWS IAM Identity Center — 建立權限集。請遵循《AWS IAM Identity Center 使用者指南》的建立許可集合中的指示。
- IAM 中透過身分提供者管理的使用者：建立聯合身分的角色。請遵循《IAM 使用者指南》的為第三方身分提供者 (聯合) 建立角色中的指示。

- IAM 角色和使用者：建立您的使用者可擔任的角色。請依照《IAM 使用者指南》中的建立角色以將許可委派給 IAM 使用者的指示進行。

根據預設，目錄儲存貯體為私有，只有明確獲得存取權的使用者才能存取。目錄儲存貯體的存取控制界限只會在儲存貯體層級設定。相反地，一般用途儲存貯體的存取控制界限可在儲存貯體、字首或物件標籤層級設定。這個差異表示，目錄儲存貯體是唯一可以包含在儲存貯體政策或 IAM 身分政策中，以提供 S3 Express One Zone 存取權的資源。

使用 S3 Express One Zone 時，除了 IAM 授權之外，您還可以透過 `CreateSession` API 操作所處理的新工作階段型機制來驗證和授權請求。您可以使用 `CreateSession` 請求臨時憑證來提供低延遲的儲存貯體存取。這些臨時憑證的範圍會設為特定目錄儲存貯體。

若要使用`CreateSession`，我們建議您使用最新版本的 AWS SDK 或使用 AWS Command Line Interface (AWS CLI)。支援的 AWS SDK 以及代表您建立、重新整 AWS CLI 理和終止工作階段。

工作階段權杖只能搭配區域 (物件層級) 操作 (`CopyObject` 和 `HeadBucket` 除外) 使用，將與授權相關的延遲分散到工作階段中的數個請求。對於區域端點 API 操作 (儲存貯體層級操作)，您可以使用不涉及管理工作階段的 IAM 授權。如需詳細資訊，請參閱 AWS Identity and Access Management 適用於 S3 快遞單一區域的 (IAM) 及 `CreateSession` 授權。

如需有關適用於 S3 Express One Zone 的 IAM 的詳細資訊，請參閱下方主題。

主題

主體

當您建立資源型政策以授予儲存貯體的存取權時，您必須使用 `Principal` 元素來指定可對該資源提出動作或操作請求的人員或應用程式。對於目錄儲存貯體政策，您可以使用下列主體：

- 一個 AWS 帳戶
- IAM 使用者
- IAM 角色

- 聯合身分使用者

如需詳細資訊，請參閱《IAM 使用者指南》中的 Principal。

資源

目錄儲存貯體的 Amazon 資源名稱 (ARN) 包含`s3express`命名空間 AWS 區域、 AWS 帳戶 ID 和目錄值區名稱，其中包括可用區域 ID。若要存取並對目錄儲存貯體執行動作，您必須使用下列 ARN 格式：

```
arn:aws:s3express:region:account-id:bucket/base-bucket-name--azid--x-s3
```

如需有關 ARN 的詳細資訊，請參閱《IAM 使用者指南》中的 Amazon Resource Names (ARNs)。如需有關資源的詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素：Resource。

S3 Express One Zone 的動作

在 IAM 身分型政策或資源型政策中，您可以定義可允許或拒絕哪些 S3 動作。S3 Express One Zone 動作對應於特定 API 操作。S3 Express One Zone 具有唯一的 IAM 命名空間，與 Amazon S3 的標準命名空間有所區別。這個命名空間是 `s3express`。

當您允許 `s3express:CreateSession` 許可時，這可讓 `CreateSession` API 操作在存取區域端點 API (也就是物件層級) 操作時擷取工作階段權杖。這些工作階段權杖會傳回憑證，可用來授予所有其他區域端點 API 操作的存取權。因此，您不需要使用 IAM 政策授予存取區域 API 操作的許可。而是由工作階段權杖啟用存取權。

如需區域 (Zone) 和區域 (Region) 端點 API 操作的詳細資訊，請參閱 S3 Express One Zone 的網路功能。若要進一步了解 `CreateSession` API 操作，請參閱《Amazon Simple Storage Service API 參考》中的 CreateSession。

您可在 IAM 政策陳述式的 `Action` 元素中指定以下動作。使用政策來授予在 AWS中執行操作的許可。在政策中使用動作時，通常會允許或拒絕存取相同名稱的 API 操作。不過，在某些情況下，單一動作可控制對多個 API 操作的存取。儲存貯體層級動作的存取權只能在 IAM 身分型政策 (使用者或角色) 中授予，無法在儲存貯體政策中授予。

S3 Express One Zone 的動作和條件索引鍵

動作	API	描述	存取層級	條件索引鍵
s3express:CreateBucket	CreateBucket	授予許可來建立新儲存貯體。	寫入	s3express:authType s3express:LocationName s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256
s3express:CreateSession	CreateSession	授予許可來建立工作階段權杖，此權杖用於授予所有區域 (物件層級) API 操作的存取權，例如 PutObject 、GetObject 等。	寫入	s3express:authType s3express:SessionMode s3express:ResourceAccount

動作	API	描述	存取層級	條件索引鍵
				s3express:signatureversion s3express:signatureAge s3express:TlsVersion s3express:x-amz-content-sha256
s3express:DeleteBucket	DeleteBucket	授予許可來刪除 URI 中命名的儲存貯體。	寫入	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

動作	API	描述	存取層級	條件索引鍵
s3express:DeleteBucketPolicy	DeleteBucketPolicy	授予許可，以在所指定的儲存貯體上刪除政策。	許可管理	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

動作	API	描述	存取層級	條件索引鍵
s3express:GetBucketPolicy	GetBucketPolicy	授予許可來傳回所指定儲存貯體的政策。	讀取	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

動作	API	描述	存取層級	條件索引鍵
s3express:ListAllMyDirectoryBuckets	ListDirectoryBuckets	授予許可來列出已驗證的請求發送者擁有的所有目錄儲存貯體。	清單	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

動作	API	描述	存取層級	條件索引鍵
s3express:PutBucketPolicy	PutBucketPolicy	授予許可來新增或取代儲存貯體上的儲存貯體政策。	許可管理	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

S3 Express One Zone 的條件索引鍵

S3 Express One Zone 定義了下列條件索引鍵，可在 IAM 政策的 Condition 元素中使用。您可以使用這些索引鍵來縮小套用政策陳述式的條件。

條件金鑰	描述	Type
s3express:authType	依身分驗證方法篩選存取權。若要限制傳入請求使用特定身分驗證方法，您可以使用此可選條件金鑰。例如，您可以使用此條件金鑰，僅允許 HTTP Authorization 標頭用在請求身分驗證中。	字串

條件金鑰	描述	Type
	有效值：REST-HEADER 、REST-QUERY-STRING	
s3express:LocationName	依特定可用區域 ID (AZ ID) 篩選對 CreateBucket API 操作的存取權，例如 usw2-az1。 範例值：usw2-az1	字串
s3express:ResourceAccount	依資源擁有者 AWS 帳戶 ID 篩選存取。 若要限制使用者、角色或應用程式存取特定 AWS 帳戶 ID 所擁有的目錄值區，您可以使用aws:ResourceAccount 或 s3express:ResourceAccount condition 索引鍵。您可以在 AWS Identity and Access Management (IAM) 身分識別政策或虛擬私有雲端 (VPC) 端點政策中使用此條件金鑰。例如，您可以使用此條件金鑰來限制 VPC 中的用戶端存取您不擁有的值區。 範例值：111122223333	字串
s3express:SessionMode	依 CreateSession API 操作請求的許可篩選存取權。根據預設，工作階段為 ReadWrite 。您可以使用此條件索引鍵將存取限於 ReadOnly，或明確拒絕 ReadWrite 存取。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 S3 Express One Zone 的目錄儲存貯體政策範例 或 CreateSession。 有效值：ReadWrite 、ReadOnly	字串

條件金鑰	描述	Type
s3express:signatureAge	依請求簽章的存在時間篩選存取權 (以毫秒為單位)。此條件僅適用於預先簽章 URL。 在 AWS 簽名版本 4 中，簽名密鑰的有效期最多為七天。因此，簽章的有效期限也是最長七天。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的「簽署請求簡介」。您可以使用此條件來進一步限制簽章存留期。 範例值：600000	數值
s3express:signatureversion	識別您要支援已驗證要求的 AWS 簽章版本。對於已驗證的請求，S3 Express One Zone 支援第 4 版簽署程序。 有效值："AWS4-HMAC-SHA256" (識別簽名版本 4)	字串
s3express:TlsVersion	依用戶端所使用的 TLS 版本篩選存取權。 您可以使用s3:TlsVersion 條件金鑰撰寫 IAM、虛擬私有雲端端點 (VPCE) 或儲存貯體政策，以根據用戶端所使用的 TLS 版本限制使用者或應用程式對目錄儲存貯體的存取。您也可以使用此條件索引鍵來撰寫需要最低 TLS 版本的政策。 範例值：1.3	數值

條件金鑰	描述	Type
s3express:x-amz-content-sha256	依儲存貯體中未簽署的內容篩選存取權。 您可以使用此條件金鑰以不允許在儲存貯體中未簽署的內容。 當您使用第 4 版簽署程序時，針對使用 Authorization 標頭的請求，會在簽署計算中新增 x-amz-content-sha256 標頭，然後將其值設定為雜湊承載。 您可以在儲存貯體政策中使用此條件索引鍵，拒絕任何尚未簽署承載的上傳項目。例如： • 拒絕使用了 Authorization 標頭來驗證請求但並未簽署承載的上傳項目。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的在單個區塊中傳輸承載。 • 拒絕使用預先簽署 URL 的上傳。預先簽章 URL 一律有 UNSIGNED_PAYLOAD 。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的身分驗證請求和身分驗證方法。 有效值：UNSIGNED-PAYLOAD	字串

如何授權和驗證 API 操作

下表列出 S3 Express One Zone API 操作的授權和身分驗證資訊。對於每項 API 操作，資料表會顯示 API 操作名稱、IAM 動作、端點類型 (區域 (Region) 或區域 (Zone)) 以及授權機制 (IAM 或工作階段型)。此資料表也會指出支援跨帳戶存取權的位置。儲存貯體層級動作的存取權只能在 IAM 身分型政策 (使用者或角色) 中授予，無法在儲存貯體政策中授予。

API	端點類型	IAM 動作	跨帳戶存取權
CreateBucket	區域性	s3express:CreateBucket	否

API	端點類型	IAM 動作	跨帳戶存取權
DeleteBucket	區域性	s3express:DeleteBucket	否
ListDirectoryBuckets	區域性	s3express:ListAllMyDirectoryBuckets	否
PutBucketPolicy	區域性	s3express:PutBucketPolicy	否
GetBucketPolicy	區域性	s3express:GetBucketPolicy	否
DeleteBucketPolicy	區域性	s3express:DeleteBucketPolicy	否
CreateSession	區域	s3express:CreateSession	是
CopyObject	區域	s3express:CreateSession	是
DeleteObject	區域	s3express:CreateSession	是
DeleteObjects	區域	s3express:CreateSession	是
HeadObject	區域	s3express:CreateSession	是
PutObject	區域	s3express:CreateSession	是
GetObjectAttributes	區域	s3express:CreateSession	是
ListObjectsV2	區域	s3express:CreateSession	是
HeadBucket	區域	s3express:CreateSession	是
CreateMultipartUpload	區域	s3express:CreateSession	是
UploadPart	區域	s3express:CreateSession	是

API	端點類型	IAM 動作	跨帳戶存取權
UploadPartCopy	區域	s3express:CreateSession	是
CompleteMultipartUpload	區域	s3express:CreateSession	是
AbortMultipartUpload	區域	s3express:CreateSession	是
ListParts	區域	s3express:CreateSession	是
ListMultipartUploads	區域	s3express:CreateSession	是

S3 Express One Zone 的 IAM 身分型政策

在您建立目錄儲存貯體或使用 Amazon S3 Express One Zone 儲存類別之前，必須先將必要的許可授予 AWS Identity and Access Management (IAM) 角色或使用者。此範例政策允許存取 CreateSession API 操作 (搭配區域端點 [物件層級] API 操作使用) 和所有區域端點 (儲存貯體層級) API 操作。此政策允許 CreateSession API 操作搭配所有目錄儲存貯體使用，但僅允許區域端點 API 操作搭配指定的目錄儲存貯體使用。若要使用此範例政策，請以您自己的資訊取代 *user input placeholders*。

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "AllowAccessRegionalEndpointAPIs",
             "Effect": "Allow",
             "Action": [
                 "s3express:DeleteBucket",
                 "s3express:DeleteBucketPolicy",
                 "s3express:CreateBucket",
                 "s3express:PutBucketPolicy",
                 "s3express:GetBucketPolicy",
```

```
                "s3express:ListAllMyDirectoryBuckets"
            ],

            "Resource": "arn:aws:s3express:region:account_id:bucket/bucket-base-
name--azid--x-s3/*"
        },
        {
            "Sid": "AllowCreateSession",
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "*"
        }
    ]
}
```

S3 Express One Zone 的目錄儲存貯體政策範例

本節提供搭配 Amazon S3 Express One Zone 儲存類別使用的目錄儲存貯體政策範例。若要使用這些政策，請將 *user input placeholders* 取代為您自己的資訊。

下列範例儲存貯體政策允許 AWS 帳戶 ID *111122223333* 使用 CreateSession API 操作搭配所指定目錄儲存貯體的預設 ReadWrite 工作階段。此政策會授予區域端點 (物件層級) API 操作的存取權。

Example - 此儲存貯體政策允許使用預設 **ReadWrite** 工作階段的 **CreateSession** 呼叫

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccess",
            "Effect": "Allow",
            "Resource": "arn:aws:s3express:us-west-2:account-id:bucket/bucket-base-
name--azid--x-s3",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            },
            "Action": [
                "s3express:CreateSession"
            ]
```

```
        }
    ]
}
```

Example - 此儲存貯體政策允許使用 **ReadOnly** 工作階段的 **CreateSession** 呼叫

下列範例儲存貯體政策允許 AWS 帳戶 ID *111122223333* 使用 CreateSession API 操作。此政策會使用 s3express:SessionMode 條件索引鍵與 ReadOnly 值來設定唯讀工作階段。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadOnlyAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3express:CreateSession",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3express:SessionMode": "ReadOnly"
                }
            }
        }
    ]
}
```

Example - 此儲存貯體政策允許 **CreateSession** 呼叫的跨帳戶存取權

下列範例儲存貯體政策允許 AWS 帳戶 ID *111122223333* 針對 AWS 帳戶 ID *444455556666* 擁有的指定目錄儲存貯體使用 CreateSession API 操作。

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

```
            "Sid": "CrossAccount",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": [
                "s3express:CreateSession"
            ],
            "Resource": "arn:aws:s3express:us-west-2:444455556666:bucket/bucket-base-
name--azid--x-s3"

        }
    ]
}
```

CreateSession 授權

Amazon S3 Express One Zone 同時支援 AWS Identity and Access Management (AWS IAM) 授權和工作階段型授權：

- 若要將區域端點 API 操作 (儲存貯體層級或控制平面、操作) 與 S3 Express One 區域搭配使用，您可以使用 IAM 授權模型，該模型不涉及工作階段管理。動作的許可會個別授予。如需詳細資訊，請參閱 AWS Identity and Access Management 適用於 S3 快遞單一區域的 (IAM)。
- 若要使用區域端點 API 操作 (物件層級，也就是資料平面操作)，您可以使用 CreateSession API 操作來建立和管理工作階段，這些工作階段經過最佳化，可提供低延遲的資料請求授權。若要擷取和使用工作階段權杖，您必須在身分型政策或儲存貯體政策中允許目錄儲存貯體的 s3express:CreateSession 動作。如需詳細資訊，請參閱 AWS Identity and Access Management 適用於 S3 快遞單一區域的 (IAM)。如果您在 Amazon S3 主控台中、透過 AWS Command Line Interface (AWS CLI) 或使用 AWS SDK 存取 S3 Express One Zone，S3 Express One Zone 會代表您建立工作階段。

如果您使用 Amazon S3 REST API，則可以使用 CreateSession API 操作來取得包含存取金鑰 ID、私密存取金鑰、工作階段權杖和到期時間的臨時安全憑證。臨時憑證提供的許可與長期安全憑證相同，例如 IAM 使用者憑證，但臨時安全憑證必須包含工作階段權杖。

工作階段模式

工作階段模式會定義工作階段的範圍。在儲存貯體政策中，您可以指定 s3express:SessionMode 條件索引鍵來控制哪些人能夠建立 ReadWrite 或 ReadOnly 工作階段。如需有關 ReadWrite 或

ReadOnly 工作階段的詳細資訊，請參閱《Amazon S3 API 參考》中 CreateSession 的 x-amz-create-session-mode 參數。如需有關要建立的儲存貯體政策的詳細資訊，請參閱 S3 Express One Zone 的目錄儲存貯體政策範例。

工作階段權杖

當您使用臨時安全憑證進行呼叫時，呼叫必須包含工作階段權杖。工作階段權杖會隨臨時憑證一併傳回。工作階段權杖的範圍設定為目錄儲存貯體，並用來驗證安全憑證有效且未過期。為保護您的工作階段，臨時安全憑證會在 5 分鐘後過期。

CopyObject 和 HeadBucket

臨時安全憑證的範圍設定為特定目錄儲存貯體，並且會自動針對所指目錄儲存貯體的所有區域 (物件層級) 操作業 API 呼叫啟用。與其他區域端點 API 操作不同的是，CopyObject 和 HeadBucket 不使用 CreateSession 身分驗證。所有 CopyObject 和 HeadBucket 請求都必須使用 IAM 憑證進行驗證和簽署。但是，CopyObject 和 HeadBucket 仍像其他區域端點 API 操作一樣，由 s3express:CreateSession 進行授權。

如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 CreateSession。

S3 Express One Zone 的安全最佳實務

在您開發和實作自己的安全政策時，可考慮使用 Amazon S3 Express One Zone 提供的多種安全功能。以下最佳實務為一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

預設封鎖公開存取和物件擁有權設定

若要使用 S3 Express One Zone 儲存類別，您必須使用 S3 目錄儲存貯體。目錄儲存貯體支援 S3 封鎖公開存取和 S3 物件擁有權。這些 S3 功能可用來稽核和管理對儲存貯體和物件的存取權。

根據預設，目錄儲存貯體的所有封鎖公開存取設定都會啟用。此外，物件擁有權會設定為儲存貯體擁有者強制執行，這表示存取控制清單 (ACL) 已停用。這些設定無法修改。如需這些功能的詳細資訊，請參閱 the section called “封鎖公開存取” 和 the section called “控制物件所有權”。

> **Note**
>
> 您無法授予存放在目錄儲存貯體中的物件存取權。您只能授予目錄儲存貯體的存取權。S3 Express One Zone 的授權模式與 Amazon S3 的授權模式不同。如需詳細資訊，請參閱 CreateSession 授權。

身分驗證和授權

S3 Express One Zone 的身分驗證和授權機制會根據您對區域 (Zone) 端點 API 操作還是區域 (Region) 端點 API 操作發出請求而有所不同。區域 (Zone) API 操作是物件層級（資料平面）操作。區域 (Region) API 操作是儲存貯體層級 (控制平面) 操作。

使用 S3 Express One Zone，您可以透過新的工作階段型機制來驗證和授權對區域 (Zone) 端點 API 操作的請求，此機制經過最佳化，可提供最低延遲。使用工作階段型身分驗證時，AWS SDK 會使用 `CreateSession` API 操作請求臨時憑證，以提供對目錄儲存貯體的低延遲存取。這些臨時憑證的範圍會設為特定目錄儲存貯體，並於 5 分鐘後過期。您可以使用這些臨時憑證來簽署區域 (物件層級) API 呼叫。如需詳細資訊，請參閱 `CreateSession` 授權。

使用 S3 Express One Zone 憑證簽署請求

您可以使用 S3 Express One Zone 憑證，透過 AWS 第 4 版簽署程序簽署區域端點 (物件層級) API 請求，並以 `s3express` 作為服務名稱。當您簽署請求時，請使用從 `CreateSession` 傳回的秘密金鑰，同時提供具有 `x-amzn-s3session-token header` 的工作階段權杖。如需詳細資訊，請參閱 CreateSession。

S3 Express One Zone 類別支援的 AWS SDK 可管理憑證並代表您進行簽署。我們建議您使用適用於 S3 Express One Zone 的 AWS SDK 來重新整理憑證並為您簽署請求。

使用 IAM 憑證簽署請求

所有區域 (儲存貯體層級) API 呼叫都必須透過 AWS Identity and Access Management (IAM) 憑證進行驗證和簽署，而非臨時工作階段憑證。IAM 憑證包含 IAM 身分的存取金鑰 ID 和私密存取金鑰。所有 `CopyObject` 和 `HeadBucket` 請求也都必須使用 IAM 憑證進行驗證和簽署。

為了讓區域 (物件層級) 操作呼叫達到最低延遲，建議您使用透過呼叫 `CreateSession` 取得的 S3 Express One Zone 憑證來簽署請求，但不包括 `CopyObject` 和 `HeadBucket` 請求。

使用 AWS CloudTrail

AWS CloudTrail 針對使用者、角色或 Amazon S3 中的 AWS 服務 所採取的動作提供記錄。您可以使用收集的信息 CloudTrail 來確定以下內容：

- 對 Amazon S3 提出的請求
- 提出請求的 IP 地址
- 提出要求的人員

- 提出請求的時間
- 有關請求的其他詳細資訊

當您設定時AWS 帳戶，預設 CloudTrail 為啟用。下列區域端點 API 作業 (儲存貯體層級或控制平面、API 作業) 會記錄到。 CloudTrail

- `CreateBucket`
- `DeleteBucket`
- `DeleteBucketPolicy`
- `PutBucketPolicy`
- `GetBucketPolicy`
- `ListDirectoryBuckets`

您可以在 CloudTrail 主控台中檢視最近的事件。若要為 Amazon S3 儲存貯體建立持續的活動和事件記錄，您可以在 CloudTrail 主控台中建立追蹤。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的 Creating a trail。

> **Note**
>
> 對於 S3 Express 單一區域，不支援區域端點 (物件層級或資料平面) API 操作 (例如，`PutObject`或`GetObject`) 的 CloudTrail 記錄。

使用 AWS 監控工具來實作監控

監控是維持 Amazon S3 和 AWS 解決方案可靠性、安全性、可用性與效能的重要環節。AWS 會提供數種工具和服務，協助您監控 Amazon S3 和其他 AWS 服務。例如，您可以監控 Amazon S3 的 Amazon CloudWatch 指標，尤其是`BucketSizeBytes`和`NumberOfObjects`儲存指標。

存放在 S3 Express One Zone 儲存類別中的物件不會反映在 Amazon S3 的 `BucketSizeBytes` 和 `NumberOfObjects` 儲存指標中。不過，S3 Express One Zone 可支援 `BucketSizeBytes` 和 `NumberOfObjects` 儲存指標。若要查看您選擇的指標，您可以指定 `StorageType` 維度來區分 Amazon S3 儲存類別和 S3 Express One Zone 儲存類別。如需詳細資訊，請參閱 使用 Amazon 監控指標 CloudWatch。

如需更多詳細資訊，請參閱 使用 Amazon 監控指標 CloudWatch 及 監控 Amazon S3。

最佳化 Amazon S3 Express 單區域效能

Amazon S3 Express One Zone 是一種高效能的單一可用區域 (AZ) S3 儲存類別，專門為對延遲最敏感的應用程式提供一致的個位數毫秒資料存取。S3 Express One Zone 是最先讓您選擇在單一可用區域內共置高效能物件儲存與 Amazon Elastic Compute Cloud、Amazon Elastic Kubernetes Service 和 Amazon Elastic Container Service 等 AWS 運算資源的 S3 儲存類別。共置儲存與運算資源可將運算效能和成本最佳化，並提高資料處理速度。

S3 Express One Zone 提供與其他 S3 儲存類別相似的效能彈性，但具有一致的個位數毫秒第一位元組讀取和寫入請求延遲，與 S3 Standard 相比快了 10 倍。S3 Express One Zone 是全新設計的產品，支援的高載輸送量可達到非常高的彙總層級。S3 Express One Zone 儲存類別使用自訂建置的架構來達到最佳效能，並透過將資料存放在高效能硬體上來提供一致的低請求延遲。S3 Express One Zone 的物件通訊協定經增強後，簡化了身分驗證和中繼資料的額外負荷。

為了進一步提高存取速度並支援每秒數十萬個請求，S3 Express One Zone 會以新的儲存貯體類型儲存資料，也就是 Amazon S3 目錄儲存貯體。每個 S3 目錄儲存貯體每秒可支援數十萬筆交易 (TPS)。

S3 Express One Zone 結合高效能、專用硬體和軟體，提供了個位數毫秒的資料存取速度，以及可針對每秒大量交易進行擴展的目錄儲存貯體，使其成為最適合請求密集型操作或效能關鍵應用程式的 Amazon S3 儲存類別。

下列主題針對使用 S3 Express One Zone 儲存類別的應用程式，說明最佳化效能的最佳實務指導方針和設計模式。

主題

- S3 Express One Zone 的效能指導方針和設計模式

S3 Express One Zone 的效能指導方針和設計模式

建置從 Amazon S3 Express One Zone 上傳和擷取物件的應用程式時，請依照我們的最佳實務指導方針來最佳化效能。若要使用 S3 Express One Zone 儲存類別，您必須建立 S3 目錄儲存貯體。S3 Express One Zone 儲存類別不支援搭配 S3 一般用途儲存貯體使用。

如需所有其他 Amazon S3 儲存類別和 S3 一般用途儲存貯體的效能指導方針，請參閱 最佳實務設計模式：最佳化 Amazon S3 效能。

為了在使用 S3 Express One Zone 儲存類別和目錄儲存貯體時讓應用程式獲得最佳效能，建議採用下列指導方針與設計模式。

主題

- 將 S3 Express One Zone 儲存與您的 AWS 運算資源共置
- 目錄儲存貯體
- 目錄儲存貯體水平擴展請求平行化
- 使用工作階段型身分驗證
- S3 其他檢查總和最佳實務
- 使用最新版 AWS SDK 和通用執行期程式庫
- 效能故障診斷

將 S3 Express One Zone 儲存與您的 AWS 運算資源共置

每個目錄儲存貯體都存放在您建立儲存貯體時選取的單一可用區域中。您可以透過在運算工作負載或資源所在位置的可用區域中建立新的目錄儲存貯體來著手進行。然後就能立即開始非常低延遲的讀取和寫入。目錄儲存貯體是最先讓您在 AWS 區域 中選擇可用區域的 S3 儲存貯體，可降低運算和儲存之間的延遲。

如果您在可用區域之間存取目錄儲存貯體，延遲將會增加。為了達到最佳效能，建議您盡可能從位於相同可用區域中的 Amazon Elastic Container Service、Amazon Elastic Kubernetes Service 和 Amazon Elastic Compute Cloud 執行個體存取目錄儲存貯體。

目錄儲存貯體

每個目錄儲存貯體每秒可支援數十萬筆交易 (TPS)。與一般用途儲存貯體不同的是，目錄儲存貯體是以階層方式將索引鍵組織成目錄，而非字首。字首是物件索引鍵名稱開頭的字元字串。您可以將字首視為以類似於目錄的方式組織資料的一種方式。但是，字首不是目錄。

字首會在一般用途儲存貯體內的平面命名空間中組織資料，而且一般用途儲存貯體內的字首數目並無限制。每個前綴每秒至少可達到 3,500 PUTPOST/DELETE或 5,500 個GET/HEAD請求。您也可以在多個字首之間平行處理請求以擴展效能。不過，在讀取和寫入操作的情況下，此擴展會逐漸發生，而不是立即發生。一般用途儲存貯體逐漸擴展到新的更高請求率時，您可能會收到一些 HTTP 狀態碼 503 (服務無法使用) 錯誤。

使用階層式命名空間時，物件索引鍵中的分隔符號非常重要。唯一支援的分隔符號為正斜線 (/)。目錄是以分隔符號邊界決定。例如，物件索引鍵 `dir1/dir2/file1.txt` 會產生目錄 `dir1/` 並自動建立 `dir2/`，以及將物件 `file1.txt` 新增至路徑 `dir1/dir2/file1.txt` 中的 `/dir2` 目錄。

物件上傳至目錄儲存貯體時所建立的目錄沒有每個字首 TPS 的限制，而且會自動預先擴展以減少發生 HTTP 503 (服務無法使用) 錯誤的機會。這種自動擴展可讓您的應用程式視需要在目錄內和目錄之間平行處理讀取和寫入請求。

目錄儲存貯體水平擴展請求平行化

您可以向目錄儲存貯體發出多個並行請求，以將請求分散到不同的連線上來獲得最大可存取頻寬，藉此達到最佳效能。S3 Express One Zone 對於目錄儲存貯體的連線數沒有任何限制。當同一目錄發生大量並行寫入時，個別目錄可以水平和自動擴展效能。

一開始建立物件索引鍵且其索引鍵名稱包含目錄時，會自動為該物件建立目錄。後續物件上傳到同一個目錄時，就不需要建立目錄，這樣可以減少物件上傳至現有目錄的延遲。

雖然淺層和深層目錄結構都支援在目錄儲存貯體內儲存物件，但目錄儲存貯體會自動水平擴展，同時上傳至相同目錄或平行同級目錄時的延遲較低。

使用工作階段型身分驗證

S3 Express One Zone 和目錄儲存貯體支援新的工作階段型授權機制，可驗證和授權對目錄儲存貯體的請求。使用工作階段型身分驗證時，AWS SDK 會自動使用 `CreateSession` API 操作來建立臨時工作階段權杖，使用該權杖可對目錄儲存貯體進行低延遲的資料請求授權。

AWS SDK 使用 `CreateSession` API 操作請求臨時憑證，然後每 5 分鐘代表您自動建立和重新整理權杖。為了充分利用 S3 Express One Zone 儲存類別的效能優勢，我們建議您使用 AWS SDK 來啟動和管理 `CreateSession` API 請求。如需此工作階段型模型的詳細資訊，請參閱 [`CreateSession` 授權]。

S3 其他檢查總和最佳實務

S3 Express One Zone 可讓您選擇用於在上傳或下載過程中驗證資料的檢查總和演算法選項。您可以選擇以下 Secure Hash 演算法 (SHA) 或循環冗餘檢查 (CRC) 資料完整性演算法之一：CRC32、CRC32C、SHA-1 和 SHA-256。S3 快速單區域儲存類別不支援以 MD5 為基礎的總和檢查碼。

CRC32 是 AWS SDK 對 S3 Express One Zone 來回傳輸資料時使用的預設檢查總和。我們建議您使用 CRC32 和 CRC32C，以獲得 S3 Express One Zone 儲存類別的最佳效能。

使用最新版 AWS SDK 和通用執行期程式庫

另外還有數種 AWS SDK 提供 AWS 通用執行期 (CRT) 程式庫，可進一步加快 S3 用戶端的效能。這些 SDK 包括 AWS SDK for Java 2.x、AWS SDK for C++ 和 AWS SDK for Python (Boto3)。CRT 型 S3 用戶端會自動使用分段上傳 API 操作和位元組範圍擷取功能來自動執行水平擴展連線，藉此提升對 S3 Express One Zone 來回傳輸物件時的效能和可靠性。

若要利用 S3 Express One Zone 儲存類別達到最高效能，建議您使用包含 CRT 程式庫的最新版 AWS SDK，或使用 AWS Command Line Interface (AWS CLI)。

效能故障診斷

對延遲敏感的應用程式重試請求

S3 Express One Zone 專為提供一致的高效能水準而打造，無需額外調整。但是，設定有利的逾時值和重試次數，可進一步協助實現一致的延遲和效能。AWS 開發套件具有可設定的逾時和重試值，您可以將它們調整為特定應用程式的容錯值。

AWS 通用執行期 (CRT) 程式庫和 Amazon EC2 執行個體類型配對

執行大量讀取和寫入操作的應用程式，可能會比未執行這些操作的應用程式需要更多的記憶體或運算能力。為要求高效能的工作負載啟動 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體時，請選擇符合應用程式所需資源量的執行個體類型。S3 Express One Zone 高效能儲存最適合搭配較大和較新的執行個體類型，這些類型具備更大量的系統記憶體，以及更強大的 CPU 和 GPU，可充分利用效能更高的儲存。我們也建議使用具備 CRT 能力的最新版 AWS SDK，如此就能更進一步加速讀取和寫入請求。

在 AWS SDK 中使用工作階段型身分驗證，而非 HTTP REST API

使用 Amazon S3 時，您也可以依循與 AWS SDK 相同的最佳實務，在使用 HTTP REST API 請求時最佳化效能。不過，採用 S3 Express One Zone 所使用的工作階段型授權和身分驗證機制時，我們強烈建議您使用 AWS SDK 來管理 `CreateSession` 及其受管工作階段權杖。AWS SDK 會代表您使用 `CreateSession` API 操作自動建立和重新整理權杖。使用 `CreateSession` 可縮短為了授權每個請求，對 AWS Identity and Access Management (IAM) 的每個請求往返延遲。

使用 S3 Express One Zone 進行開發

Amazon S3 Express One Zone 是第一款可讓您選取單一可用區域的 S3 儲存類別，還可選擇將物件儲存體與運算資源共置，藉此盡可能提供最高存取速度。您可以透過 S3 Express One Zone 儲存類別使用 S3 目錄儲存貯體來存放資料。每個目錄儲存貯體都使用 S3 Express One Zone 儲存類別，將物件儲存在您建立儲存貯體時可選取的單一可用區域中。

建立目錄儲存貯體之後，您就可以立即開始進行非常低延遲的讀取和寫入。您可以透過虛擬私有雲端 (VPC) 使用端點連線與目錄儲存貯體進行通訊，或是使用區域 (Zone) 和區域 (Region) API 操作來管理物件和目錄儲存貯體。您也可以透過 Amazon S3 主控台、AWS Command Line Interface (AWS CLI)、AWS SDK 和 Amazon S3 REST API 來使用 S3 Express One Zone 儲存類別。

Amazon S3 快速單區域儲存類別的設計可在單一可用區域內達到 99.95% 的可用性，並受到 Amazon S3 服務水準協議的支援。使用 S3 Express One Zone，您的資料會以備援方式儲存在單一可用區域內的多部裝置上。S3 Express One Zone 的設計在於透過快速偵測並修復任何遺失的備援來處理並行裝置故障。如果現有裝置發生故障，S3 Express One Zone 會自動將請求轉送到可用區域內的新裝置。這種備援有助於確保存取可用區域內的資料不間斷。

主題

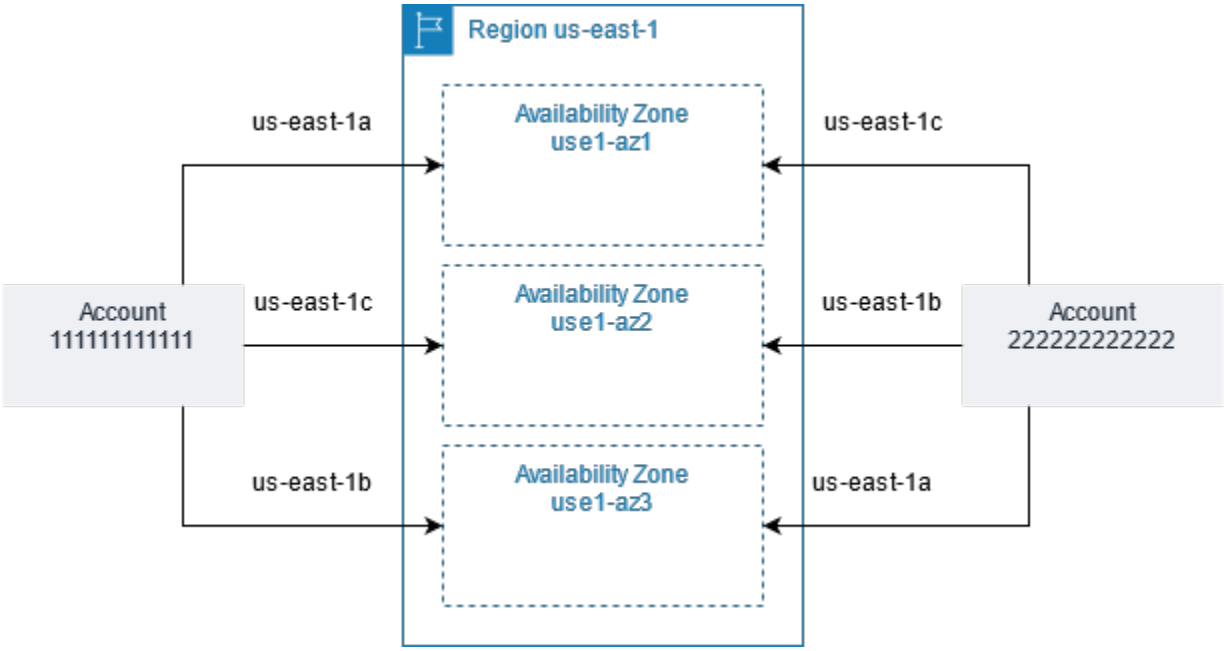
- S3 Express One Zone 可用區域和區域
- 區域和區域端點
- S3 Express One Zone API 操作

S3 Express One Zone 可用區域和區域

可用區域是一個或多個獨立的資料中心，具備 AWS 區域中的備援電源、聯網和連線能力。為了最佳化低延遲擷取，Amazon S3 Express One Zone 儲存類別中的物件會以備援方式儲存在運算工作負載本機上單一可用區域中的 S3 目錄儲存貯體中。建立目錄值區時，您可以選擇可用區域 AWS 區域 以及值區的所在位置。

AWS 將實體可用區域隨機對應至每個區域的可用區域名稱 AWS 帳戶。這種方法有助於將資源分配到可用區域中 AWS 區域，而不是可能集中在每個區域的第一個可用區域中的資源。因此，您的可`us-east-1a`用區域 AWS 帳戶 可能不代表與不同的實體位置相`us-east-1a`同 AWS 帳戶。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的區域和可用區域。

為協調各帳戶的可用區域，您必須使用 AZ ID，這是可用區域唯一且一致的識別符。例如，`use1-az1`是區域的 AZ ID，且每個`us-east-1`區域都有相同的實體位置 AWS 帳戶。下圖顯示每個帳戶的 AZ ID 都相同的情形，即使每個帳戶的可用區域名稱對應可能不同。



使用 S3 Express One Zone，您的資料會以備援方式儲存在單一可用區域內的多部裝置上。S3 快速單一區域的設計可在單一可用區域內達到 99.95% 的可用性，並受到 Amazon S3 服務等級協議的支援。如需更多資訊，請參閱單一可用區域

以下區域和可用區域支援 S3 Express One Zone：

S3 Express One Zone 支援的區域和可用區域

區域名稱	區域代碼	可用區域 ID
美國東部 (維吉尼亞北部)	us-east-1	use1-az4
		use1-az5
		use1-az6
美國西部 (奧勒岡)	us-west-2	usw2-az1
		usw2-az3
		usw2-az4
亞太區域 (東京)	ap-northeast-1	apne1-az1
		apne1-az4

區域名稱	區域代碼	可用區域 ID
歐洲 (斯德哥爾摩)	eu-north-1	eun1-az1
		eun1-az2
		eun1-az3

區域和區域端點

若要從虛擬私有雲端 (VPC) 存取 Amazon S3 Express One Zone 的區域端點，您可以使用閘道 VPC 端點。建立閘道端點後，您可以將其新增為路由表中的目標，用於從 VPC 到 S3 Express One Zone 的流量。使用閘道端點不需額外付費。如需如何設定閘道 VPC 端點的詳細資訊，請參閱 S3 Express One Zone 的網路功能。

當您使用 S3 Express One Zone 時，儲存貯體層級 (控制平面) API 操作可透過區域端點提供使用，並且稱為區域端點 API 操作。區域端點 API 操作的範例包括 `CreateBucket` 和 `DeleteBucket`。

建立目錄值區之後，您可以使用 Zonal (物件層級或資料平面端點 API 作業) 來上傳和管理目錄值區中的物件。區域端點 API 操作可透過區域端點提供使用。區域 API 操作的範例包括 `PutObject` 和 `CopyObject`。

S3 Express One Zone API 操作

Amazon S3 Express One Zone 儲存類別支援區域 (儲存貯體層級，也就是控制平面) 和區域 (物件層級，也就是資料平面) 端點 API 操作。如需詳細資訊，請參閱 S3 Express One Zone 的網路功能 及 端點和閘道 VPC 端點。

區域 (Region) 端點 API 操作

S3 Express One Zone 支援下列區域 (Region) 端點 API 操作：

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy
- ListDirectoryBuckets
- PutBucketPolicy

區域 (Zone) 端點 API 操作

S3 Express One Zone 支援下列區域 (Zone) 端點 API 操作：

- CreateSession
- CopyObject
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2
- PutObject
- AbortMultipartUpload
- CompleteMultiPartUpload
- CreateMultipartUpload
- ListMultipartUploads
- ListParts
- UploadPart
- UploadPartCopy

使用 Amazon S3 存取點管理資料存取

Amazon S3 存取點可簡化任何在 S3 中存放資料的 AWS 服務或客戶應用程式的資料存取。存取點為連接到儲存貯體的指定網路端點，您可以使用這些端點來執行 S3 物件操作，例如 `GetObject` 和 `PutObject`。每個存取點都有 S3 對於透過該存取點進行的任何請求所套用的不同許可和網路控制。每個存取點都會強制執行自訂的存取點政策，該政策可結合附加至基礎儲存貯體的儲存貯體政策運作。您可以將任何存取點設定為僅接受來自 Virtual Private Cloud (VPC) 的請求，以限制只能透過私人網路存取 Amazon S3 資料。您也可以為每個存取點設定自訂封鎖公開存取設定。

> **Note**
>
> - 您只能使用存取點對物件執行操作。您無法使用存取點來執行其他 Amazon S3 操作，例如修改或刪除儲存貯體。如需支援存取點的 S3 操作完整清單，請參閱存取點與 AWS 服務的相容性。
> - 存取點適用於部分 (但非全部) AWS 服務和功能。例如，您無法將跨區域複寫設定為透過存取點操作。如需與 S3 存取點相容的完整 AWS 服務清單，請參閱存取點與 AWS 服務的相容性。

本節說明如何使用 Amazon S3 存取點。如需有關使用儲存貯體的詳細資訊，請參閱 儲存貯體概觀。如需使用物件的資訊，請參閱「Amazon S3 物件概觀」。

主題

- 配置使用存取點的 IAM 原則
- 建立存取點
- 使用存取點
- 存取點的法規與限制

配置使用存取點的 IAM 原則

Amazon S3 存取點支援 AWS Identity and Access Management (IAM) 資源政策，可讓您根據資源、使用者或其他條件控制存取點的使用。若要讓應用程式或使用者能夠透過存取點存取物件，存取點和基礎儲存貯體都必須允許該請求。

> **Important**
>
> 將 S3 存取點新增到儲存貯體，不會變更透過儲存貯體的名稱或 Amazon Resource Name (ARN) 直接存取儲存貯體時儲存貯體的行為。針對儲存貯體的所有現有操作將繼續像以前一樣運作。您在存取點政策中包含的限制僅適用透過該存取點進行的請求。

使用 IAM 資源政策時，請務必先解決安全性警告、錯誤、一般警告和建議， AWS Identity and Access Management Access Analyzer 然後再儲存政策。IAM Access Analyzer 會比對 IAM 政策文法和最佳實務來執行政策檢查，以驗證您的政策。這些檢查會產生問題清單並提供建議，協助您撰寫具有功能性且符合安全最佳實務的政策。

若要進一步了解如何使用 IAM Access Analyzer 驗證政策，請參閱《IAM 使用者指南》中的 IAM Access Analyzer 政策驗證。若要檢視 IAM Access Analyzer 傳回的警告、錯誤和建議清單，請參閱 IAM Access Analyzer 政策檢查參考。

存取點政策範例

下列範例示範如何建立 IAM 原則以控制透過存取點進行的請求。

> **Note**
>
> 存取點政策中授予的權限只有在基礎儲存貯體也允許相同的存取時才有效。您可以透過兩種方式完成此操作：
>
> 1. (建議) 如將存取控制委派給存取點中所述，將儲存貯體的存取控制委派給存取點。
> 2. 將存取點政策中包含的相同權限新增至基礎儲存貯體的政策。範例 1 存取點政策範例示範如何修改基礎儲存貯體原則，以允許必要的存取。

Example 1 - 授予存取點政策

下列存取點政策會透過帳戶 *Jane* 中的存取點 *123456789012*，將帳戶 GET 中 IAM 使用者 PUT 的許可授予具有字首 *Jane/* 的 *my-access-point* 和 *123456789012* 物件。

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

```
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Jane"
        },
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point/
object/Jane/*"
    }]
}
```

> **Note**
>
> 若要讓存取點政策有效地授予存取給 *Jane*，基礎儲存貯體也必須允許對 *Jane* 的相同存取。您可以將存取控制從儲存貯體委派給存取點，如將存取控制委派給存取點中所述。或者，您可以將下列政策新增至基礎儲存貯體，將必要的權限授予 Jane。請注意，存取點和儲存貯體原則之間的 `Resource` 項目有所不同。
>
> ```
> {
>     "Version": "2012-10-17",
>     "Statement": [
>     {
>         "Effect": "Allow",
>         "Principal": {
>             "AWS": "arn:aws:iam::123456789012:user/Jane"
>         },
>         "Action": ["s3:GetObject", "s3:PutObject"],
>         "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/Jane/*"
>     }]
> }
> ```

Example 2 - 具有索引標籤條件的存取點政策

下列存取點政策會透過在帳戶 *123456789012* 中已將索引標籤金鑰 *data* 設定為 *finance* 值的物件存取點，將帳戶 *123456789012* 中 IAM 使用者 *Mateo* 的許可授予 GET 和 *my-access-point*。

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect":"Allow",
```

```
        "Principal" : {
            "AWS": "arn:aws:iam::123456789012:user/Mateo"
        },
        "Action":"s3:GetObject",
        "Resource" : "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point/
object/*",
        "Condition" : {
            "StringEquals": {
                "s3:ExistingObjectTag/data": "finance"
            }
        }
    }]
}
```

Example 3 - 允許儲存貯體清單的存取點政策

下列存取點政策允許帳戶 *123456789012* 中的 IAM 使用者 Arnav 檢視帳戶 *123456789012* 中儲存貯體基礎存取點 *my-access-point* 中包含的物件 。

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Arnav"
        },
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point"
    }]
}
```

Example 4 - 服務控制政策

下列服務控制政策需要使用虛擬私有雲端 (VPC) 網路來源建立所有新的存取點。有了此政策，組織中的使用者就無法建立可從網際網路存取的新存取點。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Deny",
        "Action": "s3:CreateAccessPoint",
```

```
        "Resource": "*",
        "Condition": {
            "StringNotEquals": {
                "s3:AccessPointNetworkOrigin": "VPC"
            }
        }
    }]
}
```

Example 5 - 將 S3 操作限制在 VPC 網路來源的儲存貯體政策

以下儲存貯體政策會將儲存貯體 *example-s3-bucket* 的所有 S3 物件操作的存取限制在具有 VPC 網路來源的存取點。

> **Important**
>
> 使用類似此範例所示的陳述式之前，請確定您不需要使用存取點不支援的功能，例如跨區域複寫。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:BypassGovernanceRetention",
                "s3:DeleteObject",
                "s3:DeleteObjectTagging",
                "s3:DeleteObjectVersion",
                "s3:DeleteObjectVersionTagging",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectLegalHold",
                "s3:GetObjectRetention",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging",
                "s3:ListMultipartUploadParts",
```

```
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectLegalHold",
                "s3:PutObjectRetention",
                "s3:PutObjectTagging",
                "s3:PutObjectVersionAcl",
                "s3:PutObjectVersionTagging",
                "s3:RestoreObject"
            ],
            "Resource": "arn:aws:s3:::example-s3-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:AccessPointNetworkOrigin": "VPC"
                }
            }
        }
    ]
}
```

條件索引鍵

S3 存取點具有您可以在 IAM 政策中使用的條件金鑰，來控制對資源的存取。下列條件金鑰僅代表 IAM 政策的一部分。如需完整政策範例，請參閱 存取點政策範例、the section called “將存取控制委派給存取點” 和 the section called “授予跨帳戶存取點的許可”。

s3:DataAccessPointArn

此範例顯示您可以用來比對存取點 ARN 的字串。下列範例會比對區域 AWS 帳戶 *123456789012*中的所有存取點 *us-west-2*：

```
"Condition" : {
    "StringLike": {
        "s3:DataAccessPointArn": "arn:aws:s3:us-west-2:123456789012:accesspoint/*"
    }
}
```

s3:DataAccessPointAccount

此範例顯示一個字串運算子，您可以使用它來比對存取點擁有者的帳戶 ID。下列範例會比對 AWS 帳戶 *123456789012* 擁有的所有存取點。

```
"Condition" : {
```

```
    "StringEquals": {
        "s3:DataAccessPointAccount": "123456789012"
    }
}
```

s3:AccessPointNetworkOrigin

此範例顯示一個字串運算子，您可以使用它來比對網路來源，Internet 或 VPC。下列範例僅會比對存取點與 VPC 來源。

```
"Condition" : {
    "StringEquals": {
        "s3:AccessPointNetworkOrigin": "VPC"
    }
}
```

如需將條件金鑰與 Amazon S3 搭配使用的詳細資訊，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

將存取控制委派給存取點

您可以將儲存貯體的存取控制委派給儲存貯體的存取點。下列範例儲存貯體政策允許完整存取儲存貯體擁有者帳戶所擁有的所有存取點。因此，對此儲存貯體的所有存取皆由連接至其存取點的政策所控制。我們建議您針對不需要直接存取儲存貯體的所有使用案例，以此方式設定儲存貯體。

Example 6 - 將存取控制委派給存取點的儲存貯體政策

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "*",
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Bucket owner's account
 ID" }
        }
    }]
}
```

授予跨帳戶存取點的許可

若要建立另一個帳戶所擁有之儲存貯體的存取點，您必須先指定儲存貯體名稱和帳戶擁有者 ID 來建立存取點。然後，儲存貯體擁有者必須更新儲存貯體政策，以授權來自存取點的請求。建立存取點類似於建立 DNS CNAME，其中存取點不會提供儲存貯體內容的存取權。所有儲存貯體存取權都由儲存貯體政策控制。下列範例儲存貯體政策允許儲存貯體上來自受信任 AWS 帳戶所擁有之存取點的 GET 和 LIST 請求。

將*#### ARN* 取代為值區的 ARN。

Example 7 — 儲存貯體政策委派權限給另一個 AWS 帳戶

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : ["s3:GetObject","s3:ListBucket"],
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Access point owner's
 account ID" }
        }
    }]
}
```

建立存取點

Amazon S3 提供建立和管理存取點的功能。您可以使用 AWS Management Console、 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 Amazon S3 REST API 來建立 S3 存取點。

依預設，您可以為 AWS 帳戶的每個區域建立最多 10,000 個存取點。如果對單一區域中的單一帳戶需要超過 10,000 個存取點，您可以請求增加服務配額。如需服務配額和請求增加的詳細資訊，請參閱《AWS 一般參考》中的 AWS Service Quotas。

> **Note**
>
> 由於您可能想要公開存取點名稱，讓其他使用者可以使用存取點，因此請避免在存取點名稱中新增敏感資訊。存取點名稱發佈在稱為網域名稱系統 (DNS) 的可公開存取資料庫中。

命名 Amazon S3 存取點的規則

存取點名稱必須符合下列條件：

- 在單一 AWS 帳戶 和區域內必須是唯一的
- 必須符合 DNS 命名限制
- 必須以數字或小寫字母開頭
- 長度必須介於 3 與 50 個字元之間
- 不能以連字號 (-) 開頭和結尾
- 不能包含底線 (_)、大寫字母或句點 (.)
- 無法以尾碼 `-s3alias` 結尾。存取點別名名稱會保留此尾碼。如需詳細資訊，請參閱 針對您的 S3 儲存貯體存取點使用儲存貯體樣式別名。

若要建立存取點，請參閱下列主題。

主題

- 建立存取點
- 建立受限於 Virtual Private Cloud 的存取點
- 管理存取點的公開存取

建立存取點

一個存取點只會與一個 Amazon S3 儲存貯體相關聯。如果您想要在中使用值區 AWS 帳戶，您必須先建立值區。如需建立儲存貯體的詳細資訊，請參閱「建立、設定和使用 Amazon S3 儲存貯體」。

只要知道儲存貯體名稱和儲存貯體擁有者的帳戶 ID，您就可以建立與另一個 AWS 帳戶中儲存貯體相關聯的跨帳戶存取點。不過，建立跨帳戶存取點並不會授予您存取儲存貯體中資料的權限，直到您獲授予儲存貯體擁有者的許可。儲存貯體擁有者必須透過儲存貯體政策授予存取點擁有者帳戶 (您的帳戶) 存取儲存貯體的權限。如需詳細資訊，請參閱 授予跨帳戶存取點的許可。

依預設，您可以為 AWS 帳戶的每個區域建立最多 10,000 個存取點。如果對單一區域中的單一帳戶需要超過 10,000 個存取點，您可以請求增加服務配額。如需服務配額和請求增加的詳細資訊，請參閱《AWS 一般參考》中的 AWS Service Quotas。

下列範例示範如何使用 AWS CLI 和 S3 主控台建立存取點。如需如何使用 REST API 建立存取點的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 CreateAccessPoint。

使用 S3 主控台

建立存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在頁面頂端的導覽列中，選擇目前顯示的名稱 AWS 區域。接下來，選擇您要在其中建立存取點的區域。
3. 在左側導覽窗格中，選擇 Access Points (存取點)。
4. 在 Access Points (存取點) 頁面上，選擇 Create access point (建立存取點)。
5. 在存取點名稱欄位中，輸入存取點名稱。如需存取點命名的詳細資訊，請參閱「命名 Amazon S3 存取點的規則」。
6. 針對儲存貯體名稱，請指定您要與存取點搭配使用的 S3 儲存貯體。

   若要使用帳戶中的儲存貯體，請選取選擇此帳戶中的儲存貯體，然後輸入或瀏覽儲存貯體名稱。

   若要使用其他值區中的值區 AWS 帳戶，請選擇「在其他帳戶中指定值區」，然後輸入值區的 AWS 帳戶 ID 和名稱。

   > **Note**
   >
   > 如果您使用其他值區中的值區 AWS 帳戶，值區擁有者必須更新值區政策，以授權來自存取點的要求。如需儲存貯體政策範例，請參閱「授予跨帳戶存取點的許可」。

7. 選擇一個網路來源。如果您選擇 Virtual Private Cloud (VPC)，請輸入要與存取點搭配使用的 VPC ID。

   如需存取點網路來源的詳細資訊，請參閱「建立受限於 Virtual Private Cloud 的存取點」。
8. 在 Block Public Access settings for this Access Point (存取點的封鎖公開存取權限設定)下，選取要套用至該存取點的封鎖公開存取權限設定。依預設，新存取點的所有封鎖公開存取設定都會啟用。建議您啟用所有設定，除非您知道您有特別需要停用任一設定。

   > **Note**
   >
   > 建立存取點之後，您便無法變更其封鎖公有存取設定。

   如需對存取點使用 Amazon S3 封鎖公開存取的詳細資訊，請參閱「管理存取點的公開存取」。

9. (選用) 在 Access point policy - optional (存取點政策 - 選用)下，指定存取點政策。在儲存您的政策之前，請務必解決任何安全性警告、錯誤、一般警告，以及建議。如需指定存取點政策的詳細資訊，請參閱「存取點政策範例」。
10. 選擇 Create access point (建立新的存取點)。

使用 AWS CLI

下列範例命令會針對帳戶 *111122223333* 中的儲存貯體 *DOC-EXAMPLE-BUCKET* 建立名為 *example-ap* 的存取點。若要建立存取點，請將指定下列內容的請求傳送至 Amazon S3：

- 存取點名稱。如需命名規則的詳細資訊，請參閱the section called “命名 Amazon S3 存取點的規則”。
- 您要存取點與其建立關聯的儲存貯體名稱。
- 擁有值區 AWS 帳戶 的帳戶 ID。

```
aws s3control create-access-point --name example-ap --account-id 111122223333 --
bucket DOC-EXAMPLE-BUCKET
```

當您在不同的值區中使用存取點來建立存取點時 AWS 帳戶，請加入`--bucket-account-id`參數。下列範例命令會使用位於 AWS 帳戶 *444455556666* 的儲存貯體 *DOC-EXAMPLE-BUCKET2*，在 AWS 帳戶 *111122223333* 中建立存取點。

```
aws s3control create-access-point --name example-ap --account-id 111122223333 --
bucket DOC-EXAMPLE-BUCKET --bucket-account-id 444455556666
```

建立受限於 Virtual Private Cloud 的存取點

建立存取點時，您可以選擇讓存取點可從網際網路存取，也可以指定透過該存取點提出的所有請求都必須來自特定 Virtual Private Cloud (VPC)。可從網際網路存取的存取點表示具有 `Internet` 的網路來源。可以從網際網路上的任何地方使用，受限於存取點、基礎儲存貯體和相關資源 (例如請求的物件) 既有的任何其他存取限制。只能從指定 VPC 存取的存取點具有 VPC 的網路原始伺服器，並且 Amazon S3 會拒絕對非源自該 VPC 的存取點進行的任何請求。

> **Important**
>
> 您只能在建立存取點時指定存取點的網路來源。建立存取點之後，您便無法變更其網路來源。

若要將存取點限制在僅限 VPC 存取，請在建立存取點的請求中包含 VpcConfiguration 參數。在 VpcConfiguration 參數中，您可以指定要能夠使用存取點的 VPC ID。如果透過存取點發出要求，則要求必須來自 VPC，否則 Amazon S3 將拒絕該要求。

您可以使用 AWS CLI、 AWS SDK 或 REST API 擷取存取點的網路原點。如果存取點已指定 VPC 組態，則其網路來源為 VPC。否則，存取點的網路來源為 Internet。

Example

範例：建立限制 VPC 存取的存取點

下列範例會在帳戶 123456789012 中的儲存貯體 example-bucket 建立名為 example-vpc-ap 的存取點，僅允許來自 vpc-1a2b3c VPC 的存取。然後，此範例會驗證新的存取點是否具有 VPC 的網路來源。

AWS CLI

```
aws s3control create-access-point --name example-vpc-ap --account-id 123456789012 --
bucket example-bucket --vpc-configuration VpcId=vpc-1a2b3c
```

```
aws s3control get-access-point --name example-vpc-ap --account-id 123456789012

{
    "Name": "example-vpc-ap",
    "Bucket": "example-bucket",
    "NetworkOrigin": "VPC",
    "VpcConfiguration": {
        "VpcId": "vpc-1a2b3c"
    },
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2019-11-27T00:00:00Z"
}
```

若要使用存取點搭配 VPC，您必須修改 VPC 端點的存取原則。VPC 端點會允許流量從您的 VPC 流向 Amazon S3。它們會具有存取控制政策，可控制如何允許 VPC 內的資源與 Amazon S3 互動。只有在

VPC 端點政策同時授予存取點和基礎儲存貯體的存取時，透過存取點從 VPC 到 Amazon S3 的請求才會成功。

> **Note**
>
> 若要讓資源只能在 VPC 中存取，請務必為您的 VPC 端點建立私有託管區域。若要使用私有託管區域，請修改您的 VPC 設定以便 VPC 網路屬性、`enableDnsHostnames` 和 `enableDnsSupport` 設定為 `true`。

下列範例政策陳述式會設定 VPC 端點，以允許呼叫 `GetObject` 來取得名為 `awsexamplebucket1` 的儲存貯體和名為 `example-vpc-ap` 的存取點。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::awsexamplebucket1/*",
            "arn:aws:s3:us-west-2:123456789012:accesspoint/example-vpc-ap/object/*"
        ]
    }]
}
```

> **Note**
>
> 此範例中的 `"Resource"` 宣告使用 Amazon Resource Name (ARN) 來指定存取點。如需存取點 ARN 的更多資訊，請參閱 使用存取點。

如需 VPC 端點政策的詳細資訊，請參閱《VPC 使用者指南》中的使用 Amazon S3 的端點政策。

管理存取點的公開存取

Amazon S3 存取點對每個存取點支援獨立的封鎖公開存取設定。建立存取點時，您可以指定適用該存取點的封鎖公開存取設定。對於透過存取點提出的任何請求，Amazon S3 會評估該存取點、基礎儲存貯體和儲存貯體擁有者帳戶的封鎖公開存取設定。如果這些設定中有任一指出應該封鎖請求，則 Amazon S3 會拒絕請求。

如需 S3 封鎖公開存取功能的詳細資訊，請參閱封鎖對 Amazon S3 儲存體的公開存取權。

> **Important**
>
> - 依預設，存取點的所有封鎖公開存取設定都會啟用。您必須明確停用您不想套用至存取點的任何設定。
> - Amazon S3 目前不支援在建立存取點後變更存取點的封鎖公開存取權限設定。

Example

範例：建立使用自訂封鎖公開存取設定的存取點

此範例會使用非預設的封鎖公開存取設定，為帳戶 123456789012 中儲存貯體 example-bucket 名為 example-ap 的存取點。然後此範例會擷取新存取點的組態，以驗證其封鎖公開存取設定。

AWS CLI

```
aws s3control create-access-point --name example-ap --account-id
 123456789012 --bucket example-bucket --public-access-block-configuration
 BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=true,RestrictPublicBuckets=t
```

```
aws s3control get-access-point --name example-ap --account-id 123456789012

{
    "Name": "example-ap",
    "Bucket": "example-bucket",
    "NetworkOrigin": "Internet",
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": false,
        "IgnorePublicAcls": false,
        "BlockPublicPolicy": true,
```

```
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2019-11-27T00:00:00Z"
}
```

使用存取點

您可以使用、、 AWS 開發套件或 S3 REST API AWS Management Console， AWS CLI透過存取點存取 Amazon S3 儲存貯體中的物件。

存取點具有 Amazon Resource Name (ARN)。存取點 ARN 類似於儲存貯體 ARN，但它們經過明確輸入且將存取點的區域和存取點擁有者的 AWS 帳戶 ID 編碼。如需 ARN 的詳細資訊，請參閱《AWS 一般參考》中的 Amazon Resource Name (ARN)。

存取點 ARN 使用格式 `arn:aws:s3:region:account-id:accesspoint/resource`。例如：

- arn:aws:s3:us-west-2:123456789012:accesspoint/test 代表區域 `test` 的帳戶 `123456789012` 所擁有的存取點，名為 `us-west-2`。
- arn:aws:s3:us-west-2:123456789012:accesspoint/* 代表區域 `123456789012` 中帳戶 `us-west-2` 下的所有存取點。

透過存取點存取物件的 ARN 會使用格式 `arn:aws:s3:region:account-id:accesspoint/access-point-name/object/resource`。例如：

- arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/unit-01 代表物件 `unit-01`，透過區域 `test` 的帳戶 `123456789012` 所擁有名為 `us-west-2` 的存取點存取。
- arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/* 代表區域 `test` 的帳戶 `123456789012` 中存取點 `us-west-2` 的所有物件。
- arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/unit-01/finance/* 代表區域 `unit-01/finance/` 的帳戶 `test` 中，存取點 `123456789012` 字首 `us-west-2` 下的所有物件。

透過 S3 存取點存取儲存貯體

S3 存取點僅支援 virtual-host-style 定址。若要定址透過存取點的儲存貯體，請使用下列格式。

```
https://AccessPointName-AccountId.s3-accesspoint.region.amazonaws.com
```

Note

- 如果您的存取點名稱包含破折號 (-) 字元，請在 URL 中加入破折號，並在帳戶 ID 之前插入另一個破折號。例如，若要使用區域 `finance-docs` 中由 123456789012 帳戶所擁有名為 `us-west-2` 的存取點，則適當的 URL 是 `https://finance-docs-123456789012.s3-accesspoint.us-west-2.amazonaws.com`。
- S3 存取點不支援 HTTP 存取，只能透過 HTTPS 進行安全存取。

主題

如果您具有虛擬私有雲端 (VPC)，請參閱使用 VPC 端點和 S3 存取點管理 Amazon S3 存取。

監控與記錄存取點

Amazon S3 會記錄透過存取點和對管理存取點之 API 提出的請求，例如 `CreateAccessPoint` 和 `GetAccessPointPolicy`。若要監控和管理使用模式，您也可以為存取點設定 Amazon CloudWatch Logs 請求指標。

主題

CloudWatch 要求量度

若要瞭解並改善使用存取點之應用程式的效能，您可以使 CloudWatch 用 Amazon S3 請求指標。請求指標有助您監控 Amazon S3 請求，以快速找出並處理操作問題。

請求指標預設會在儲存貯體層級提供。不過，您可以使用共用字首、物件標籤或存取點來定義請求指標的篩選條件。當您建立存取點篩選條件時，請求指標組態會包含對您指定存取點的請求。您可以接收指標、設定警示以及存取儀表板，以檢視透過此存取點執行的即時操作。

您必須在主控台中設定請求指標或使用 Amazon S3 API 來選擇請求指標。請求指標會在一些處理延遲之後，以 1 分鐘的間隔提供。請求指標的費率與 CloudWatch 自訂指標相同。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

若要建立依存取點篩選的請求指標組態，請參閱 建立依字首、物件標籤或存取點篩選的指標組態。

請求日誌

您可以記錄透過存取點和對管理存取點之 API 發出的請求，例如使用伺服器存取記錄和 AWS CloudTrail的 `CreateAccessPoint` 和 `GetAccessPointPolicy`,。

CloudTrail 透過存取點發出之要求的記錄項目包括記錄`resources`區段中的存取點 ARN。

舉例而言，假設您的組態如下：

- 區域 `us-west-2` 中名為 `DOC-EXAMPLE-BUCKET1` 的儲存貯體，其中包含名為 `my-image.jpg` 的物件
- 與 `DOC-EXAMPLE-BUCKET1` 關聯且名為 `my-bucket-ap` 的存取點
- 的 AWS 帳戶 識別碼 `123456789012`

下列範例顯示先前組態之 CloudTrail 記錄項目的`resources`區段：

```
"resources": [
        {"type": "AWS::S3::Object",
            "ARN": "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/my-image.jpg"
        },
        {"accountId": "123456789012",
            "type": "AWS::S3::Bucket",
            "ARN": "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
        },
        {"accountId": "123456789012",
            "type": "AWS::S3::AccessPoint",
            "ARN": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-bucket-ap"
        }
    ]
```

如需 S3 伺服器存取日誌的詳細資訊，請參閱 使用伺服器存取記錄記錄要求。如需有關的詳細資訊 AWS CloudTrail，請參閱什麼是 AWS CloudTrail？ 在《AWS CloudTrail 使用者指南》中。

透過 Amazon S3 主控台使用 Amazon S3 存取點

本節說明如何使用 AWS Management Console來管理和使用您的 Amazon S3 存取點。開始之前，請導覽至存取點的詳細資訊頁面，以管理或使用此存取點，如下列程序所述。

主題

列出您帳戶的存取點

若要列出所有建立的存取點 AWS 帳戶

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在頁面頂端的導覽列中，選擇目前顯示的名稱 AWS 區域。接下來，選擇您要列出存取點的區域。
3. 在主控台左側的導覽窗格中，選擇 Access points (存取點)。
4. 在「存取點」頁面的「存取點」下，檢視 AWS 區域.
5. (選用) 在 Region (區域) 下拉式功能表旁邊的文字欄位中，輸入名稱，依名稱搜尋存取點。
6. 選擇存取點的名稱以管理或使用此存取點。

列出儲存貯體的存取點

列出單一儲存貯體 AWS 帳戶 的所有存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在頁面頂端的導覽列中，選擇目前顯示的名稱， AWS 區域 然後選擇您要列出存取點的地區。
3. 在主控台左側的導覽窗格中，選擇 Buckets (儲存貯體)。

4. 在 Buckets (儲存貯體) 頁面上，選取儲存貯體的名稱以列出其存取點。
5. 在儲存貯體詳細資料頁面上，選擇 Access points (存取點) 標籤。
6. 選擇存取點的名稱以管理或使用此存取點。

檢視某個存取點的組態詳細資訊

1. 導覽至存取點詳細資訊頁面，以檢視此存取點的詳細資訊，如列出您帳戶的存取點中所述。
2. 在 Access point overview (存取點概觀)下，檢視所選存取點的組態詳細資訊和屬性。

使用存取點

1. 導覽至存取點詳細資訊頁面以使用此存取點，如列出您帳戶的存取點中所述。
2. 在 Objects (物件) 標籤下，選擇您要透過存取點來存取的一或多個物件的名稱。在物件操作頁面上，主控台會在儲存貯體名稱上方顯示標籤，以表示您目前使用的存取點。當您使用存取點時，只能執行存取點許可所允許的物件操作。

> **Note**
>
> - 主控台視圖一律會顯示儲存貯體中的所有物件。如此程序所述使用存取點會限制您可以在這些物件上執行的操作，但不會限制您是否能看到它們存在於儲存貯體中。
> - S3 管理主控台不支援使用 Virtual Private Cloud (VPC) 存取點來存取儲存貯體資源。若要從 VPC 存取點存取儲存貯體資源，請使用 AWS CLI、 AWS 開發套件或 Amazon S3 REST API。

檢視某個存取點的封鎖公開存取設定

1. 導覽至存取點詳細資訊頁面，以檢視此存取點的設定，如列出您帳戶的存取點中所述。
2. 選擇 Permissions (許可)。
3. 在 Access point policy (存取點政策)下，檢閱存取點的封鎖公開存取設定。

> **Note**
>
> 建立存取點後，您就無法變更存取點的封鎖公開存取設定。

編輯存取點政策

1. 導覽至存取點詳細資訊頁面，以編輯此存取點的政策，如列出您帳戶的存取點中所述。
2. 選擇 Permissions (許可)。
3. 在 Access point policy (存取點政策)下，選擇 Edit (編輯)。
4. 在文字欄位中輸入存取點政策。主控台會自動顯示存取點的 Amazon Resource Name (ARN)，您可以在政策中使用該名稱。

刪除存取點

1. 導覽至您的帳戶或特定儲存貯體的存取點清單，如列出您帳戶的存取點中所述。
2. 選取您要刪除的存取點名稱旁的選項按鈕。
3. 選擇 Delete (刪除)。
4. 在出現的文字欄位中，輸入存取點名稱，以確認您要刪除此存取點，然後選擇 Delete (刪除)。

針對您的 S3 儲存貯體存取點使用儲存貯體樣式別名

在您建立存取點時，Amazon S3 會自動產生可使用的別名，而不是儲存貯體名稱來進行資料存取。您可以針對存取點資料平面操作中使用此存取點別名，而不是 Amazon Resource Name (ARN)。如需這些操作的清單，請參閱 存取點與 AWS 服務的相容性。

下面顯示了名稱為 *my-access-point* 的存取點的 ARN 和存取點別名範例。

- ARN – `arn:aws:s3:`*region*`:`*account-id*`:accesspoint/`*my-access-point*
- 存取點別名 – *my-access-point*`-`*hrzrlukc5m36ft7okagglf3gmwluquse1b*`-s3alias`

如需 ARN 的詳細資訊，請參閱《AWS 一般參考》中的 Amazon Resource Name (ARN)。

存取點別名

存取點別名是在與 Amazon S3 儲存貯體相同的命名空間內建立的。此別名會自動產生且無法變更。存取點別名符合有效 Amazon S3 儲存貯體名稱的所有要求，並由下列部分組成：

access point prefix`-`*metadata*`-s3alias`

> **Note**
>
> 存取點別名會保留該 `-s3alias` 尾碼，且該尾碼不能用於儲存貯體或存取點名稱。如需 Amazon S3 儲存貯體命名規則的詳細資訊，請參閱儲存貯體命名規則。

存取點別名使用案例和限制

採用存取點時，可以使用存取點別名，而不需要進行大量的程式碼變更。

建立存取點時，Amazon S3 會自動產生存取點別名，如下列範例所示。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control create-access-point --bucket example-s3-bucket1 --name my-access-point --
account-id 111122223333
{
    "AccessPointArn":
    "arn:aws:s3:region:111122223333:accesspoint/my-access-point",
    "Alias": "my-access-point-aqfqprnstn7aefdfbarligizwgyfouse1a-s3alias"
}
```

您可以在任何資料平面操作中使用此存取點別名，而不是 Amazon S3 儲存貯體名稱。如需這些操作的清單，請參閱 存取點與 AWS 服務的相容性。

下列`get-object`命令 AWS CLI 範例會使用值區的存取點別名來傳回指定物件的相關資訊。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3api get-object --bucket my-access-point-aqfqprnstn7aefdfbarligizwgyfouse1a-
s3alias --key dir/my_data.rtf my_data.rtf

{
    "AcceptRanges": "bytes",
    "LastModified": "2020-01-08T22:16:28+00:00",
    "ContentLength": 910,
    "ETag": "\"00751974dc146b76404bb7290f8f51bb\"",
    "VersionId": "null",
    "ContentType": "text/rtf",
    "Metadata": {}
}
```

限制

- 客戶無法設定別名。
- 存取點上的別名無法刪除、修改或停用。
- 您可以在某些資料平面操作中使用此存取點別名，而不是 Amazon S3 儲存貯體名稱。如需這些操作的清單，請參閱 存取點與 S3 操作的相容性.
- Amazon S3 控制平面操作無法使用存取點別名。如需 Amazon S3 控制平面操作的清單，請參閱 Amazon Simple Storage Service API 參考中的 Amazon S3 控制。
- 您無法在 Amazon S3 主控台中使用 S3 存取點別名做為 Move 操作的來源或目的地。
- 別名無法用於 AWS Identity and Access Management (IAM) 政策。
- 別名不能用作 S3 伺服器存取日誌的記錄目的地。
- 別名無法用作記錄 AWS CloudTrail 檔的記錄目的地。
- Amazon SageMaker GroundTruth 不支持接入點別名。

使用存取點進行相容的 Amazon S3 操作

下列範例示範如何在 Amazon S3 中搭配相容操作使用存取點。

主題

- 存取點與 AWS 服務的相容性
- 存取點與 S3 操作的相容性
- 透過存取點要求物件
- 透過存取點上傳物件
- 透過存取點刪除物件
- 透過存取點別名列出物件
- 透過存取點將標籤集新增至物件
- 透過使用 ACL 的存取點授予許可

存取點與 AWS 服務的相容性

Amazon S3 存取點別名允許需要 S3 儲存貯體名稱的應用程式，輕鬆使用存取點。您可以在使用 S3 存取點名稱的任何位置，使用 S3 儲存貯體名稱來存取 S3 中的資料。如需詳細資訊，請參閱 存取點別名使用案例和限制。

存取點與 S3 操作的相容性

您可以使用下列 Amazon S3 API 子集，透過存取點來存取儲存貯體：下列所有操作都可以接受存取點 ARN 或存取點別名：

S3 操作

- AbortMultipartUpload
- CompleteMultipartUpload
- CopyObject (僅限同一區域的副本)
- CreateMultipartUpload
- DeleteObject
- DeleteObjectTagging
- GetBucketAcl
- GetBucketCors
- GetBucketLocation
- GetBucketNotificationConfiguration
- GetBucketPolicy
- GetObject
- GetObjectAcl
- GetObjectAttributes
- GetObjectLegalHold
- GetObjectRetention
- GetObjectTagging
- HeadBucket
- HeadObject
- ListMultipartUploads
- ListObjects
- ListObjectsV2
- ListObjectVersions
- ListParts

- Presign
- PutObject
- PutObjectLegalHold
- PutObjectRetention
- PutObjectAcl
- PutObjectTagging
- RestoreObject
- UploadPart
- UploadPartCopy (僅限同一區域的副本)

透過存取點要求物件

下列範例會透過區域 us-west-2 中帳戶 ID 123456789012 所擁有的存取點 prod 要求 my-image.jpg 物件，並將下載的檔案儲存為 download.jpg。

AWS CLI

```
aws s3api get-object --key my-image.jpg --bucket arn:aws:s3:us-
west-2:123456789012:accesspoint/prod download.jpg
```

透過存取點上傳物件

下列範例會透過區域 us-west-2 中帳戶 ID 123456789012 所擁有的存取點別名 my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b-s3alias 來上傳物件 my-image.jpg。

AWS CLI

```
aws s3api put-object --bucket my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b-
s3alias --key my-image.jpg --body my-image.jpg
```

透過存取點刪除物件

下列範例會透過區域 us-west-2 中帳戶 ID 123456789012 所擁有的存取點 prod 刪除物件 my-image.jpg。

AWS CLI

```
aws s3api delete-object --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/prod
 --key my-image.jpg
```

透過存取點別名列出物件

下列範例會透過區域 us-west-2 中帳戶 ID 123456789012 所擁有的存取點別名 my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b-s3alias 來列出物件。

AWS CLI

```
aws s3api list-objects-v2 --bucket my-access-point-
hrzrlukc5m36ft7okagglf3gmwluquse1b-s3alias
```

透過存取點將標籤集新增至物件

下列範例透過區域 us-west-2 中帳戶 ID 123456789012 所擁有的存取點 prod 新增標籤集至現有的物件 my-image.jpg。

AWS CLI

```
aws s3api put-object-tagging --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/
prod --key my-image.jpg --tagging TagSet=[{Key="finance",Value="true"}]
```

透過使用 ACL 的存取點授予許可

下列範例示範如何透過區域 us-west-2 中帳戶 ID 123456789012 所擁有的存取點 prod 套用 ACL 至現有的物件 my-image.jpg。

AWS CLI

```
aws s3api put-object-acl --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/prod
 --key my-image.jpg --acl private
```

存取點的法規與限制

Amazon S3 存取點具有以下法規與限制：

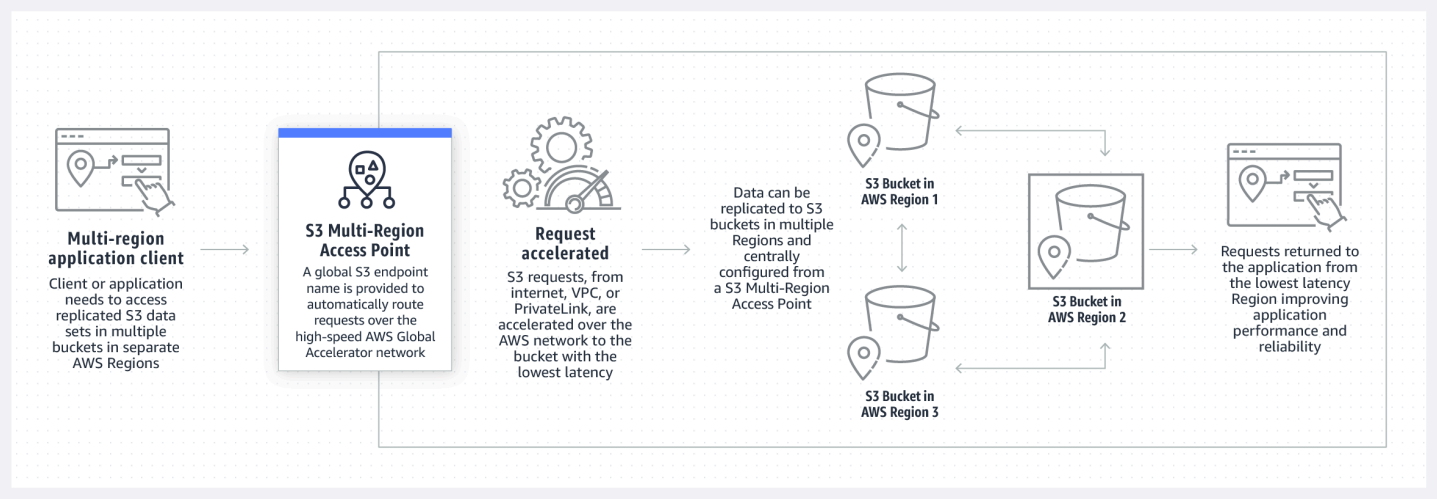
- 每個存取點只會與確切一個儲存貯體相關聯，您必須在建立存取點時指定該儲存貯體。建立存取點之後，您即無法將其與不同的儲存貯體建立關聯。不過，您可以刪除存取點，然後使用相同的名稱建立另一個存取點，並將該新的存取點與不同的儲存貯體建立關聯。
- 存取點名稱必須符合特定條件。如需存取點命名的詳細資訊，請參閱「命名 Amazon S3 存取點的規則」。
- 建立存取點之後，您便無法變更其 Virtual Private Cloud (VPC) 組態。
- 存取點政策的大小限制為 20 KB。
- 每 AWS 帳戶 個區域最多可以建立 10,000 個存取點。如果對單一區域中的單一帳戶需要超過 10,000 個存取點，您可以請求增加服務配額。如需服務配額和請求增加的詳細資訊，請參閱《AWS 一般參考》中的 AWS Service Quotas。
- 如果 AWS 區域 您擁有超過 1,000 個存取點，則無法在 Amazon S3 主控台中依名稱搜尋存取點。
- 您無法使用存取點作為 S3 複寫的目標。如需複寫的詳細資訊，請參閱 複製物件概觀。
- 您無法在 Amazon S3 主控台中使用 S3 存取點別名做為 Move 操作的來源或目的地。
- 您只能使用 virtual-host-style URL 來定址存取點。如需 virtual-host-style 定址的更多資訊，請參閱存取及列出 Amazon S3 儲存貯體。
- 控制存取點功能的 API 操作 (例如 `PutAccessPoint` 和 `GetAccessPointPolicy`) 不支援跨帳戶呼叫。
- 使用 REST API 向存取點發出要求時，您必須使用 AWS 簽章版本 4。如需驗證請求的詳細資訊，請參閱 Amazon 簡單儲存服務 API 參考中的驗證請求 (AWS 簽名版本 4)。
- 存取點僅支援透過 HTTPS 的要求。對於透過 HTTP 發出的任何請求，Amazon S3 會以 HTTP 重新導向自動回應，以將請求升級為 HTTPS。
- 存取點不支援匿名存取。
- 跨帳戶存取點不會授予您存取資料的權限，直到您獲授予儲存貯體擁有者的許可。儲存貯體擁有者一律會保留資料的最終存取控制權，並且必須更新儲存貯體政策，才能授權來自跨帳戶存取點的請求。若要檢視儲存貯體政策範例，請參閱 配置使用存取點的 IAM 原則。
- 在 Amazon S3 主控台中檢視跨帳戶存取點時，存取欄會顯示未知。Amazon S3 主控台無法判斷相關聯儲存貯體和物件是否具有公開存取權。除非您需要特定使用案例的公開組態，否則建議您和儲存貯體擁有者封鎖存取點和儲存貯體的所有公開存取權。如需詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

Amazon S3 中的多區域存取點

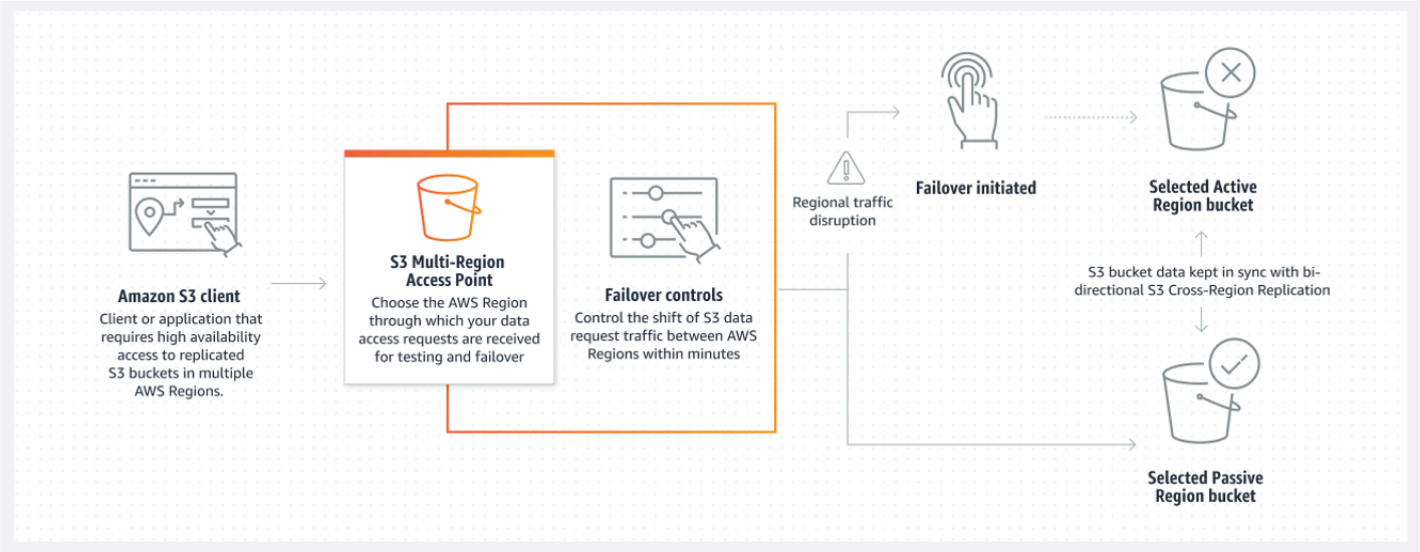
Amazon S3 多區域存取點可提供全域端點，其中應用程式可用來滿足來自多個 AWS 區域的 S3 儲存貯體的請求。您可以使用多區域存取點，進而使用與單一區域中使用的相同架構來建置多區域應用程式，然後在全球任何地方執行這些應用程式。多區域存取點不會透過擁塞的公有網際網路傳送請求，而是提供內建的網路恢復能力，並加速對 Amazon S3 的網際網路請求。對多區域存取點全球端點發出的應用程式請求，可用AWS Global Accelerator來透過 AWS 全球網路自動路由到具有作用中路由狀態的最近 S3 儲存貯體。

建立多區域存取點時，您可以指定一組 AWS 區域 要透過該多區域存取點提供資料的儲存位置。您可以使用 S3 跨區域複寫 (CRR) 來同步這些區域中的儲存貯體間的資料。然後，您可以透過多區域存取點全域端點請求或寫入資料。Amazon S3 會自動將請求從最近的可用區域提供給複寫的資料集。多區域存取點也會與在 Amazon Virtual Private Cloud (VPC) 中執行的應用程式相容，包括使用 AWS PrivateLink 適用於 Amazon S3 的應用程式。

下圖是主動-主動組態中 Amazon S3 多區域存取點的圖形表示法。此圖顯示 Amazon S3 請求如何自動路由到最接近的主動 AWS 區域中的儲存貯體。



下圖是主動-被動組態中 Amazon S3 多區域存取點的圖形表示法。此圖顯示如何控制 Amazon S3 資料存取流量，在主動和被動 AWS 區域之間進行容錯移轉。



若要了解如何使用多區域存取點，請參閱教學課程：Amazon S3 多區域存取點入門指南。

主題

- 建立多區域存取點
- 設定多區域存取點以搭配 AWS PrivateLink 使用
- 透過多區域存取點提出請求

建立多區域存取點

若要在 Amazon S3 中建立多區域存取點，請執行下列動作：

- 指定多區域存取點的名稱。
- 在您要為多區域存取點提供請求的每個 AWS 區域 值區中選擇一個值區。
- 設定多區域存取點的 Amazon S3 封鎖公開存取設定。

您可以在建立請求中提供所有資訊，而 Amazon S3 會以非同步方式對此進行處理。Amazon S3 提供了一個字符，您可以使用該字符來監控非同步建立請求的狀態。

請務必解決安全性警告、錯誤、一般警告，以及 AWS Identity and Access Management Access Analyzer 建議，然後再儲存政策。IAM Access Analyzer 會比對 IAM 政策文法和最佳實務來執行政策檢查，以驗證您的政策。這些檢查會產生問題清單並提供可行的建議，協助您撰寫具有功能性且符合安全最佳實務的政策。若要進一步了解如何使用 IAM Access Analyzer 驗證政策，請參閱《IAM 使用者

指南》中的 IAM Access Analyzer 政策驗證。若要檢視 IAM Access Analyzer 傳回的警告、錯誤和建議清單，請參閱 IAM Access Analyzer 政策檢查參考。

當您使用 API 時，建立多區域存取點的請求是非同步的。當您提交建立多區域存取點的請求時，Amazon S3 會同步授權該請求。然後，它會立即傳回您可以用來追蹤建立請求進度的字符。如需追蹤非同步請求，以建立及管理多區域存取點的詳細資訊，請參閱「搭配支援的 API 操作使用多區域存取點」。

建立多區域存取點後，您可以為其建立存取控制政策。每個多區域存取點都可以設定相關政策。多區域存取點政策是一種以資源為基礎的政策，您可以依資源、使用者或其他條件限制對多區域存取點的使用。

> **Note**
>
> 若要讓應用程式或使用者能夠透過多區域存取點存取物件，下列兩個政策都必須允許該請求：
>
> - 多區域存取點的存取政策。
> - 包含物件之基礎儲存貯體的存取政策
>
> 當兩個政策不同時，更嚴格的政策優先。
> 若要簡化多區域存取點的許可管理，您可以將存取控制從儲存貯體委派給多區域存取點。如需詳細資訊，請參閱 the section called “多區域存取點政策範例”。

透過現有儲存貯體名稱或 Amazon Resource Name (ARN) 存取儲存貯體時，搭配多區域存取點使用儲存貯體不會變更儲存貯體的行為。針對儲存貯體的所有現有操作會繼續像以前一樣運作。您在多區域存取點政策中包含的限制僅適用透過該多區域存取點進行的請求。

您可以在建立多區域存取點之後更新其政策，但您無法刪除政策。但是，您可以更新多區域存取點政策，拒絕所有許可。

主題

- 命名 Amazon S3 多區域存取點的規則
- 為 Amazon S3 多區域存取點選擇儲存貯體的規則
- 建立 Amazon S3 多區域存取點
- 使用 Amazon S3 多區域存取點封鎖公開存取
- 檢視 Amazon S3 多區域存取點組態詳細資訊

- 刪除多區域存取點

命名 Amazon S3 多區域存取點的規則

當您建立多區域存取點時，您可以為其命名，該名稱是您選擇的字串。建立多區域存取點之後，您無法變更其名稱。名稱對於您的 AWS 帳戶必須是唯一的，並且其必須符合 多區域存取點約束與限制 中列出的命名要求。為了協助您識別多區域存取點，請使用對您、您的組織有意義或是可以反映案例的名稱。

您可以在叫用多區域存取點管理操作時使用此名稱，例如 `GetMultiRegionAccessPoint` 和 `PutMultiRegionAccessPointPolicy`。此名稱不會用來向多區域存取點傳送請求，而且不需要向使用多區域存取點提出請求的用戶端公開。

當 Amazon S3 建立多區域存取點時，它會自動為其指派別名。此別名是一個以 `.mrap` 結尾的唯一英數字元字串。別名用於建構多區域存取點的主機名稱和 Amazon 資源名稱 (ARN)。完整名稱也是以多區域存取點的別名為基礎。

您無法從其別名判斷多區域存取點的名稱，因此您可以揭露別名，而不會暴露多區域存取點的名稱、用途或擁有者的風險。Amazon S3 會為每個新的多區域存取點選取別名，且別名無法變更。如需有關定址多區域存取點的詳細資訊，請參閱「透過多區域存取點提出請求」。

多區域存取點別名在一段時間內都是唯一的，且並非以多區域存取點的名稱或組態為基礎。如果您建立多區域存取點，然後將其刪除並建立另一個具有相同名稱和組態的存取點，則第二個多區域存取點的別名與第一個不同。新的多區域存取點絕對不會與先前的多區域存取點具有相同的別名。

為 Amazon S3 多區域存取點選擇儲存貯體的規則

每個多區域存取點都會關聯至您要滿足請求的區域。多區域存取點必須與每個區域中的一個儲存貯體關聯。您可以指定請求中每個儲存貯體的名稱，以建立多區域存取點。支援多區域存取點的儲存貯體可以與擁有多區域存取點 AWS 帳戶 的值區相同，也可以位於其他存取點。 AWS 帳戶

單一儲存貯體可供多個多區域存取點使用。

> **Important**
>
> - 您只能在建立多區域存取點時，指定與其關聯的儲存貯體。將其建立之後，您就無法從多區域存取點組態新增、修改或刪除儲存貯體。若要變更儲存貯體，您必須刪除整個多區域存取點並新建一個。

- 您無法刪除屬於多區域存取點的儲存貯體。如果您要刪除連接至多重區域存取點的儲存貯體，請先刪除多重區域存取點。
- 如果您將其他帳戶擁有的儲存貯體新增至您的多區域存取點，該儲存貯體擁有者也必須更新其儲存貯體政策，將存取許可授予多區域存取點。否則，多區域存取點無法從該儲存貯體擷取資料。如需範例政策，示範如何授予這類存取，請參閱多區域存取點政策範例。
- 並非所有區域都支援多區域存取點。如需支援區域的清單，請參閱 多區域存取點約束與限制。

您可以建立複寫規則，以同步儲存貯體之間的資料。這些規則可讓您自動將資料從來源儲存貯體複製到目的地儲存貯體。將儲存貯體連接至多區域存取點並不會影響複寫的運作方式。在稍後的章節中，將會介紹使用多區域存取點設定複寫。

Important

當您對多區域存取點提出請求時，多區域存取點不會知道多區域存取點中儲存貯體的資料內容。因此，取得請求的儲存貯體可能不包含請求的資料。若要在 Amazon S3 儲存貯體中建立與多區域存取點相關聯的資料集，建議您設定 S3 跨區域複寫 (CRR)。如需詳細資訊，請參閱設定複寫以搭配多區域存取點使用。

建立 Amazon S3 多區域存取點

下列範例示範如何使用 Amazon S3 主控台建立多區域存取點。

使用 S3 主控台

若要建立多區域存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左導覽窗格中，選擇 Multi-Region Access Points (多區域存取點)。
3. 選擇建立多區域存取點，開始建立多區域存取點。
4. 多區域存取點頁面上，在多區域存取點名稱欄位中，提供多區域存取點的名稱。
5. 選取要與此多區域存取點建立關聯的儲存貯體。您可以選擇帳戶中的儲存貯體，也可以從其他帳戶中選擇儲存貯體。

> **Note**
>
> 不論從您的帳戶或其他帳戶，您至少須新增一個儲存貯體。此外，請注意，多區域存取點每個 AWS 區域僅支援一個儲存貯體。因此，您無法從相同區域新增兩個儲存貯體。根據預設停用的AWS 區域 不受支援。

- 若要新增您帳戶中的儲存貯體，請選擇新增儲存貯體。系統會顯示您帳戶中的所有儲存貯體清單。您可以依名稱搜尋儲存貯體，或依字母順序排序儲存貯體名稱。
- 若要從其他帳戶新增儲存貯體，請選擇從其他帳戶新增儲存貯體。請確定您知道確切的值區名稱和 AWS 帳戶 ID，因為您無法搜尋或瀏覽其他帳戶中的值區。

> **Note**
>
> 您必須輸入有效的 AWS 帳戶 ID 和值區名稱。儲存貯體也須位於受支援的區域中，否則當您嘗試建立多區域存取點時會遇到錯誤訊息。如需支援多區域存取點的區域清單，請參閱多區域存取點限制。

6. (選用) 若您要移除已新增的儲存貯體，請選擇移除。

> **Note**
>
> 建立多區域存取點後，就無法在此多區域存取點新增或移除儲存貯體。

7. 在 Block Public Access settings for this Access Point (此多區域存取點的封鎖公開存取權限設定) 下，選取要套用至該多區域存取點的封鎖公開存取設定。依預設，新的多區域存取點的所有封鎖公開存取設定都會啟用。我們建議您啟用所有設定，除非您知道您有特別需要停用任一設定。

> **Note**
>
> 在建立多區域存取點的封鎖公開存取設定後，您便無法再變更設定。因此，如果您要封鎖公開存取，請確定您的應用程式在沒有公開存取的情況下正常運作，然後再建立多區域存取點。

8. 選擇 Create Multi-Region Access Point (建立多區域存取點)。

> **Important**
>
> 當您將其他帳戶擁有的儲存貯體新增至您的多區域存取點，該儲存貯體擁有者也必須更新其儲存貯體政策，將存取許可授予多區域存取點。否則，多區域存取點無法從該儲存貯體擷取資料。如需範例政策，示範如何授予這類存取，請參閱多區域存取點政策範例。

使用 AWS CLI

您可以使用 AWS CLI 建立多區域存取點。當您建立多區域存取點時，您必須提供其將支援的所有儲存貯體。建立多區域存取點之後，您就無法將儲存貯體新增至該存取點。

下列範例示範使用 AWS CLI建立了含有兩個儲存貯體的多區域存取點。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control create-multi-region-access-point --account-id 111122223333 --details '{
        "Name": "simple-multiregionaccesspoint-with-two-regions",
        "PublicAccessBlock": {
            "BlockPublicAcls": true,
            "IgnorePublicAcls": true,
            "BlockPublicPolicy": true,
            "RestrictPublicBuckets": true
        },
        "Regions": [
            { "Bucket": "example-s3-bucket1" },
            { "Bucket": "example-s3-bucket2" }
        ]
    }' --region us-west-2
```

使用 Amazon S3 多區域存取點封鎖公開存取

每個區域存取點都有不同的 Amazon S3 封鎖公開存取設定。這些設定會與擁有多區域存取點和基礎值區的「封鎖公用存取」設定搭配運作。 AWS 帳戶

當 Amazon S3 授權請求時，它會套用這些設定的限制性最高的組合。如果任何這些資源 (多區域存取點擁有者帳戶、基礎儲存貯體或儲存貯體擁有者帳戶) 的封鎖公開存取設定封鎖了所請求動作或資源的存取，Amazon S3 會拒絕該請求。

我們建議您啟用所有封鎖公開存取設定，除非您有特別需要停用任一設定。依預設，多區域存取點的所有封鎖公開存取設定都會啟用。如果啟用封鎖公開存取，多區域存取點將無法接受網際網路請求。

> **Important**
>
> 在建立多區域存取點的封鎖公開存取設定後，您便無法再變更設定。

如需 Amazon S3 封鎖公開存取的詳細資訊，請參閱「封鎖對 Amazon S3 儲存體的公開存取權」。

檢視 Amazon S3 多區域存取點組態詳細資訊

下列範例示範如何使用 Amazon S3 主控台檢視多區域存取點組態詳細資訊。

使用 S3 主控台

若要建立多區域存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左導覽窗格中，選擇 Multi-Region Access Points (多區域存取點)。
3. 選擇您要檢視其組態詳細資訊的多區域存取點名稱。

   - 屬性標籤會列出與您的多區域存取點相關聯的所有儲存貯體、建立日期、Amazon Resource Name (ARN) 和別名。 AWS 帳戶 ID 欄也會列出與您的多區域存取點相關聯之外部帳戶所擁有的任何儲存貯體。
   - 許可標籤會列出封鎖公開存取設定，這些設定套用至與此多區域存取點相關聯的儲存貯體。您也可以檢視多區域存取點的政策 (若您已建立)。許可頁面上的資訊警示也會列出已啟用封鎖公開存取設定之多區域存取點的所有儲存貯體 (不論在您的帳戶或其他帳戶)。
   - 複寫和容錯移轉標籤則提供映射檢視，其中有與多區域存取點相關聯的儲存貯體，以及儲存貯體所在區域。若有來自其他帳戶的儲存貯體，且您無權從中提取資料，則複寫摘要映射的區域會以紅色標記，表示 AWS 區域 取得複寫狀態時發生錯誤。

     > **Note**
     >
     > 若要從外部帳戶中的儲存貯體擷取複寫狀態資訊，儲存貯體擁有者必須在其儲存貯體政策中授予您 `s3:GetBucketReplication` 權限。

     此標籤也會針對與您的多區域存取點搭配使用的區域，提供複寫指標、複寫規則和容錯移轉狀態。

使用 AWS CLI

您可以使用 AWS CLI 檢視多區域存取點的組態詳細資料。

下列 AWS CLI 範例會取得您目前的多區域存取點設定。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control get-multi-region-access-point --account-id 111122223333 --name example-s3-bucket1
```

刪除多區域存取點

下列程序說明如何使用 Amazon S3 主控台刪除多區域存取點。

刪除多區域存取點時，不會刪除與多區域存取點相關聯的儲存貯體，僅會刪除多區域存取點本身。

使用 S3 主控台

刪除多區域存取點

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左導覽窗格中，選擇 Multi-Region Access Points (多區域存取點)。
3. 選取多區域存取點名稱旁的選項按鈕。
4. 選擇刪除。
5. 在「刪除多區域存取點」對話方塊中，輸入要刪除的 AWS 值區名稱。

   > **Note**
   >
   > 請務必輸入有效的儲存貯體名稱。否則，刪除按鈕將遭停用。

6. 選擇刪除，確認刪除多區域存取點。

使用 AWS CLI

您可以使用刪 AWS CLI 除多區域存取點。此動作不會刪除與多區域存取點相關聯的儲存貯體，僅會刪除多區域存取點本身。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control delete-multi-region-access-point --account-id 123456789012 --details
 Name=example-multi-region-access-point-name
```

設定多區域存取點以搭配 AWS PrivateLink 使用

您可以使用多區域存取點，來路由 AWS 區域 之間的 Amazon S3 請求流量。每個多區域存取點全域端點會從多個來源路由 Amazon S3 資料請求流量，而無需使用單一端點建立複雜的聯網組態。這些資料請求流量來源包括：

- 來自虛擬私有雲端 (VPC) 的流量
- 來自內部部署資料中心通過 AWS PrivateLink 的流量
- 來自公有網際網路的流量

如果您建立與 S3 多區域存取點的 AWS PrivateLink 連線，您可以使用簡單的網路架構和組態，透過私有連線將 S3 請求路由到 AWS 或跨多個 AWS 區域。使用 AWS PrivateLink 時，不需要設定 VPC 對等互連。

主題

設定多區域存取點以搭配 AWS PrivateLink 使用

AWS PrivateLink 使用 Virtual Private Cloud (VPC) 中的私有 IP 地址為您提供與 Amazon S3 的私有連線。您可以在 VPC 內佈建一個或多個介面端點，以連接到 Amazon S3 多區域存取點。

您可以透過 AWS Management Console、AWS CLI 或 AWS 開發套件建立多區域存取點的 com.amazonaws.s3-global.accesspoint 端點。若要進一步了解如何設定多區域存取點的界面端點，請參閱《VPC 使用者指南》中的介面 VPC 端點。

若要透過界面端點向多區域存取點提出要求，請依照下列步驟設定 VPC 和多區域存取點。

若要設定多區域存取點以搭配 AWS PrivateLink 使用

1. 建立或擁有可連線至多區域存取點的適當 VPC 端點。如需建立 VPC 端點的詳細資訊，請參閱 VPC 使用者指南中的《介面 VPC 端點》。

> **Important**
>
> 務必建立 com.amazonaws.s3-global.accesspoint 端點。其他端點類型無法存取多區域存取點。

建立此 VPC 端點之後，VPC 中的所有多區域存取點請求都會透過此端點路由 (如果您已為端點啟用私有 DNS)。其預設為啟用。

2. 如果多區域存取點政策不支援來自 VPC 端點的連線，您將需要更新它。
3. 確認個別儲存貯體政策將允許存取多區域存取點的使用者。

請記住，多區域存取點的運作方式是將請求路由至儲存貯體，而不是自行履行請求。請務必記住這一點，因為請求的建立者必須具有多區域存取點的許可，並允許存取多區域存取點中的個別儲存貯體。否則，請求可能會被路由到建立者沒有許可滿足請求的儲存貯體。多區域存取點和相關聯的儲存貯體可以由相同或不同的 AWS 帳戶擁有。不過，如果正確設定許可，則來自不同帳戶的 VPC 可以使用多區域存取點。

因此，VPC 端點政策必須允許存取多區域存取點，以及您希望能夠滿足請求的每個基礎儲存貯體。例如，假設您具有別名為 `mfzwi23gnjvgw.mrap` 的多區域存取點。它是由儲存貯體 `DOC-EXAMPLE-BUCKET1` 和 `DOC-EXAMPLE-BUCKET2` 提供支援，並且全部都由 AWS 帳戶 `123456789012` 擁有。在這種情況下，下列 VPC 端點政策會允許 `GetObject` 從 VPC 向 `mfzwi23gnjvgw.mrap` 提出的請求由任一後備儲存貯體來實現。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Sid": "Read-buckets-and-MRAP-VPCE-policy",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*",
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET2/*",
            "arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap/object/*"
        ]
```

```
    }]
}
```

如先前所述，您也必須確保多區域存取點政策已設定為可透過 VPC 端點支援存取。您不需要指定請求存取的 VPC 端點。下列範例政策會向嘗試將多區域存取點用於 GetObject 請求的任何申請者授予存取權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Sid": "Open-read-MRAP-policy",
        "Effect": "Allow",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
          ],
        "Resource": "arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap/object/*"
    }]
}
```

當然，各個儲存貯體都需要一個政策來支援透過 VPC 端點提交的請求的存取。下列範例政策會授予任一匿名使用者的讀取存取權限，其中包含透過 VPC 端點發出的請求。

```
{
    "Version":"2012-10-17",
   "Statement": [
   {
       "Sid": "Public-read",
       "Effect":"Allow",
       "Principal": "*",
       "Action": "s3:GetObject",
       "Resource": [
           "arn:aws:s3:::DOC-EXAMPLE-BUCKET1",
           "arn:aws:s3:::DOC-EXAMPLE-BUCKET2/*"]
    }]
}
```

如需有關編輯 VPC 端點政策的詳細資訊，請參閱《VPC 使用者指南》中的使用 VPC 端點控制對服務的存取。

從 VPC 端點刪除對多區域存取點的存取

如果您擁有多區域存取點，並想要從介面端點刪除對其的存取，則您必須為多區域存取點提供新的存取政策，以防止透過 VPC 端點存取的請求存取。但是，如果多區域存取點中的儲存貯體支援透過 VPC 端點的請求，則它們將繼續支援這些請求。如果您想防止該支援，則您還必須更新儲存貯體的政策。為多區域存取點提供新的存取政策，僅會防止對多區域存取點的存取，不會防止對基礎儲存貯體的存取。

> **Note**
>
> 您無法刪除多區域存取點的存取政策。若要刪除對多區域存取點的存取，您必須提供新的存取政策以及您想要的已修改存取。

您可以更新儲存貯體政策，防止透過 VPC 端點的請求，而非更新多區域存取點的存取政策。在這種情況下，使用者仍然可以透過 VPC 端點存取多區域存取點。但是，如果多區域存取點請求被路由到儲存貯體政策阻止存取的儲存貯體，則該請求會產生錯誤訊息。

透過多區域存取點提出請求

類似於其他資源，Amazon S3 多區域存取點具有 Amazon Resource Name (ARN)。您可以使用 AWS Command Line Interface (AWS CLI)、AWS SDK 或 Amazon S3 API，使用這些 ARN，將請求導向至多區域存取點。您也可以使用這些 ARN 來識別存取控制政策中的多區域存取點。多區域存取點 ARN 不會包含或揭露多區域存取點的名稱。如需 ARN 的詳細資訊，請參閱《AWS 一般參考》中的 Amazon Resource Name (ARN)。

> **Note**
>
> 多區域存取點別名和 ARN 無法互換使用。

多區域存取點 ARN 會使用下列格式：

`arn:aws:s3::`*`account-id`*`:accesspoint/`*`MultiRegionAccessPoint_alias`*

以下是多區域存取點 ARN 的幾個範例：

- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap` 代表由 AWS 帳戶 `123456789012` 擁有且具有別名 `mfzwi23gnjvgw.mrap` 的多區域存取點。

- arn:aws:s3::123456789012:accesspoint/* 代表帳戶 123456789012 下的所有多區域存取點。此 ARN 與帳戶 123456789012 的所有多區域存取點相符，但不符合任何區域 Amazon S3 Access Points，因為 ARN 不包含 AWS 區域。此 ARN arn:aws:s3:us-west-2:123456789012:accesspoint/* 與帳戶 123456789012 的區域 us-west-2 中的所有區域 Amazon S3 Access Points，但不符合任何多區域存取點。

透過多區域存取點存取的物件的 ARN 會使用下列格式。

arn:aws:s3::*account_id*:accesspoint/*MultiRegionAccessPoint_alias*//*key*

與多區域存取點 ARN 一樣，透過多區域存取點存取的物件的 ARN 不包含 AWS 區域。請見下方範例。

- arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//-01 代表由帳戶 123456789012 擁有且透過具有別名 mfzwi23gnjvgw.mrap 的多區域存取點存取的 -01。
- arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//* 代表透過帳戶 123456789012 中具有別名 mfzwi23gnjvgw.mrap 的多區域存取點存取的所有物件。
- arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//-01/finance/* 代表為帳戶 123456789012 中具有別名 mfzwi23gnjvgw.mrap 的多區域存取點而在 -01/finance/ 下存取的所有物件。

多區域存取點主機名稱

您可以使用多區域存取點的主機名稱，透過多區域存取點存取 Amazon S3 中的資料。請求可以從公有網際網路導向至這個主機名稱。如果您已針對多區域存取點設定一或多個網際網路閘道，則請求也可以從公有網際網路或虛擬私有雲端 (VPC)導向至這個主機名稱。如需建立 VPC 介面端點以搭配使用多區域存取點的詳細資訊，請參閱 設定多區域存取點以搭配 AWS PrivateLink 使用。

若要使用 VPC 端點，從 VPC 透過多區域存取點提出請求，您可以使用 AWS PrivateLink。當您使用 AWS PrivateLink 對多區域存取點提出請求時，您無法直接使用以 *region*.vpce.amazonaws.com 結尾的端點特定區域 網域名稱系統 (DNS) 名稱。此主機名稱不會具有與其相關聯的憑證，因此無法直接使用它。您仍然可以使用 VPC 端點的公有 網域名稱系統 (DNS) 名稱作為 CNAME 或 ALIAS 目標。或者，您可以在端點上啟用私有 網域名稱系統 (DNS)，並使用標準的多區域存取點 *MultiRegionAccessPoint_alias*.accesspoint.s3-global.amazonaws.com 網域名稱系統 (DNS) 名稱，如本節所述。

當您透過多區域存取點，對 API 進行 Amazon S3 資料操作請求時 (例如，GetObject)，請求的主機名稱如下。

MultiRegionAccessPoint_alias.accesspoint.s3-global.amazonaws.com

例如，若要透過具有別名 mfzwi23gnjvgw.mrap 的多區域存取點提出 GetObject 請求，請對主機名稱 mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com 提出請求。主機名稱的 s3-global 部分指出此主機名稱不適用於特定區域。

透過多區域存取點提出請求類似於透過單一區域存取點提出請求。不過，務必要注意以下差異：

- 多區域存取點 ARN 不包含 AWS 區域。他們遵循格式 arn:aws:s3::*account-id*:accesspoint/*MultiRegionAccessPoint_alias*。
- 對於透過 API 操作提出的請求 (這些請求不需要使用 ARN)，多區域存取點會使用不同的端點結構描述。結構描述是 *MultiRegionAccessPoint_alias*.accesspoint.s3-global.amazonaws.com – 例如 mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com。請注意與單一區域存取點相比的差異：
  - 多區域存取點主機名稱使用其別名，而非多區域存取點名稱。
  - 多區域存取點主機名稱不包含擁有者的 AWS 帳戶 ID。
  - 多區域存取點主機名稱不包含 AWS 區域。
  - 多區域存取點主機名稱包含 s3-global.amazonaws.com 而非 s3.amazonaws.com。
- 多區域存取點請求必須使用簽章版本 4A (SigV4A) 進行簽署。當您使用 AWS 開發套件時，SDK 會自動將 SigV4 轉換為 SigV4A。因此，請確認您的 AWS SDK 支援 SigV4a 作為簽署實作，用來簽署全域 AWS 區域 請求。如需 SigV4A 的詳細資訊，請參閱《AWS 一般參考》中的簽署 AWS API 請求。

多區域存取點和 Amazon S3 Transfer Acceleration

Amazon S3 Transfer Acceleration 是一種可快速將資料傳輸至儲存貯體的功能。Transfer Acceleration 是在個別儲存貯體層級上設定的。如需 Transfer Acceleration 的詳細資訊，請參閱 使用 Amazon S3 Transfer Acceleration 設定快速安全的檔案傳輸。

多區域存取點會使用與 Transfer Acceleration 類似的加速傳輸機制，以透過 AWS 網路傳送大型物件。因此，當透過多區域存取點傳送請求時，您不需要使用 Transfer Acceleration。這項提升的傳輸效能會自動納入多區域存取點。

主題

- 許可
- 多區域存取點約束與限制

- 多區域存取點請求路由
- Amazon S3 多區域存取點容錯移轉控制
- 設定複寫以搭配多區域存取點使用
- 搭配支援的 API 操作使用多區域存取點
- 監控與記錄透過多區域存取點對基礎資源提出的請求

許可

Amazon S3 多區域存取點可以簡化多個 AWS 區域中 Amazon S3 儲存貯體的資料存取。多區域存取點是具名全域端點，您可以用來執行 Amazon S3 資料存取物件操作，例如 `GetObject` 和 `PutObject`。每個多區域存取點都可對透過全域端點提出的任何請求具有不同的許可和網路控制。

每個多區域存取點也可以強制執行自訂的存取政策，該政策可結合附加至基礎儲存貯體的儲存貯體政策運作。請求若要成功，下列政策都必須允許操作：

- 多區域存取點政策
- 基礎 AWS Identity and Access Management (IAM) 政策
- 基礎儲存貯體政策 (請求的路由位置)

您可以將任何多區域存取點政策設定為僅接受來自特定 IAM 使用者或群組的請求。如需如何執行此操作的範例，請參閱 the section called “多區域存取點政策範例” 中的範例 2。若要限制只能透過私有網絡存取 Amazon S3 資料，您可以將多區域存取點政策設定為僅接受來自虛擬私有雲端 (VPC) 的請求。

例如，假設您使用 AWS 帳戶中名為 `AppDataReader` 的使用者，透過多區域存取點提出 `GetObject` 請求。為了協助確保請求不會遭到拒絕，`AppDataReader` 使用者必須透過多區域存取點及多區域存取點下的每個儲存貯體獲授予 `s3:GetObject` 許可。`AppDataReader` 將無法從任何未授予此許可的儲存貯體中擷取資料。

> **Important**
>
> 透過儲存貯體名稱或 Amazon Resource Name (ARN) 直接存取儲存貯體時，將儲存貯體的存取控制委派給多區域存取點政策不會變更儲存貯體的行為。直接針對儲存貯體進行的所有操作將繼續像以前一樣運作。您在多區域存取點政策中包含的限制僅適用透過該多區域存取點提出的請求。

管理多區域存取點的公開存取

多區域存取點對每個多區域存取點支援獨立的封鎖公開存取設定。建立多區域存取點時，您可以指定適用該多區域存取點的封鎖公開存取設定。

> **Note**
>
> 即使多區域存取點的獨立封鎖公開存取設定已停用，在此帳戶的封鎖公開存取設定 (您的帳戶) 或外部儲存貯體的封鎖公開設定下啟用的任何封鎖公開存取設定仍然適用。

對於透過多區域存取點提出的任何請求，Amazon S3 會評估下列項目的封鎖公開存取設定：

- 多區域存取點
- 基礎儲存貯體 (包括外部儲存貯體)
- 擁有多區域存取點的帳戶
- 擁有基礎儲存貯體 (包括外部帳戶) 的帳戶

如果這些設定中有任一指出應該封鎖請求，則 Amazon S3 會拒絕請求。如需 Amazon S3 封鎖公開存取功能的詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

> **Important**
>
> 依預設，多區域存取點的所有封鎖公開存取設定都會啟用。您必須明確停用您不想套用至多區域存取點的任何設定。
> 在建立多區域存取點的封鎖公開存取設定後，您便無法再變更設定。

檢視多區域存取點的封鎖公開存取設定

檢視多區域存取點的封鎖公開存取設定

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Multi-Region Access Points (多區域存取點)。
3. 選擇您要檢閱的多區域存取點名稱。
4. 選擇 許可 標籤。

5. 在 Block Public Access settings for this Access Point (此多區域存取點的封鎖公開存取設定) 下，檢閱多區域存取點的封鎖公開存取設定。

> **Note**
>
> 在建立多區域存取點後，您無法編輯封鎖公開存取設定。因此，如果您要封鎖公開存取，請確定您的應用程式在沒有公開存取的情況下正常運作，然後再建立多區域存取點。

使用多區域存取點政策

下列範例多區域存取點政策會將存取權授予 IAM 使用者，以從您的多區域存取點列出和下載檔案。若要使用此範例政策，請以您自己的資訊取代 *user input placeholders*。

```
 {
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::123456789012:user/JohnDoe"
         },
         "Action":[
            "s3:ListBucket",
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias",
            "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias/object/
*"
         ]
      }
   ]
}
```

若要使用 AWS Command Line Interface (AWS CLI)，將您的多區域存取點政策與指定的多區域存取點建立關聯，請使用下列 `put-multi-region-access-point-policy` 命令。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。每個多區域存取點只能有一個政策，因此對 `put-multi-region-access-point-policy` 動作提出的請求會取代與所指定多區域存取點相關聯的任何現有政策。

AWS CLI

```
aws s3control put-multi-region-access-point-policy
--account-id 111122223333
--details { "Name": "DOC-EXAMPLE-BUCKET-MultiRegionAccessPoint",
 "Policy": "{ \"Version\": \"2012-10-17\", \"Statement\": { \"Effect\":
 \"Allow\", \"Principal\": { \"AWS\": \"arn:aws:iam::111122223333:root
\" }, \"Action\": [\"s3:ListBucket\", \"s3:GetObject\"], \"Resource\":
 [ \"arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias",
 \"arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias/object/*
\" ] } }" }
```

若要查詢先前操作的結果，請使用下列命令：

AWS CLI

```
aws s3control describe-multi-region-access-point-operation
--account-id 111122223333
--request-token-arn requestArn
```

若要擷取您的多區域存取點政策，請使用下列命令：

AWS CLI

```
aws s3control get-multi-region-access-point-policy
--account-id 111122223333
--name=DOC-EXAMPLE-BUCKET-MultiRegionAccessPoint
```

編輯多區域存取點政策

多區域存取點政策 (以 JSON 撰寫) 可讓與此多區域存取點搭配使用的 Amazon S3 儲存貯體存取儲存體。您可以允許或拒絕特定主體對多區域存取點執行各種動作。當透過多區域存取點將請求路由至儲存貯體時，多區域存取點的這兩個存取政策都適用。限制性較高的存取政策一律優先採用。

> **Note**
>
> 如果儲存貯體包含其他帳戶所擁有的物件，則多區域存取點政策不會套用至其他 AWS 帳戶 擁有的物件。

在套用多區域存取點政策之後，無法刪除該政策。您可以編輯政策或建立覆寫現有政策的新政策。

編輯多區域存取點政策

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Multi-Region Access Points (多區域存取點)。
3. 選擇您要編輯其政策的多區域存取點名稱。
4. 選擇 許可 標籤。
5. 向下捲動至 Multi-Region Access Point policy (多區域存取點政策)。選擇 Edit (編輯)來更新政策 (以 JSON 表示)。
6. Edit Multi-Region Access Point policy (編輯多區域存取點政策) 頁面即會出現。您可以直接將政策輸入文字欄位中，也可以選擇 Add statement (新增陳述式)，從下拉式清單中選取政策元素。

> **Note**
>
> 主控台會自動顯示多區域存取點的 Amazon Resource Name (ARN)，您可以在政策中使用該 ARN。例如多區域存取點政策，請見 the section called “多區域存取點政策範例”。

多區域存取點政策範例

Amazon S3 多區域存取點支援 AWS Identity and Access Management (IAM) 資源政策。您可以使用這些政策，依資源、使用者或其他條件控制多區域存取點的使用。若要讓應用程式或使用者能夠透過多區域存取點存取物件，多區域存取點和基礎儲存貯體都必須允許該相同的存取。

若要允許同時對多區域存取點和基礎儲存貯體的存取，請執行下列其中一項操作：

- (建議) 若要在使用 Amazon S3 多區域存取點時簡化存取控制，請將 Amazon S3 儲存貯體的存取控制委派給多區域存取點。如需如何執行此操作的範例，請參閱本節中的範例 1。
- 將多區域存取點政策中的相同許可新增至基礎儲存貯體政策。

> **Important**
>
> 透過儲存貯體名稱或 Amazon Resource Name (ARN) 直接存取儲存貯體時，將儲存貯體的存取控制委派給多區域存取點政策不會變更儲存貯體的行為。直接針對儲存貯體進行的所有操作

將繼續像以前一樣運作。您在多區域存取點政策中包含的限制僅適用透過該多區域存取點提出的請求。

Example 1 – 在儲存貯體政策中，委派存取權給特定多區域存取點 (針對相同帳戶或跨帳戶)

下列範例儲存貯體政策允許完整儲存貯體存取特定多區域存取點。這表示對此儲存貯體的所有存取皆由附加至多區域存取點的政策控制。我們建議您針對不需要直接存取儲存貯體的所有使用案例，以此方式設定儲存貯體。您可以將此儲存貯體政策結構用於同一帳戶或其他帳戶的多區域存取點。

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "*",
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointArn" : "MultiRegionAccessPoint_ARN" }
        }
    }]
}
```

Note

如果您要授予存取權給多個多區域存取點，請務必列出每個多區域存取點。

Example 2 - 在您的多區域存取點政策中將帳戶存取權授予多區域存取點

下列多區域存取點政策允許帳戶 *123456789012* 許可，以列出和讀取由 *MultiRegionAccessPoint_ARN* 定義多區域存取點中包含的物件。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
       "Effect": "Allow",
       "Principal": {
```

```
            "AWS":"arn:aws:iam::123456789012:user/JohnDoe"
         },
         "Action":[
            "s3:ListBucket",
            "s3:GetObject"
         ],
         "Resource":[
            "MultiRegionAccessPoint_ARN",
            "MultiRegionAccessPoint_ARN/object/*"
         ]
      }
   ]
}
```

Example 3 - 允許儲存貯體清單的多區域存取點政策

下列多區域存取點政策允許帳戶 *123456789012* 許可，以列出由 *MultiRegionAccessPoint_ARN* 定義多區域存取點中包含的物件。

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
       "Effect": "Allow",
       "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/JohnDoe"
        },
        "Action": "s3:ListBucket",
        "Resource": "MultiRegionAccessPoint_ARN"
       }
    ]
}
```

多區域存取點約束與限制

Amazon S3 中的多區域存取點具有以下約束與限制：

- 多區域存取點名稱：
  - 在單一 AWS 帳戶中必須是唯一的
  - 必須以數字或小寫字母開頭
  - 長度必須介於 3 與 50 個字元之間

  - 不能以連字號 (-) 開頭和結尾
  - 不能包含底線 (_)、大寫字母或句點 (.)
  - 建立之後，就無法再度編輯。
- 多區域存取點別名是由 Amazon S3 產生的，無法編輯或重複使用。
- 您無法使用閘道端點透過多區域存取點存取資料。但您可以使用介面端點，透過多區域存取點存取資料。若要使用 AWS PrivateLink，您必須建立 VPC 端點。如需詳細資訊，請參閱 設定多區域存取點以搭配 AWS PrivateLink 使用。
- 若要將多區域存取點與 Amazon 搭配使用 CloudFront，您必須將多區域存取點設定為`Custom Origin`分發類型。如需有關各種原點類型的詳細資訊，請參閱搭 CloudFront 配分佈使用各種原點。如需將多區域存取點與 Amazon 搭配使用的詳細資訊 CloudFront，請參閱 Storage 部落格上的跨多個區域建置主動-主動式、接近型應用程式。AWS
- 多區域存取點最低需求：
  - Transport Layer Security (TLS) v1.2
  - 簽章版本 4 (SigV4A)

    多區域存取點支援簽章版本 4A。這個版本的 SigV4 允許為多個 AWS 區域簽署請求。此功能對於可能導致來自數個區域之一的資料存取的 API 操作非常有用。使用 AWS SDK 時，您需要提供認證，而對多區域存取點的請求將使用簽名版本 4A，而無需額外設定。請務必檢查您的 AWS SDK 與 SigV4a 演算法的相容性。如需有關 SigV4a 的詳細資訊，請 AWS參閱. AWS 一般參考

    > **Note**
    >
    > 若要將 SigV4a 與臨時安全登入資料搭配使用 (例如，使用 AWS Identity and Access Management (IAM) 角色時，您可以從區域 () 端點要求臨時登入資料。 AWS Security Token Service AWS STS如果您從全局 AWS STS 端點 (`sts.amazonaws.com`) 請求臨時憑據，則必須首先將會話令牌的區域兼容性設置為全局端點有效 AWS 區域。如需詳細資訊，請參閱 AWS STS《IAM 使用者指南》 AWS 區域中的「管理」。
- 多區域存取點不支援匿名請求。
- 多區域存取點限制：
  - 不支援 IPv6。
  - 不支援 Amazon S3 on Outposts 儲存貯體。
  - 使用「多區域存取點」ARN 時，「多區域存取點」僅支援使用「多區域存取點」作為目的地的複製作業。

  - 不支援 S3 批次操作功能。
- 某些 AWS SDK 不受支援。若要確認多區域存取點支援哪些 AWS SDK，請參閱與 AWS SDK 的相容性。
- 多區域存取點的服務配額如下：
  - 每個帳戶最多可有 100 個多區域存取點。
  - 單一多區域存取點的限制為 17 個區域。
- 建立多區域存取點之後，您就無法從多區域存取點組態新增、修改或刪除儲存貯體。若要變更儲存貯體，您必須刪除整個多區域存取點並新建一個。如果刪除多區域存取點中的跨帳戶儲存貯體，重新連接此儲存貯體的唯一方法是，使用該帳戶中相同的名稱與區域，重新建立儲存貯體。
- 只有在刪除多區域存取點之後，才能刪除多區域存取點中使用的基礎儲存貯體 (位在相同帳戶中)。
- 所有建立或維護多區域存取點的控制平面請求，都必須路由至 US West (Oregon) 區域。對於多區域存取點資料平面請求，不需要指定區域。
- 對於多區域存取點容錯移轉控制平面，請求必須路由至這五個受支援區域的其中一個。
  - US East (N. Virginia)
  - US West (Oregon)
  - Asia Pacific (Sydney)
  - Asia Pacific (Tokyo)
  - Europe (Ireland)
- 您的多區域存取點僅支援下 AWS 區域列值區：
  - US East (N. Virginia)
  - US East (Ohio)
  - US West (N. California)
  - US West (Oregon)
  - Asia Pacific (Mumbai)
  - Asia Pacific (Osaka)
  - Asia Pacific (Seoul)
  - Asia Pacific (Singapore)
  - Asia Pacific (Sydney)
  - Asia Pacific (Tokyo)
  - Canada (Central)
  - Europe (Frankfurt)

- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)
- South America (São Paulo)

多區域存取點請求路由

當您透過多區域存取點提出請求時，Amazon S3 會決定與多區域存取點相關聯的儲存貯體中哪些儲存貯體最接近您。然後，Amazon S3 會將請求導向該儲存貯體，而不管其所在的 AWS 區域。

在多區域存取點將請求路由到最接近的儲存貯體之後，Amazon S3 處理請求的操作將與您直接向儲存貯體提出請求一樣。多區域存取點不知道 Amazon S3 儲存貯體的資料內容。因此，取得請求的儲存貯體可能不包含請求的資料。若要在 Amazon S3 儲存貯體中建立與多區域存取點相關聯的資料集，您可以設定 S3 跨區域複寫 (CRR)。然後，任何儲存貯體都可以成功履行 請求。

Amazon S3 會根據下列規則指示多區域存取點請求：

- Amazon S3 根據鄰近性最佳化要滿足的請求。其可查看多區域存取點支援的儲存貯體，並將請求轉送到最接近的儲存貯體。
- 如果請求指定現有資源 (例如 `GetObject`)，Amazon S3 在滿足請求時不會考慮物件的名稱。這表示即便一個物件存在於多區域存取點的一個儲存貯體中，您的請求仍可以被路由到不包含該物件的儲存貯體。這樣一來，系統將會向用戶端 傳回 404 錯誤訊息。

  為避免 404 錯誤，建議您為儲存貯體設定 S3 跨區域複寫 (CRR)。當您想要的物件位於多區域存取點的儲存貯體中，但它不位於請求要路由到的特定儲存貯體中時，複寫便有助於解決潛在問題。如需設定複寫的詳細資訊，請參閱 設定複寫以搭配多區域存取點使用。

  為了確保使用您想要的特定物件來滿足您的請求，也建議您開啟儲存貯體版本控制，並在請求中包含版本 ID。這個方式有助於確保您擁有所要尋找的物件的正確版本。啟用版本控制的儲存貯體也可讓您復原意外覆寫的物件。如需更多詳細資訊，請參閱在 S3 儲存貯體中使用 S3 版本控制。
- 如果請求是建立一個資源 (例如，`PutObject` 或 `CreateMultipartUpload`)，則 Amazon S3 會使用最接近的儲存貯體來滿足請求。例如，假設一家影片公司想要支援從世界各地的影片上傳到儲存貯體。當使用者對多區域存取點提出 PUT 請求時，物件會放入最接近的儲存貯體中。若要讓全球大眾皆能以最低延遲下載該上傳影片，您能搭配使用雙向複寫和 CRR。搭配使用 CRR 與雙向複寫的話，可以確保與多區域存取點相關聯的儲存貯體中，所有內容皆保持同步。如需有關搭配多區域存取點使用複寫的詳細資訊，請參閱 設定複寫以搭配多區域存取點使用。

Amazon S3 多區域存取點容錯移轉控制

使用 Amazon S3 多區域存取點容錯移轉控制，您可以在區域流量中斷期間維持業務持續性，同時也為您的應用程式提供多區域架構以滿足合規和備援需求。如果您的區域流量中斷，您可以使用多區域存取點容錯移轉控制，來選取 Amazon S3 多區域存取點後面的哪個 AWS 區域將處理資料存取和儲存請求。

若要支援容錯移轉，您可以在主動-被動組態中設定多區域存取點，流量在正常情況下流向主動區域，以及待命時流向被動區域進行容錯移轉。

例如，若要對您選擇的 AWS 區域執行容錯移轉，您可以將流量從主要 (主動) 區域轉移至次要 (被動) 區域。在這樣的主動-被動組態中，一個儲存貯體為主動儲存貯體並接受流量，而另一個儲存貯體則為被動儲存貯體且不接受流量。被動儲存貯體用於災難復原。當您啟動容錯移轉時，所有流量 (例如 GET 或 PUT 請求) 都會導向至處於主動狀態的儲存貯體 (位於某個區域)，並遠離處於被動狀態的儲存貯體 (位於另一個區域)。

如果您已使用雙向複寫規則啟用 S3 跨區域複寫 (CRR)，則可以在容錯移轉期間將您的儲存貯體保持同步。此外，如果您已在主動-主動組態中啟用 CRR，Amazon S3 多區域存取點也可以從最接近的儲存貯體位置擷取資料，進而改善應用程式效能。

AWS 區域 支援

使用 Amazon S3 多區域存取點容錯移轉控制，您的 S3 儲存貯體可以位於支援多區域存取點的 17 個區域中的任何一個區域。您可以一次跨任何兩個區域啟動容錯移轉。

> **Note**
>
> 雖然一次只能在兩個區域之間啟動容錯移轉，但您可以在多區域存取點中同時個別更新多個區域的路由狀態。

下列主題示範如何使用和管理 Amazon S3 多存取存取點容錯移轉控制。

主題

- Amazon S3 多區域存取點路由狀態
- 使用 Amazon S3 多區域存取點容錯移轉控制
- Amazon S3 多區域存取點容錯移轉控制錯誤

Amazon S3 多區域存取點路由狀態

您的 Amazon S3 多區域存取點容錯移轉組態會決定與多區域存取點搭配使用之 AWS 區域的路由狀態。您可以將 Amazon S3 多區域存取點設定為處於主動-主動狀態或主動-被動狀態。

- 主動-主動 - 在主動-主動組態中，所有要求都會自動傳送到多區域存取點中最接近的 AWS 區域。在將多區域存取點設定為處於主動-主動狀態之後，所有區域都可以接收流量。如果主動-主動組態中發生流量中斷，網路流量會自動重新導向至其中一個主動區域。
- 主動-被動 - 在主動-被動組態中，多區域存取點中的主動區域會接收流量，而被動區域則不會接收流量。如果您打算在發生災難時使用 S3 容錯移轉控制來啟動容錯移轉，請在測試和執行災難復原規劃時，於主動-被動組態中設定多區域存取點。

使用 Amazon S3 多區域存取點容錯移轉控制

本節說明如何使用 AWS Management Console，來管理和使用您的 Amazon S3 多區域存取點容錯移轉控制。

在 AWS Management Console中多區域存取點詳細資訊頁面上的 Failover configuration (容錯移轉組態) 區段中，有兩個容錯移轉控制：Edit routing status (編輯路由狀態) 和 Failover (容錯移轉)。您可以如下使用這些控制項：

- Edit routing status (編輯路由狀態) - 您可以選擇 Edit routing status (編輯路由狀態)，在多區域存取點的單一請求中，手動編輯最多 17 個 AWS 區域中的路由狀態。您可以基於作下列用途使用 Edit routing status (編輯路由狀態)：
  - 設定或編輯多區域存取點中一或多個區域的路由狀態
  - 將兩個區域設定為處於主動-被動狀態，來建立多區域存取點的容錯移轉組態
  - 手動容錯移轉您的區域
  - 在區域之間手動切換流量
- Failover (容錯移轉) - 當您選擇 Failover (容錯移轉) 來啟動容錯移轉時，只會更新已設定為處於主動-被動狀態之兩個區域的路由狀態。在您透過選擇 Failover (容錯移轉) 所啟動的容錯移轉期間，會自動切換兩個區域之間的路由狀態。

設定多區域存取點中區域的路由狀態

您可以在多區域存取點的單一請求中手動更新最多 17 個 AWS 區域的路由狀態，方法是在多區域存取點詳細資訊頁面上的 Failover configuration (容錯移轉組態) 區段中選擇 Edit routing status (編輯路由狀態)。不過，當您選擇 Failover (容錯移轉) 來啟動容錯移轉時，只會更新已設定為處於主動-被動狀態之

兩個區域的路由狀態。在您透過選擇 Failover (容錯移轉) 所啟動的容錯移轉期間，會自動切換兩個區域之間的路由狀態。

您可以基於下列用途使用 Edit routing status (編輯路由狀態) (如下列程序所述)：

- 設定或編輯多區域存取點中一或多個區域的路由狀態
- 將兩個區域設定為處於主動-被動狀態，來建立多區域存取點的容錯移轉組態
- 手動容錯移轉您的區域
- 在區域之間手動切換流量

使用 S3 主控台

更新多區域存取點中區域的路由狀態

1. 登入 AWS 管理主控台。
2. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
3. 在左導覽窗格中，選擇 Multi-Region Access Points (多區域存取點)。
4. 選擇您要更新的多區域存取點。
5. 選擇 Replication and failover (複寫和容錯移轉) 索引標籤。
6. 選取一或多個您要編輯其路由狀態的區域。

   > **Note**
   >
   > 若要啟動容錯移轉，必須至少將一個 AWS 區域指定為 Active (主動)，而且在您的多區域存取點中必須將一個區域指定為 Passive (被動)。

7. 選擇 Edit routing status (編輯路由狀態)。
8. 在出現的對話方塊中，針對每個區域的 Routing status (路由狀態) 選取 Active (主動) 或 Passive (被動)。

   主動狀態允許將流量路由至該區域。被動狀態阻止任何流量導向至該地區。

   如果要針對多區域存取點建立容錯移轉組態，或啟動容錯移轉，必須至少將一個 AWS 區域指定為 Active (主動)，而且在您的多區域存取點中必須將一個區域指定為 Passive (被動)。
9. 選擇 Save routing status (儲存路由狀態)。重新導向流量大約需要 2 分鐘。

在針對多區域存取點提交 AWS 區域的路由狀態之後，您可以驗證路由狀態變更。若要驗證這些變更，請前往 Amazon CloudWatch (網址為 https://console.aws.amazon.com/cloudwatch/)，來監控 Amazon S3 資料請求流量 (例如，GET 和 PUT 請求) 在主動區域與被動區域之間的轉移情況。在容錯移轉期間，任何現有的連線都不會終止。現有連線會繼續進行，直到其達到成功或失敗狀態為止。

使用 AWS CLI

> **Note**
>
> 您可以對下列五個區域中的任何一個執行多區域存取點 AWS CLI 路由命令：
>
> - ap-southeast-2
> - ap-northeast-1
> - us-east-1
> - us-west-2
> - eu-west-1

下列範例命令會更新目前的多區域存取點路由組態。若要更新儲存貯體的主動或被動狀態，請將 TrafficDialPercentage 值設為 100 表示主動，以及設為 0 表示被動。在此範例中，*DOC-EXAMPLE-BUCKET-1* 會設為主動，而 *DOC-EXAMPLE-BUCKET-2* 會設為被動。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control submit-multi-region-access-point-routes
--region ap-southeast-2
--account-id 111122223333
--mrap MultiRegionAccessPoint_ARN
--route-updates Bucket=DOC-EXAMPLE-BUCKET-1,TrafficDialPercentage=100
                Bucket=DOC-EXAMPLE-BUCKET-2,TrafficDialPercentage=0
```

下列範例命令會取得您更新的多區域存取點路由組態。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 111122223333
--mrap MultiRegionAccessPoint_ARN
```

啟動容錯移轉

當您在多區域存取點詳細資料頁面的 Failover configuration 容錯移轉組態區段中選擇 Failover (容錯移轉) 來啟動容錯移轉時，Amazon S3 請求流量會自動轉移到替代 AWS 區域。容錯移轉程序會在 2 分鐘內完成。

您可以一次跨任何兩個 AWS 區域啟動容錯移轉 (有 17 個區域，其中支援多區域存取點)。然後，容錯移轉事件會記錄在 AWS CloudTrail 中。容錯移轉完成後，您可以在 Amazon CloudWatch 中監控 Amazon S3 流量，以及新主動區域的任何流量路由更新。

> **Important**
>
> 若要在資料複寫期間跨儲存貯體將所有中繼資料和物件保持同步，建議您建立雙向複寫規則，並啟用複本修改同步，然後再設定您的容錯移轉控制。
> 雙向複寫規則有助於確保將資料寫入流量容錯移轉至其中的 Amazon S3 儲存貯體時，該資料隨後會複寫回來源儲存貯體。複本修改同步有助於確保在雙向複寫期間，物件中繼資料也會在儲存貯體之間同步。
> 如需設定複寫以支援容錯移轉的詳細資訊，請參閱 the section called “儲存貯體複寫”。

在複寫的儲存貯體之間啟動容錯移轉

1. 登入 AWS 管理主控台。
2. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
3. 在左導覽窗格中，選擇 Multi-Region Access Points (多區域存取點)。
4. 選擇您要用來啟動容錯移轉的多區域存取點。
5. 選擇 Replication and failover (複寫和容錯移轉) 索引標籤。
6. 向下捲動至 Failover configuration (容錯移轉組態) 區段，然後選取兩個 AWS 區域。

   > **Note**
   >
   > 若要啟動容錯移轉，必須至少將一個 AWS 區域指定為 Active (主動)，而且在您的多區域存取點中必須將一個區域指定為 Passive (被動)。主動狀態允許將流量導向至某個區域。被動狀態阻止任何流量導向至該地區。

7. 選擇 Failover (容錯移轉)。

8. 在對話方塊中，再次選擇 Failover (容錯移轉) 來啟動容錯移轉程序。在此過程中，會自動切換兩個區域的路由狀態。所有新流量都會導向到變成主動的區域，並且流量會阻止導向至變成被動的區域。重新導向流量大約需要 2 分鐘。

   啟動容錯移轉程序之後，您可以驗證流量變更。若要驗證這些變更，請前往 Amazon CloudWatch (網址為 https://console.aws.amazon.com/cloudwatch/)，來監控 Amazon S3 資料請求流量 (例如，GET 和 PUT 請求) 在主動區域與被動區域之間的轉移情況。在容錯移轉期間，任何現有的連線都不會終止。現有連線會繼續進行，直到其達到成功或失敗狀態為止。

檢視您的 Amazon S3 多區域存取點路由狀態

使用 S3 主控台

檢視您的 Amazon S3 多區域存取點的路由狀態

1. 登入 AWS 管理主控台。
2. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
3. 在左導覽窗格中，選擇 Multi-Region Access Points (多區域存取點)。
4. 選擇您要檢閱的多區域存取點。
5. 選擇 Replication and failover (複寫和容錯移轉) 索引標籤。此頁面顯示多區域存取點的路由組態詳細資訊和摘要、相關聯的複寫規則，以及複寫指標。您可以在 Failover configuration (容錯移轉組態) 區段中查看區域的路由狀態。

使用 AWS CLI

下列範例 AWS CLI 命令會取得指定區域的目前多區域存取點路由組態。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 111122223333
--mrap MultiRegionAccessPoint_ARN
```

> **Note**
>
> 此命令只能針對下列五個區域執行：
>
> - ap-southeast-2

- ap-northeast-1
- us-east-1
- us-west-2
- eu-west-1

Amazon S3 多區域存取點容錯移轉控制錯誤

當更新多區域存取點的容錯移轉組態時，您可能會遇到下列其中一個錯誤：

- HTTP 400 錯誤的請求如果您在更新容錯移轉組態時輸入無效的多區域存取點 ARN，就會發生這種錯誤。您可以檢閱多區域存取點政策，以確認您的多區域存取點 ARN。若要檢閱或更新您的多區域存取點政策，請參閱編輯多區域存取點政策。如果您在更新 Amazon S3 多區域存取點容錯移轉控制時使用空字串或隨機字串，也可能發生此錯誤。請確定使用多區域存取點 ARN 格式：

  `arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`
- HTTP 503 速度變慢：如果您在短時間內傳送太多請求，就會發生此錯誤。遭拒的請求會導致錯誤。
- HTTP 409 衝突：當兩個以上並行路由組態更新請求將單一多區域存取點設為目標時，就會發生此錯誤。第一個請求成功，但任何其他請求都會失敗並顯示錯誤。
- HTTP 405 方法不允許：當您在啟動容錯移轉時選取只有一個 AWS 區域的多區域存取點時，就會發生此錯誤。您必須選取兩個區域，然後才能啟動容錯移轉。否則會傳回一個錯誤。

設定複寫以搭配多區域存取點使用

當您對多區域存取點端點提出請求時，Amazon S3 會自動將請求路由到最接近您的儲存貯體。在做出這個決策時，Amazon S3 不會考慮請求的內容。如果您向 GET 物件提出請求，則您的請求可能會被路由到沒有此物件複本的儲存貯體。如果發生這種情況，您會收到 HTTP 狀態碼 404 (找不到) 錯誤。如需有關多區域存取點請求路由的詳細資訊，請參閱 the section called “要求路由”。

如果您希望多區域存取點能夠擷取物件，而不管哪個儲存貯體收到請求，則您必須設定 Amazon S3 跨區域複寫 (CRR)。

例如，請考慮具有三個儲存貯體的多區域存取點：

- 區域 us-west-2 中名為 my-bucket-usw2 的儲存貯體，其中包含物件 my-image.jpg
- 區域 ap-south-1 中名為 my-bucket-aps1 的儲存貯體，其中包含物件 my-image.jpg
- 區域 eu-central-1 中名為 my-bucket-euc1 的儲存貯體，其中不包含物件 my-image.jpg

在此情況下，如果您對物件 `my-image.jpg` 提出 `GetObject` 請求，則該請求的成功與否取決於哪個儲存貯體會收到您的請求。由於 Amazon S3 不會考慮請求的內容，因此可能會將您的 `GetObject` 請求路由到 `my-bucket-euc1` 儲存貯體 (如果該儲存貯體回應最接近)。即使您的物件位於多區域存取點的儲存貯體中，您也會收到 404 (找不到) 錯誤，因為接收請求的個別儲存貯體沒有該物件。

啟用跨區域複寫 (CRR) 可協助避免此結果。使用適當的複製規則，`my-image.jpg` 物件便會複製到 `my-bucket-euc1` 儲存貯體。因此，如果 Amazon S3 將您的請求路由到該儲存貯體，您現在可以擷取物件。

對指派給多區域存取點的儲存貯體，複寫功能會正常運作。Amazon S3 不會對位於多區域存取點中的儲存貯體執行任何特殊複寫處理。如需有關在您的儲存貯體中設定複寫的詳細資訊，請參閱 設定即時複製。

搭配多區域存取點使用複寫的建議

如需在使用多區域存取點時取得最佳的複寫效能，建議您採取下列動作：

- 設定 S3 複寫時間控制 (S3 RTC)。若要在可預測的時間範圍內跨不同區域複寫您的資料，您可以使用 S3 RTC。S3 RTC 會在 15 分鐘內，複寫 99.99% 在 Amazon S3 中存放的新物件 (由服務水準協議支援)。如需更多詳細資訊，請參閱 the section called “使用 S3 複寫時間控制”。對於 S3 RTC 需另外付費。如需詳細資訊，請參閱 Amazon S3 定價。
- 使用雙向複寫，以支援在透過多區域存取點更新儲存貯體時，保持儲存貯體同步。如需更多詳細資訊，請參閱 the section called “針對您的多區域存取點建立雙向複寫規則”。
- 建立跨帳戶多區域存取點，將資料複寫到個別 AWS 帳戶 中的儲存貯體。此方法提供帳戶層級的分隔，因此可以從來源儲存貯體以外，不同區域的不同帳戶存取和複寫資料。設定跨帳戶多區域存取點無需額外費用。如果您是儲存貯體擁有者，但不擁有多區域存取點，則只需支付資料傳輸和請求費用。多區域存取點擁有者需支付資料路由和網際網路加速費用。如需詳細資訊，請參閱 Simple Storage Service (Amazon S3) 定價。
- 針對每個複寫規則啟用複本修改同步，也會將您物件的中繼資料變更保持同步。如需詳細資訊，請參閱啟用複本修改同步。
- 啟用 Amazon CloudWatch 指標來監控複寫事件。CloudWatch 指標費用適用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

主題

針對您的多區域存取點建立雙向複寫規則

複寫規則可讓物件跨區域進行自動和非同步複製。單向複寫規則有助於確保將資料從一個 AWS 區域的一個來源儲存貯體完全複寫到另一個區域的目的地儲存貯體。當設定單向複寫時，會建立從來源儲存貯體 (DOC-EXAMPLE-BUCKET-1) 複寫至目的地儲存貯體 (DOC-EXAMPLE-BUCKET-2) 的複寫規則。如同所有複寫規則，您可以將單向複寫規則套用至整個 Amazon S3 儲存貯體，也可以套用至由字首或物件索引標籤篩選的物件子集。

> **Important**
>
> 如果您的使用者只會使用目的地儲存貯體中的物件，建議您使用單向複寫。如果您的使用者要上傳或修改目的地儲存貯體中的物件，請使用雙向複寫，保持所有儲存貯體同步。如果您打算將多區域存取點用於容錯移轉，建議您使用雙向複寫。若要設定雙向複寫，請參閱 the section called “針對您的多區域存取點建立雙向複寫規則”。

針對您的多區域存取點建立單向複寫規則

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Multi-Region Access Points (多區域存取點)。
3. 選擇多區域存取點名稱。
4. 選擇 Replication and failover (複寫和容錯移轉) 索引標籤。
5. 向下捲動至 Replication rules (複寫規則)，然後選擇 Create replication rules (建立複寫規則)。請確保您有足夠的權限建立複寫規則，否則版本控制將遭停用。

   > **Note**
   >
   > 您只能針對帳戶內的儲存貯體建立複寫規則。若要為外部儲存貯體建立複寫規則，儲存貯體擁有者必須為這些儲存貯體建立複寫規則。

6. 在建立複寫規則頁面上，選擇將物件從一或多個來源儲存貯體複寫到一或多個目的地儲存貯體範本。

   > **Important**
   >
   > 當您使用此範本建立複寫規則時，它們會取代任何已指派給儲存貯體的現有複寫規則。

若要新增或修改任何現有的複寫規則，而不是取代它們，請前往主控台中每個儲存貯體的 Management (管理) 索引標籤，然後在 Replication rules (複寫規則) 區段中編輯規則。您也可以使用 AWS CLI、SDK 或 REST API 來新增或修改現有的複寫規則。如需更多詳細資訊，請參閱 複寫組態。

7. 在來源和目的地區段中，請在來源儲存貯體下，選取一或多個您要從中複寫物件的儲存貯體。針對複寫選擇的所有儲存貯體 (來源和目的地) 都必須啟用 S3 版本控制，且每個儲存貯體必須位於不同的 AWS 區域 中。如需 S3 版本控制的詳細資訊，請參閱在 Amazon S3 儲存貯體中使用儲存貯體。

   在目的地儲存貯體下，選取一或多個您要複寫物件的目的地儲存貯體。

   > **Note**
   >
   > 請確認您具有建立複寫所需的讀取和複寫權限，否則將出現錯誤。如需更多詳細資訊，請參閱建立 IAM 角色。

8. 在 Replication rule configuration (複寫規則組態) 區段中，選擇複寫規則在建立時將 Enabled (啟用) 還是 Disabled (停用)。

   > **Note**
   >
   > 您無法在 Replication rule name (複寫規則名稱) 方塊中輸入名稱。建立複寫規則時，會根據您的組態產生複寫規則名稱。

9. 在 Scope (範圍) 區段中，針對您的複寫選擇適當的範圍。

   - 若要複寫整個儲存貯體，請選擇 Apply to all objects in the bucket (套用至儲存貯體中的所有物件)。
   - 若要複寫儲存貯體中的物件子集，請選擇Limit the scope of this rule using one or more filters (使用一或多個篩選器限制此規則的範圍)。

     您可以使用字首、物件索引標籤或兩者的組合來篩選物件。

     - 若要限制複寫名稱以相同字串 (例如，`pictures`) 開頭的所有物件，請在 Prefix (字首) 方塊中輸入字首。

       如果您輸入的字首是資料夾名稱，請務必使用 / (正斜線) 等分隔符號來表示階層 (例如，`pictures/`)。如需詳細了解字首，請參閱使用字首組織物件。

- 若要複寫具有一個或多個物件索引標籤的所有物件，請選擇 Add tag (新增標籤)，然後在方塊中輸入鍵/值對。若要新增另一個索引標籤，請重複此程序，。如需物件標籤的詳細資訊，請參閱 使用標籤分類儲存空間。

10. 向下捲動至 Additional replication options (其他複寫選項) 區段，然後選取您要套用的複寫選項。

> **Note**
>
> 建議您套用下列選項：
>
> - Replication time control (RTC) (複寫時間控制 (RTC)) - 若要在可預測的時間範圍內跨不同區域複寫您的資料，您可以使用 S3 複寫時間控制 (S3 RTC)。S3 RTC 會在 15 分鐘內，複寫 99.99% 在 Amazon S3 中存放的新物件 (由服務水準協議支援)。如需更多詳細資訊，請參閱 the section called “使用 S3 複寫時間控制”。
> - Replication metrics and notifications (複寫指標和通知) - 啟用 Amazon CloudWatch 指標，以監控複寫事件。
> - 刪除標記複寫 — 複寫由 S3 刪除操作建立的刪除標記。由生命週期規則建立的刪除標記不會複寫。如需詳細資訊，請參閱在儲存貯體間複寫刪除標記。
>
> 對於 S3 RTC 和 CloudWatch 複寫指標和通知需支付額外費用。如需詳細資訊，請參閱 Amazon S3 定價和 Amazon CloudWatch 定價。

11. 如果您正在撰寫新的複寫規則，取代現有複寫規則，請選取 I acknowledge that by choosing Create replication rules, these existing replication rules will be overwritten (我確認藉由選擇 Create replication rules (建立複寫規則)，這些現有的複寫規則將遭到覆寫)。
12. 選擇建立複寫規則，以建立並儲存新的單向複寫規則。

針對您的多區域存取點建立雙向複寫規則

複寫規則可讓物件跨區域進行自動和非同步複製。雙向複寫規則可確保資料在不同 AWS 區域 的兩個以上儲存貯體之間完全同步。當設定雙向複寫時，會建立從來源儲存貯體 (DOC-EXAMPLE-BUCKET-1) 複寫至包含複本之儲存貯體 (DOC-EXAMPLE-BUCKET-2) 的複寫規則。然後，建立從包含複本 (DOC-EXAMPLE-BUCKET-2) 的儲存貯體複寫至來源儲存貯體 (DOC-EXAMPLE-BUCKET-1) 的第二個複寫規則。

如同所有複寫規則，您可以將雙向複寫規則套用至整個 Amazon S3 儲存貯體，也可以套用至由字首或物件索引標籤篩選的物件子集。您也可以針對每個複寫規則啟用複本修改同步，將您物件的中繼資

料變更保持同步。您可以透過 Amazon S3 主控台、AWS CLI、AWS SDK、Amazon S3 REST API 或 AWS CloudFormation 啟用複本修改同步。

若要監控 Amazon CloudWatch 中物件和物件中繼資料的複寫進度，請啟用 S3 複寫指標和通知。如需詳細資訊，請參閱使用複寫指標和 Amazon S3 事件通知監控進度。

針對您的多區域存取點建立雙向複寫規則

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Multi-Region Access Points (多區域存取點)。
3. 選擇您要更新的多區域存取點名稱。
4. 選擇 Replication and failover (複寫和容錯移轉) 索引標籤。
5. 向下捲動至 Replication rules (複寫規則)，然後選擇 Create replication rules (建立複寫規則)。
6. 在 Create replication rules (建立複製規則) 頁面上，選擇 Replicate objects among all specified buckets (在所有指定儲存貯體之間複製物件) 範本。Replicate objects among all specified buckets (在所有指定儲存貯體之間複製物件) 範本會針對您的儲存貯體設定雙向複寫 (具有容錯移轉功能)。

   > **Important**
   >
   > 當您使用此範本建立複寫規則時，它們會取代任何已指派給儲存貯體的現有複寫規則。若要新增或修改任何現有的複寫規則，而不是取代它們，請前往主控台中每個儲存貯體的 Management (管理) 索引標籤，然後在 Replication rules (複寫規則) 區段中編輯規則。您也可以使用 AWS CLI、AWS SDK 或 Amazon S3 REST API 來新增或修改現有的複寫規則。如需更多詳細資訊，請參閱 複寫組態。

7. 在 Buckets (儲存貯體) 區段中，至少選取兩個您要從中複寫物件的儲存貯體。針對複寫選擇的所有儲存貯體都必須啟用 S3 版本控制，且每個儲存貯體必須位於不同的 AWS 區域中。如需 S3 版本控制的詳細資訊，請參閱在 Amazon S3 儲存貯體中使用儲存貯體。

   > **Note**
   >
   > 請確認您具有建立複寫所需的讀取和複寫權限，否則將出現錯誤。如需更多詳細資訊，請參閱建立 IAM 角色。

8. 在 Replication rule configuration (複寫規則組態) 區段中，選擇複寫規則在建立時將 Enabled (啟用) 還是 Disabled (停用)。

> **Note**
>
> 您無法在 Replication rule name (複寫規則名稱) 方塊中輸入名稱。建立複寫規則時，會根據您的組態產生複寫規則名稱。

9. 在 Scope (範圍) 區段中，針對您的複寫選擇適當的範圍。

   - 若要複寫整個儲存貯體，請選擇 Apply to all objects in the bucket (套用至儲存貯體中的所有物件)。
   - 若要複寫儲存貯體中的物件子集，請選擇Limit the scope of this rule using one or more filters (使用一或多個篩選器限制此規則的範圍)。

     您可以使用字首、物件索引標籤或兩者的組合來篩選物件。

     - 若要限制複寫名稱以相同字串 (例如，`pictures`) 開頭的所有物件，請在 Prefix (字首) 方塊中輸入字首。

       如果您輸入的字首是資料夾名稱，您必須使用 / (正斜線) 作為最後一個字元 (例如，`pictures/`)。
     - 若要複寫具有一個或多個物件索引標籤的所有物件，請選擇 Add tag (新增標籤)，然後在方塊中輸入鍵/值對。若要新增另一個索引標籤，請重複此程序，。如需物件標籤的詳細資訊，請參閱 使用標籤分類儲存空間。

10. 向下捲動至 Additional replication options (其他複寫選項) 區段，然後選取您要套用的複寫選項。

> **Note**
>
> 建議您套用下列選項，尤其是當您打算將多區域存取點設定為支援容錯移轉時：
>
> - Replication time control (RTC) (複寫時間控制 (RTC)) - 若要在可預測的時間範圍內跨不同區域複寫您的資料，您可以使用 S3 複寫時間控制 (S3 RTC)。S3 RTC 會在 15 分鐘內，複寫 99.99% 在 Amazon S3 中存放的新物件 (由服務水準協議支援)。如需更多詳細資訊，請參閱 the section called “使用 S3 複寫時間控制”。
> - Replication metrics and notifications (複寫指標和通知) - 啟用 Amazon CloudWatch 指標，以監控複寫事件。
> - 刪除標記複寫 — 複寫由 S3 刪除操作建立的刪除標記。由生命週期規則建立的刪除標記不會複寫。如需詳細資訊，請參閱在儲存貯體間複寫刪除標記。

- Replica modification sync (複本修改同步) - 針對每個複寫規則啟用複本修改同步，也會將您物件的中繼資料變更保持同步。如需詳細資訊，請參閱啟用複本修改同步。

對於 S3 RTC 和 CloudWatch 複寫指標和通知需支付額外費用。如需詳細資訊，請參閱 Amazon S3 定價和 Amazon CloudWatch 定價。

11. 如果您正在撰寫新的複寫規則，取代現有複寫規則，請選取 I acknowledge that by choosing Create replication rules, these existing replication rules will be overwritten (我確認藉由選擇 Create replication rules (建立複寫規則)，這些現有的複寫規則將遭到覆寫)。
12. 選擇 Create replication rules (建立複寫規則)，以建立並儲存新的雙向複寫規則。

檢視您的多區域存取點複寫規則

透過多區域存取點，您便可以設定單向或雙向複寫規則。如需詳細了解如何管理複寫規則，請參閱使用 Amazon S3 主控台管理複寫規則。

檢視您的多區域存取點複寫規則

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Multi-Region Access Points (多區域存取點)。
3. 選擇多區域存取點名稱。
4. 選擇 Replication and failover (複寫和容錯移轉) 索引標籤。
5. 向下捲動至複寫規則區段。本節列出所有針對您的多區域存取點，所建立的複寫規則。

> **Note**
>
> 若您已從其他帳戶新增儲存貯體至此多區域存取點，則必須具有儲存貯體擁有者的 `s3:GetBucketReplication` 權限，才能檢視該儲存貯體的複寫規則。

搭配支援的 API 操作使用多區域存取點

Amazon S3 提供一組可管理多區域存取點的操作。Amazon S3 會同步處理其中一些操作，也會非同步處理另一些操作。當您叫用非同步操作時，Amazon S3 會先同步授權請求的操作。如果授權成功，Amazon S3 會傳回一個字符，而您可以用它來追蹤請求的操作的進度和結果。

> **Note**
>
> 透過 Amazon S3 主控台提出的請求始終同步。主控台會等到請求完成後，才會允許您提交其他請求。

您可以使用主控台來檢視非同步作業的目前狀態和結果，也可以`DescribeMultiRegionAccessPointOperation`在 AWS CLI、 AWS SDK 或 REST API 中使用。Amazon S3 會在回應非同步操作時提供追蹤字符。您可以將該追蹤字符作為 `DescribeMultiRegionAccessPointOperation` 的參數。當您包括追蹤字符時，Amazon S3 接著會傳回指定操作的目前狀態和結果，包括任何錯誤或相關的資源資訊。Amazon S3 會同步執行 `DescribeMultiRegionAccessPointOperation` 操作。

所有建立或維護多區域存取點的控制平面請求，都必須路由至 `US West (Oregon)` 區域。對於多區域存取點資料平面請求，不需要指定區域。對於多區域存取點容錯移轉控制平面，請求必須路由至五個受支援區域的其中一個。如需多區域存取點支援區域的詳細資訊，請參閱多區域存取點約束與限制。

此外，您必須將`s3:ListAllMyBuckets`權限授與提出管理多區域存取點請求的使用者、角色或其他 AWS Identity and Access Management (IAM) 實體。

下列範例示範如何在 Amazon S3 中搭配相容操作使用多區域存取點。

主題

- 與 SDK 的多區域存取點相 AWS 服務 容性 AWS
- 多區域存取點與 S3 操作的相容性
- 檢視您的多區域存取點路由狀態
- 更新您的基礎 Amazon S3 儲存貯體政策
- 更新多區域存取點路由組態
- 在您的多區域存取點中將物件新增至儲存貯體
- 從您的多區域存取點中擷取物件
- 列出存放在以多區域存取點為基礎之儲存貯體的物件
- 搭配多區域存取點使用預先簽章的 URL
- 使用以多區域存取點設定為請求者付款的儲存貯體

與 SDK 的多區域存取點相 AWS 服務 容性 AWS

若要將多區域存取點用於需要 Amazon S3 儲存貯體名稱的應用程式，請在使用 SDK 提出請求時，使用多區域存取點的 Amazon 資源名稱 (ARN)。 AWS 若要檢查哪些 AWS SDK 與多區域存取點相容，請參閱與 AWS SDK 的相容性。

多區域存取點與 S3 操作的相容性

您可以使用下列 Amazon S3 資料平面 API 操作，對儲存貯體中與多區域存取點相關聯的物件執行動作。下列 S3 操作可以接受多區域存取點 ARN：

- AbortMultipartUpload
- CompleteMultipartUpload
- CreateMultipartUpload
- DeleteObject
- DeleteObjectTagging
- GetObject
- GetObjectAcl
- GetObjectLegalHold
- GetObjectRetention
- GetObjectTagging
- HeadObject
- ListMultipartUploads
- ListObjectsV2
- ListParts
- PutObject
- PutObjectAcl
- PutObjectLegalHold
- PutObjectRetention
- PutObjectTagging
- RestoreObject
- UploadPart

> **Note**
>
> 使用「多區域存取點」ARN 時，「多區域存取點」僅支援使用「多區域存取點」作為目的地的複製作業。

您可以使用下列 Amazon S3 控制平面操作，來建立和管理您的多區域存取點：

- CreateMultiRegionAccessPoint
- DescribeMultiRegionAccessPointOperation
- GetMultiRegionAccessPoint
- GetMultiRegionAccessPointPolicy
- GetMultiRegionAccessPointPolicyStatus
- GetMultiRegionAccessPointRoutes
- ListMultiRegionAccessPoints
- PutMultiRegionAccessPointPolicy
- SubmitMultiRegionAccessPointRoutes

檢視您的多區域存取點路由狀態

AWS CLI

下列範例命令會擷取您的多區域存取點路由組態，以便您可以查看儲存貯體目前的路由狀態。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 111122223333
--mrap arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap
```

SDK for Java

下列適用於 Java 的 SDK 程式碼會擷取您的多區域存取點路由組態，以便您可以查看儲存貯體目前的路由狀態。若要使用此範例語法，請以您自己的資訊取代 *user input placeholders*。

```
S3ControlClient s3ControlClient = S3ControlClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(credentialsProvider)
```

```
    .build();

GetMultiRegionAccessPointRoutesRequest request =
 GetMultiRegionAccessPointRoutesRequest.builder()
    .accountId("111122223333")
    .mrap("arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap")
    .build();

GetMultiRegionAccessPointRoutesResponse response =
 s3ControlClient.getMultiRegionAccessPointRoutes(request);
```

SDK for JavaScript

下列程 JavaScript 式碼 SDK 會擷取您的「多區域存取點」路由設定，以便您查看值區目前的路由狀態。若要使用此範例語法，請以您自己的資訊取代 *user input placeholders*。

```
const REGION = 'us-east-1'

const s3ControlClient = new S3ControlClient({
  region: REGION
})

export const run = async () => {
  try {
    const data = await s3ControlClient.send(
      new GetMultiRegionAccessPointRoutesCommand({
        AccountId: '111122223333',
        Mrap: 'arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap',
      })
    )
    console.log('Success', data)
    return data
  } catch (err) {
    console.log('Error', err)
  }
}

run()
```

SDK for Python

下列適用於 Python 的 SDK 程式碼會擷取您的多區域存取點路由組態，以便您可以查看儲存貯體目前的路由狀態。若要使用此範例語法，請以您自己的資訊取代 *user input placeholders*。

```
s3.get_multi_region_access_point_routes(
        AccountId=111122223333,
        Mrap=arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap)['Routes']
```

更新您的基礎 Amazon S3 儲存貯體政策

若要授予適當的存取權，您也必須更新基礎 Amazon S3 儲存貯體政策。下列範例將存取控制委派給多區域存取點政策。將存取控制委派給多區域存取點政策後，當透過多區域存取點進行請求時，系統不會再使用該儲存貯體政策來存取控制。

以下是範例儲存貯體政策，可將存取控制委派給多區域存取點政策。若要使用此範例儲存貯體政策，請以您自己的資訊取代 *user input placeholders*。若要透過 AWS CLI put-bucket-policy命令套用此原則，如下一個範例所示，請將原則儲存在檔案中，例如policy.json。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": { "AWS": "*" },
    "Effect": "Allow",
    "Action": ["s3:*"],
    "Resource": ["arn:aws:s3:::111122223333/*", "arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
    "Condition": {
      "StringEquals": {
        "s3:DataAccessPointAccount": "444455556666"
      }
    }
  }
}
```

下列 put-bucket-policy 範例命令會將更新的 S3 儲存貯體政策與您的 S3 儲存貯體建立關聯：

```
aws s3api put-bucket-policy
  --bucket DOC-EXAMPLE-BUCKET
  --policy file:///tmp/policy.json
```

更新多區域存取點路由組態

下列範例命令會更新多區域存取點路由組態。可以針對下列五個區域執行多區域存取點路由命令：

- ap-southeast-2

- ap-northeast-1
- us-east-1
- us-west-2
- eu-west-1

在多區域存取點路由組態中，您可以將儲存貯體設為主動或被動路由狀態。主動儲存貯體會接收流量，而被動儲存貯體則不會接收流量。您可以將儲存貯體的 TrafficDialPercentage 值設為 100 表示主動或 0 表示被動，來設定儲存貯體的路由狀態。

AWS CLI

下列範例命令會更新您的多區域存取點路由組態。在此範例中，*DOC-EXAMPLE-BUCKET1* 會設為主動狀態，而 *DOC-EXAMPLE-BUCKET2* 會設為被動狀態。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control submit-multi-region-access-point-routes
--region ap-southeast-2
--account-id 111122223333
--mrap arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap
--route-updates Bucket=DOC-EXAMPLE-BUCKET1,TrafficDialPercentage=100
                Bucket=DOC-EXAMPLE-BUCKET2,TrafficDialPercentage=0
```

SDK for Java

下列適用於 Java 的 SDK 程式碼會更新您的多區域存取點路由組態。若要使用此範例語法，請以您自己的資訊取代 *user input placeholders*。

```
S3ControlClient s3ControlClient = S3ControlClient.builder()
    .region(Region.ap-southeast-2)
    .credentialsProvider(credentialsProvider)
    .build();

SubmitMultiRegionAccessPointRoutesRequest request =
 SubmitMultiRegionAccessPointRoutesRequest.builder()
    .accountId("111122223333")
    .mrap("arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap")
    .routeUpdates(
        MultiRegionAccessPointRoute.builder()
            .region("eu-west-1")
            .trafficDialPercentage(100)
```

```
            .build(),
        MultiRegionAccessPointRoute.builder()
            .region("ca-central-1")
            .bucket("111122223333")
            .trafficDialPercentage(0)
            .build()
    )
    .build();

SubmitMultiRegionAccessPointRoutesResponse response =
 s3ControlClient.submitMultiRegionAccessPointRoutes(request);
```

SDK for JavaScript

下列 JavaScript 程式碼 SDK 會更新您的多區域存取點路由設定。若要使用此範例語法，請以您自己的資訊取代 *user input placeholders*。

```
const REGION = 'ap-southeast-2'

const s3ControlClient = new S3ControlClient({
  region: REGION
})

export const run = async () => {
  try {
    const data = await s3ControlClient.send(
      new SubmitMultiRegionAccessPointRoutesCommand({
        AccountId: '111122223333',
        Mrap: 'arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap',
        RouteUpdates: [
          {
            Region: 'eu-west-1',
            TrafficDialPercentage: 100,
          },
          {
            Region: 'ca-central-1',
            Bucket: 'DOC-EXAMPLE-BUCKET1',
            TrafficDialPercentage: 0,
          },
        ],
      })
    )
    console.log('Success', data)
```

```
    return data
  } catch (err) {
    console.log('Error', err)
  }
}

run()
```

SDK for Python

下列適用於 Python 的 SDK 程式碼會更新您的多區域存取點路由組態。若要使用此範例語法，請以您自己的資訊取代 *user input placeholders*。

```
s3.submit_multi_region_access_point_routes(
        AccountId=111122223333,
        Mrap=arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap,
        RouteUpdates= [{
            'Bucket': DOC-EXAMPLE-BUCKET,
            'Region': ap-southeast-2,
            'TrafficDialPercentage': 10
        }])
```

在您的多區域存取點中將物件新增至儲存貯體

若要將與多區域存取點相關聯的物件新增至儲存貯體，您可以使用 PutObject 操作。若要將多區域存取點中的所有儲存貯體保持同步，請啟用跨區域複寫。

> **Note**
>
> 若要使用這項操作，您必須具有多區域存取點的 `s3:PutObject` 許可。如需多區域存取點許可需求的詳細資訊，請參閱 許可。

AWS CLI

下列範例資料平面請求會將 *example.txt* 上傳至指定的多區域存取點。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
aws s3api put-object --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap --key example.txt --
body example.txt
```

SDK for Java

```
S3Client s3Client = S3Client.builder()
        .build();

PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
        .key("example.txt")
        .build();

s3Client.putObject(objectRequest, RequestBody.fromString("Hello S3!"));
```

SDK for JavaScript

```
const client = new S3Client({});

async function putObjectExample() {
    const command = new PutObjectCommand({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
        Key: "example.txt",
        Body: "Hello S3!",
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3

client = boto3.client('s3')
client.put_object(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap',
```

```
    Key='example.txt',
    Body='Hello S3!'
)
```

從您的多區域存取點中擷取物件

若要從多區域存取點中擷取物件，您可以使用 GetObject 操作。

> **Note**
>
> 若要使用這項 API 操作，您必須具有多區域存取點的 `s3:GetObject` 許可。如需多區域存取點許可需求的詳細資訊，請參閱 許可。

AWS CLI

下列範例資料平面請求會從指定的多區域存取點中擷取 *example.txt*，並將其下載為 *downloaded_example.txt*。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
aws s3api get-object --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap --
key example.txt downloaded_example.txt
```

SDK for Java

```
S3Client s3 = S3Client
   .builder()
   .build();

GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
    .key("example.txt")
    .build();

s3Client.getObject(getObjectRequest);
```

SDK for JavaScript

```
const client = new S3Client({})
```

```
async function getObjectExample() {
    const command = new GetObjectCommand({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
        Key: "example.txt"
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3

client = boto3.client('s3')
client.get_object(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap',
    Key='example.txt'
)
```

列出存放在以多區域存取點為基礎之儲存貯體的物件

若要傳回存放在以多區域存取點為基礎之儲存貯體的物件清單，請使用 ListObjectsV2 操作。在下列範例命令中，使用多區域存取點的 ARN，列出指定多區域存取點的所有物件。在此情況下，多區域存取點 ARN 為：

`arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap`

> **Note**
>
> 若要使用這項 API 操作，您必須具有多區域存取點的 `s3:ListBucket` 許可和基礎儲存貯體。如需多區域存取點許可需求的詳細資訊，請參閱 許可。

AWS CLI

下列範例資料平面請求會列出以 ARN 所指定多區域存取點為基礎之儲存貯體中的物件。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
aws s3api list-objects-v2 --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap
```

SDK for Java

```
S3Client s3Client = S3Client.builder()
        .build();

String bucketName = "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap";

ListObjectsV2Request listObjectsRequest = ListObjectsV2Request
    .builder()
    .bucket(bucketName)
    .build();

 s3Client.listObjectsV2(listObjectsRequest);
```

SDK for JavaScript

```
const client = new S3Client({});

async function listObjectsExample() {
    const command = new ListObjectsV2Command({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3
```

```
client = boto3.client('s3')
client.list_objects_v2(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap'
)
```

搭配多區域存取點使用預先簽章的 URL

您可以使用預先簽章的 URL 來產生一個 URL，允許其他人透過 Amazon S3 多區域存取點存取 Amazon S3 儲存貯體。當建立預先簽章的 URL 時，您會將其與 S3 上傳 (PutObject) 或 S3 下載 (GetObject) 等特定物件動作建立關聯。您可以共用預先簽章的 URL，且擁有存取權的任何人都可以執行內嵌在 URL 中的動作，如同原始簽章使用者一樣。

預先簽章的 URL 具有到期日。過了到期時間，URL 將不再運作起作用。

在您搭配預先簽章的 URL 使用 S3 多區域存取點之前，請先檢查 AWS SDK 與 SigV4a 演算法的相容性。驗證您的 SDK 版本是否支援 SigV4a 作為簽署實作，用來簽署全域 AWS 區域 請求。如需搭配 Amazon S3 使用預先簽章 URL 的詳細資訊，請參閱使用預先簽章的 URL 共用物件。

下列範例示範如何搭配預先簽章的 URL 使用多區域存取點。若要使用這些範例，請以您自己的資訊取代 *user input placeholders*。

AWS CLI

```
aws s3 presign
 arn:aws:s3::123456789012:accesspoint/MultiRegionAccessPoint_alias/example-file.txt
```

SDK for Python

```
import logging
import boto3
from botocore.exceptions import ClientError

s3_client = boto3.client('s3',aws_access_key_id='xxx',aws_secret_access_key='xxx')
s3_client.generate_presigned_url(HttpMethod='PUT',ClientMethod="put_object",
 Params={'Bucket':'arn:aws:s3::123456789012:accesspoint/
abcdef0123456.mrap','Key':'example-file'})
```

SDK for Java

```
S3Presigner s3Presigner = S3Presigner.builder()
```

```
    .credentialsProvider(StsAssumeRoleCredentialsProvider.builder()
        .refreshRequest(assumeRole)
        .stsClient(stsClient)
        .build())
    .build();

GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
    .key("example-file")
    .build();

GetObjectPresignRequest preSignedReq = GetObjectPresignRequest.builder()
    .getObjectRequest(getObjectRequest)
    .signatureDuration(Duration.ofMinutes(10))
    .build();

PresignedGetObjectRequest presignedGetObjectRequest =
 s3Presigner.presignGetObject(preSignedReq);
```

> **Note**
>
> 若要將 SigV4a 與臨時安全登入資料搭配使用 (例如，使用 IAM 角色時)，請務必從 AWS Security Token Service ()AWS STS中的區域端點申請臨時登入資料，而非全域端點。如果您使用 AWS STS (`sts.amazonaws.com`) 的全域端點， AWS STS 將會從全域端點產生臨時認證，而 Sig4A 不支援。因此，您會收到錯誤。若要解決此問題，請使用列出的任何區域端點 AWS STS。

使用以多區域存取點設定為請求者付款的儲存貯體

若與多區域存取點相關聯的 S3 儲存貯體設定為使用請求者付款，請求者將支付儲存貯體請求、下載和任何多區域存取點的相關費用。如需詳細資訊，請參閱 Simple Storage Service (Amazon S3) 定價。

以下範例是多區域存取點的資料平面請求，該多區域存取點連接至請求者付款儲存貯體。

AWS CLI

若要從連接至請求者付款儲存貯體的多區域存取點下載物件，您必須將 `--request-payer requester` 指定為 get-object 請求的一部分。您也必須指定儲存貯體中檔案的名稱，以及應存放下載檔案的位置。

```
aws s3api get-object --bucket MultiRegionAccessPoint_ARN --request-payer requester
 --key example-file-in-bucket.txt example-location-of-downloaded-file.txt
```

SDK for Java

若要從連接至請求者付款儲存貯體的多區域存取點下載物件，您必須將 RequestPayer.REQUESTER 指定為 GetObject 請求的一部分。您也必須指定儲存貯體中檔案的名稱，以及儲存檔案的位置。

```
GetObjectResponse getObjectResponse = s3Client.getObject(GetObjectRequest.builder()
    .key("example-file.txt")
    .bucket("arn:aws:s3::
123456789012:accesspoint/abcdef0123456.mrap")
    .requestPayer(RequestPayer.REQUESTER)
    .build()
).response();
```

監控與記錄透過多區域存取點對基礎資源提出的請求

Amazon S3 會記錄透過多區域存取點和對管理它們的 API 操作提出的請求，例如 CreateMultiRegionAccessPoint 和 GetMultiRegionAccessPointPolicy。透過多區域存取點對 Amazon S3 提出的請求會使用多區域存取點主機名稱顯示在 Amazon S3 伺服器存取日誌和 AWS CloudTrail 日誌中。存取點的主機名稱採用此格式 *MRAP_alias*.accesspoint.s3-global.amazonaws.com。例如，假設您有下列儲存貯體和多區域存取點組態：

- 區域 us-west-2 中名為 my-bucket-usw2 的儲存貯體，其中包含物件 my-image.jpg。
- 區域 ap-south-1 中名為 my-bucket-aps1 的儲存貯體，其中包含物件 my-image.jpg。
- 區域 eu-central-1 中名為 my-bucket-euc1 的儲存貯體，其中不包含名為 my-image.jpg 的物件。
- 名為 my-mrap 且別名為 mfzwi23gnjvgw.mrap 的多區域存取點被設定為可滿足來自所有三個儲存貯體的請求。
- 您的 AWS 帳戶 ID 是 123456789012。

透過任何儲存貯體直接擷取 my-image.jpg 的請求會顯示在您的日誌中，其主機名稱為 *bucket_name*.s3.*Region*.amazonaws.com。

如果您改為透過多區域存取點提出請求，Amazon S3 會先判定不同區域中的哪個儲存貯體最接近。在 Amazon S3 判定要使用哪個儲存貯體來完成請求後，它會將請求傳送到該儲存貯體，並使用多區域存取點主機名稱記錄操作。在此範例中，如果 Amazon S3 將請求轉送到 `my-bucket-aps1`，您的日誌會反映來自 `my-bucket-aps1` 且針對 `my-image.jpg` 的成功 `GET` 請求，其中使用了 `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com` 的主機名稱。

> **Important**
>
> 多區域存取點不知道基礎儲存貯體的資料內容。因此，取得請求的儲存貯體可能不包含請求的資料。例如，如果 Amazon S3 判定 `my-bucket-euc1` 儲存貯體最接近，您的日誌將反映來自 `my-bucket-euc1` 且針對 `my-image.jpg` 的失敗 `GET` 請求，其中使用了 `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com` 的主機名稱。如果請求被路由到 `my-bucket-usw2`，您的日誌將表示一個成功的 `GET` 請求。

如需 Amazon S3 伺服器存取日誌的詳細資訊，請參閱「使用伺服器存取記錄記錄要求」。如需 AWS CloudTrail 的詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的什麼是 AWS CloudTrail？。

監控與記錄對多區域存取點管理 API 操作提出的請求

Amazon S3 提供數個可管理多區域存取點的 API 操作，例如 `CreateMultiRegionAccessPoint` 和 `GetMultiRegionAccessPointPolicy`。當您使用 AWS Command Line Interface (AWS CLI)、AWS 開發套件或 Amazon S3 REST API 對這些 API 操作提出請求時，Amazon S3 會以非同步方式處理這些請求。如果您擁有請求的適當許可，Amazon S3 會傳回這些請求的字符。您可以搭配使用此字符與 `DescribeAsyncOperation`，從而協助您檢視正在進行的非同步操作狀態。Amazon S3 會同步處理 `DescribeAsyncOperation` 請求。您可以使用 Amazon S3 主控台、AWS CLI、開發套件或 REST API 檢視非同步請求的狀態。

> **Note**
>
> 主控台只會顯示過去 14 天內提出的非同步請求的狀態。若要檢視較舊的請求的狀態，請使用 AWS CLI、開發套件或 REST API。

非同步管理操作可能處於以下幾種狀態的其中一種：

NEW

Amazon S3 已收到請求並正準備執行操作。

IN_PROGRESS

Amazon S3 目前正在執行操作。

SUCCESS

操作成功。回應包含相關資訊，例如 CreateMultiRegionAccessPoint 請求的多區域存取點別名。

FAILED

操作失敗。回應包含錯誤訊息，其中會表示請求失敗的原因。

使用具有多區域存取點的 AWS CloudTrail

您可以使用 AWS CloudTrail 檢視、搜尋、下載、封存、分析和回應您 AWS 基礎設施的帳戶活動。透過多區域存取點和 CloudTrail 日誌記錄，您可以識別下列項目：

- 誰採取了哪些行動
- 針對哪些資源採取行動
- 事件發生時間
- 有關事件的其他詳細資訊

您可以使用此日誌記錄資訊，協助分析和回應透過多區域存取點發生的活動。

如何設定多區域存取點的 AWS CloudTrail

若要針對建立或維護多區域存取點的任何操作啟用 CloudTrail 記錄，您必須設定 CloudTrail 記錄，以便記錄美國西部 (奧勒岡) 區域中的事件。無論您在提出請求時位於哪個區域，或多區域存取點支援哪些區域，您皆必須以此方式設定日誌紀錄組態。建立或維護多區域存取點的所有請求皆會透過美國西部 (奧勒岡) 區域路由。建議您將此區域新增至現有線索，或建立一個包含此區域和與多區域存取點關聯的所有區域的新線索。

Amazon S3 會記錄透過多區域存取點提出的請求以及對管理存取點的 API 操作提出的請求，例如 CreateMultiRegionAccessPoint 和 GetMultiRegionAccessPointPolicy。當您透過多區域存取點記錄這些請求時，它們會顯示在您的 AWS CloudTrail 記錄中，其中包含多區域存取點的主機名稱。例如，如果您透過具有別名 mfzwi23gnjvgw.mrap 的多區域存取點對儲存貯體提出請求，則 CloudTrail 日誌中的項目將具有 mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com 的主機名稱。

多區域存取點會將請求路由至最近的儲存貯體，因此，當您查看多區域存取點的 CloudTrail 日誌時，您會看到基礎儲存貯體提出的請求。其中一些請求可能是直接向儲存貯體提出的請求，而不是透過多區域存取點路由的請求。檢視流量時，請務必牢記這點。當儲存貯體位於多區域存取點時，仍然可以直接對該儲存貯體提出請求，而無需透過多區域存取點。

建立和管理多區域存取點時涉及非同步事件。非同步請求在 CloudTrail 日誌中沒有完成事件。如需非同步請求的詳細資訊，請參閱 監控與記錄對多區域存取點管理 API 操作提出的請求。

如需 AWS CloudTrail 的詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的什麼是 AWS CloudTrail？。

Amazon S3 安全性

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，這些架構是為了滿足對安全性最敏感的組織的需求而打造的。

安全是 AWS 與您之間共同承擔的責任。共同責任模型 將此描述為雲端的安全和雲端內的安全：

雲端本身的安全

AWS 負責保護中執行 AWS 服務的基礎結構 AWS 雲端。 AWS 還為您提供可以安全使用的服務。第三方稽核人員定期檢測及驗證安全的效率也是我們 AWS 合規計劃的一部分。若要了解適用於 Amazon S3 的合規計劃，請參閱合規計劃範圍內的AWS 服務。

雲端內部的安全

您的責任取決於您使用的 AWS 服務。您也必須對資料敏感度、組織要求，以及適用法律和法規等其他因素負責。對於 Amazon S3，您的責任包含下列領域：

- 管理您的資料，包括物件所有權和加密。
- 將您的資產分類。
- 使用 IAM 角色和其他服務組態對資料進行管理存取，以套用適當的許可。
- 啟用 Amazon S3 GuardDuty 的偵探控制，例如AWS CloudTrail或亞馬遜。

本文件協助您了解如何在使用 Amazon S3 時套用共同責任模型。下列主題說明如何將 Amazon S3 設定為符合您的安全與合規目標。您也將學習如何使用其他可 AWS 協助您監控和保護 Amazon S3 資源的服務。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱什麼是 S3 Express One Zone？ 和 目錄值區。

主題

- Amazon S3 的資料保護
- 使用加密來保護資料

- 網際網路流量隱私權
- AWS PrivateLink 適用於 Amazon S3
- 存取管理
- 使用跨來源資源分享 (CORS)
- 在 Amazon S3 中記錄和監控
- Amazon S3 的合規驗證
- Amazon S3 的恢復能力
- Amazon S3 的基礎設施安全性
- Amazon S3 中的組態與漏洞分析
- Amazon S3 的安全最佳實務
- 使用託管安全服務監控數據 AWS 安全

Amazon S3 的資料保護

Amazon S3 提供高耐用性儲存基礎設施，專為關鍵任務及主要資料儲存體所設計。S3 Standard、S3 Intelligent-Tiering、S3 標準 – IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 會將物件重複存放在 AWS 區域中至少三個可用區域的多個裝置上。可用區域是一個或多個獨立的資料中心，具備 AWS 區域中的備援電源、聯網和連線能力。可用區域與任何其他可用區域之間都存在有意義的實際距離 (數公里) 分隔，儘管所有可用區域相互距離都在 100 公里 (60 英里) 之內。S3 單區域 – IA 儲存類別會在單一可用區域內，跨多個裝置重複存放資料。這些服務旨在透過快速偵測並修復任何遺失的備援功能來處理並行裝置故障，而且也會定期使用總和檢查碼來驗證資料的完整性。

Amazon S3 Standard 儲存體提供以下功能：

- 受 Amazon S3 服務水準協議支援。
- 設計為保障在特定一年內，提供 99.999999999% 的物件耐用性與 99.99% 的物件可用性。
- S3 Standard、S3 Intelligent-Tiering、S3 標準 – IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 全部設計用於在整個 Amazon S3 可用區域遺失的情況下維持資料。

Amazon S3 進一步使用版本控制來保護您的資料。您可以使用版本控制功能來保留、擷取和恢復在 Amazon S3 儲存貯體中存放的每個物件的每個版本。透過版本控制，您就可以輕鬆地復原失誤的使用

者動作和故障的應用程式。根據預設，要求會擷取最近寫入的版本。您可以在請求中指定物件版本，藉此擷取舊版物件。

除了 S3 版本控制之外，您還可以使用 Amazon S3 物件鎖定和 S3 複寫來保護您的資料。如需詳細資訊，請參閱教學課程：使用 S3 版本控制、S3 物件鎖定和 S3 複寫，保護 Amazon S3 上的資料，以防意外刪除或發生應用程式錯誤。

基於資料保護目的，我們建議您使用保護 AWS 帳戶 認證並設定個別使用者帳戶 AWS Identity and Access Management，以便每位使用者僅獲得履行其工作職責所需的權限。

如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需 FIPS 和 FIPS 端點的詳細資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

以下安全最佳實務也可用來解決 Amazon S3 中的資料保護問題：

- Implement server-side encryption
- Enforce encryption of data in transit
- Consider using Macie with Amazon S3
- Identify and audit all your Amazon S3 buckets
- Monitor Amazon Web Services security advisories

使用加密來保護資料

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

資料保護是指保護往返 Amazon S3 的傳輸中資料，以及存放在 Amazon S3 資料中心內磁碟的靜態資料。您可以使用 Secure Socket Layer/Transport Layer Security (SSL/TLS) 或用戶端加密，保護傳輸中的資料。下列選項皆可讓您保護 Amazon S3 中的靜態資料：

- 伺服器端加密 — Amazon S3 會先加密物件，然後再將物件儲存到 AWS 資料中心的磁碟上，然後在下載物件時解密物件。

  根據預設，所有 Amazon S3 儲存貯體都設定了加密，所有上傳到 S3 儲存貯體的新物件都會在靜態時自動加密。伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 是 Amazon S3 中每個儲存貯體的預設加密組態。若要使用不同類型的加密，您可以指定 S3 PUT 請求中要使用的伺服器端加密類型，也可以在目的地儲存貯體中設定預設加密組態。

  如果您想要在要PUT求中指定不同的加密類型，您可以使用伺服器端加密 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS)、使用金鑰的雙層伺服器端加密 (DSSE-KMS)，或使用客戶提供的金 AWS KMS 鑰 (SSE-C) 進行伺服器端加密。若您想在目的地儲存貯體中設定不同的預設加密組態，您可以使用 SSE-KMS 或 DSSE-KMS。

  如需伺服器端加密的每個選項的詳細資訊，請參閱 使用伺服器端加密保護資料。

  若要設定伺服器端加密，請參閱：

  - 使用 Amazon S3 受管金鑰 (SSE-S3) 指定伺服器端加密
  - 使用 AWS KMS (SSE-KMS) 指定伺服器端加密
  - the section called “指定 DSSE-KMS”
  - 使用客戶提供金鑰 (SSC-C) 指定伺服器端加密

- 用戶端加密 - 在用戶端加密資料，並將加密的資料上傳至 Amazon S3。在這種情況下，您可以管理加密程序、加密金鑰和相關工具。

  若要設定用戶端加密，請參閱 使用用戶端加密保護資料。

若要查看多少百分比的儲存體位元組已加密，您可以使用 Amazon S3 Storage Lens 指標。S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。如需詳細資訊，請參閱使用 S3 Storage Lens 評估儲存活動和用量。如需完整的指標清單，請參閱 S3 Storage Lens 指標詞彙表。

如需伺服器端加密和用戶端加密的相關資訊，請檢閱下列主題。

主題

- 使用伺服器端加密保護資料
- 使用用戶端加密保護資料

使用伺服器端加密保護資料

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

伺服器端加密是指接收資料的應用程式或服務在目的地將資料加密。Amazon S3 會在物件層級將資料寫入資料中心的磁碟時加密，並在您存取 AWS 資料時為您解密。只要您驗證要求並具備存取許可，存取加密物件或未加密物件的方式並無不同。例如，如果您使用預先簽章的 URL 來分享物件，加密物件與未加密物件的 URL 運作方式會相同。此外，當您列出儲存貯體中的物件時，清單 API 操作會傳回所有物件清單，無論其是否經過加密。

根據預設，所有 Amazon S3 儲存貯體都設定了加密，所有上傳到 S3 儲存貯體的新物件都會在靜態時自動加密。伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 是 Amazon S3 中每個儲存貯體的預設加密組態。若要使用不同類型的加密，您可以指定 S3 PUT 請求中要使用的伺服器端加密類型，也可以在目的地儲存貯體中設定預設加密組態。

如果您想要在要PUT求中指定不同的加密類型，您可以使用伺服器端加密 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS)、使用金鑰的雙層伺服器端加密 (DSSE-KMS)，或使用客戶提供的金 AWS KMS 鑰 (SSE-C) 進行伺服器端加密。若您想在目的地儲存貯體中設定不同的預設加密組態，您可以使用 SSE-KMS 或 DSSE-KMS。

> **Note**
>
> 您不可以同時對同一個物件套用不同類型的伺服器端加密。

若您需要加密現有物件，請使用 S3 批次操作和 S3 清查。如需詳細資訊，請參閱使用 Amazon S3 批次操作來加密物件和 在 Amazon S3 物件上執行大規模批次操作。

根據您選擇管理加密金鑰的方式，以及您想套用的加密層級數量，針對伺服器端加密，您只能選擇下列四個選項之一。

使用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密

根據預設，所有 Amazon S3 儲存貯體都設定了加密。伺服器端加密的預設為使用 Amazon S3 受管金鑰 (SSE-S3)。每個物件都使用不重複的金鑰加密。SSE-S3 使用定期輪換的根金鑰自行加密金鑰，提供額外的防護。SSE-S3 使用目前最強大的其中一種區塊加密法 (256 位元進階加密標準 (AES-256))，加密您的資料。如需詳細資訊，請參閱 使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密。

使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 進行伺服器端加密

伺服器端加密 AWS KMS keys (SSE-KMS) 是透過與 Amazon S3 整合的 AWS KMS 服務來提供。使用 AWS KMS，您可以更好地控制您的鑰匙。例如，您可以檢視個別金鑰、編輯控制政策，並遵循AWS CloudTrail中的金鑰。此外，您可以建立和管理客戶受管金鑰，或是使用您、您服務和您區域唯一的 AWS 受管金鑰 。如需詳細資訊，請參閱 使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)。

使用 AWS Key Management Service (AWS KMS) 金鑰 (DSSE-KMS) 進行雙層伺服器端加密

使用 AWS KMS keys (DSSE-KMS) 的雙層伺服器端加密與 SSE-KMS 類似，但是 DSSE-KMS 會套用兩個個別的物件層級加密層，而不是單一層。由於兩層加密都套用至伺服器端的物件，因此您可以使用各種 AWS 服務 和工具來分析 S3 中的資料，同時使用可滿足合規需求的加密方法。如需詳細資訊，請參閱 搭配 AWS KMS 金鑰 (DSSE-KMS) 使用雙層伺服器端加密。

使用客戶提供金鑰 (SSE-C) 的伺服器端加密

使用「伺服器端加密搭配客戶提供金鑰 (SSE-C)」時，您負責管理加密金鑰，而 Amazon S3 會在將物件寫入磁碟時進行加密，並在您存取物件時予以解密。如需詳細資訊，請參閱 搭配客戶提供的金鑰 (SSE-C) 使用伺服器端加密。

Amazon S3 現在會自動加密所有新物件

Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。使用 256 位元進階加密標準 (AES-256) 的 SSE-S3 會自動套用至所有新的儲存貯體，以及套用至任何尚未設定預設加密的現有 S3 儲存貯體。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可用於 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台，以及 AWS Command Line Interface (AWS CLI) 和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。

以下各節回答有關此更新的問題。

Amazon S3 是否會針對已設定預設加密的現有儲存貯體變更預設加密設定？

沒有 對於已設定 SSE-S3 或伺服器端加密且已設定 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS) 的現有儲存貯體，預設加密組態沒有變更。如需如何設定儲存貯體預設加密行為的詳細資訊，請參閱 對 Amazon S3 儲存貯體設定預設伺服器端加密行為。如需 SSE-S3 和 SSE-KMS 加密設定的詳細資訊，請參閱 使用伺服器端加密保護資料。

是否會在未設定預設加密的現有儲存貯體上啟用預設加密？

是。Amazon S3 現在可在所有現有未加密的儲存貯體上設定預設加密，套用伺服器端加密與 S3 受管金鑰 (SSE-S3)，作為上傳至這些儲存貯體之新物件的基本加密層級。已存在於現有未加密儲存貯體中的物件將不會自動加密。

如何檢視新物件上傳的預設加密狀態？

目前，您可以在 AWS CloudTrail 日誌、S3 庫存和 S3 儲存鏡頭、Amazon S3 主控台中檢視新物件上傳的預設加密狀態，以及 AWS Command Line Interface (AWS CLI) 和 AWS 開發套件中的其他 Amazon S3 API 回應標頭。

- 若要檢視您的 CloudTrail 事件，請參閱《AWS CloudTrail 使用指南》中的「在 CloudTrail 主控台中檢視 CloudTrail 事件」。 CloudTrail 日誌可為 Amazon S3 提供 API 追蹤，以PUT及傳POST送給 Amazon S3 的請求。當使用預設加密來加密值區中的物件時， CloudTrail 記錄檔PUT和 POST API 要求將包含下列欄位做為名稱-值配對：`"SSEApplied":"Default_SSE_S3"`
- 若要檢視 S3 清查中新物件上傳的自動加密狀態，請將 S3 清查報告設定為包含 Encryption (加密) 的中繼資料欄位，然後在報告中查看每個新物件的加密狀態。如需詳細資訊，請參閱設定 Amazon S3 清查。
- 若要檢視 S3 Storage Lens 中新物件上傳的自動加密狀態，請設定 S3 Storage Lens 儀表板，並在儀表板的 Data protection (資料保護) 類別中查看 Encrypted bytes (加密位元組) 和 Encrypted object count (加密物件計數) 指標。如需詳細資訊，請參閱 建立 Amazon S3 Storage Lens 儀表板 及 在儀表板上檢視 S3 Storage Lens 指標。
- 若要在 Amazon S3 主控台中檢視儲存貯體層級的自動加密狀態，請在 Amazon S3 主控台中檢查 Amazon S3 儲存貯體的預設加密。如需詳細資訊，請參閱 設定預設加密。
- 若要在 () 和 AWS 開發套件中檢視為其他 Amazon S3 API 回應標頭的自動加密狀態，請在使用物件動作 API AWS Command Line Interface (例如PutObject和AWS CLIGetObject) `x-amz-server-side-encryption` 時檢查回應標頭。

我必須做什麼才能利用此變更？

您不需要對現有的應用程式進行任何變更。因為您的所有儲存貯體都已啟用預設加密，所有上傳到 Amazon S3 的新物件都會自動加密。

是否可以針對寫入至儲存貯體的新物件停用加密？

否。SSE-S3 是新的加密基本層級，適用於所有要上載至儲存貯體的新物件。您再也無法停用新物件上傳的加密。

我的費用是否會受到影響嗎？

否。搭配 SSE-S3 的預設加密可免費使用。我們會照常向您收取儲存、請求和其他 S3 功能的費用。如需定價，請參閱 Amazon S3 定價。

Amazon S3 是否會加密現有未加密的物件？

否。從 2023 年 1 月 5 日開始，Amazon S3 只會自動加密新物件上傳。若要加密現有物件，您可以使用 S3 Batch Operations 來建立物件的加密複本。這些加密複本將保留現有的物件資料和名稱，並將使用您指定的加密金鑰加密。如需詳細資訊，請參閱《AWS 儲存體部落格》中的使用 Amazon S3 Batch Operations 加密物件。

在此版本之前，我並未針對儲存貯體啟用加密。我是否需要變更存取物件的方式？

否。搭配 SSE-S3 的預設加密會在資料寫入 Amazon S3 時自動加密該資料，並在您存取該資料時為您將其解密。您存取自動加密物件的方式沒有變更。

我是否需要變更存取用戶端加密物件的方式？

否。上傳到 Amazon S3 之前已加密的所有用戶端加密物件都會在 Amazon S3 內以加密的密文物件形式送達。這些物件現在將會有另一層 SSE-S3 加密。使用用戶端加密物件的工作負載不需要對用戶端服務或授權設定進行任何變更。

> **Note**
>
> HashiCorp 在建立沒有客戶定義加密組態的新 S3 儲存貯體後，未使用 AWS 提供者更新版本的 Terraform 使用者可能會看到意外的漂移。若要避免這種偏差，請將您的 Terraform AWS 提供者版本更新為下列其中一個版本：任何4.x發行版本3.76.1、或。2.70.4

使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會

自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

所有上傳到 Amazon S3 儲存貯體的新物件均會使用伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 進行加密。

伺服器端加密保護靜態資料。Amazon S3 會使用不重複的金鑰加密每個物件。它使用定期輪換的金鑰自行加密金鑰，提供額外的防護。Amazon S3 伺服器端加密使用 256 位元 Galois/計數器模式中的進階加密標準 (AES-GCM) 來加密所有上傳的物件。

將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 搭配使用不需另外付費。不過，請求設定預設加密功能會產生標準 Amazon S3 請求費用。如需定價的資訊，請參閱 Amazon S3 定價。

若您希望上傳的資料僅使用 Amazon S3 受管金鑰加密，您可以使用下列儲存貯體政策。例如，除非要求包含 `x-amz-server-side-encryption`標頭以要求伺服器端加密，否則以下儲存貯體政策會拒絕上傳物件的許可權限：

```
{
  "Version": "2012-10-17",
  "Id": "PutObjectPolicy",
  "Statement": [
    {
      "Sid": "DenyObjectsThatAreNotSSES3",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::example-s3-bucket/*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption": "AES256"
        }
      }
    }
   ]
}
```

> **Note**
>
> 伺服器端加密只會加密物件資料，非物件中繼資料。

伺服器端加密的 API 支援

根據預設，所有 Amazon S3 儲存貯體都設定了加密，所有上傳到 S3 儲存貯體的新物件都會在靜態時自動加密。伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 是 Amazon S3 中每個儲存貯體的預設加密組態。若要使用不同類型的加密，您可以指定 S3 PUT 請求中要使用的伺服器端加密類型，也可以在目的地儲存貯體中設定預設加密組態。

如果您想要在要PUT求中指定不同的加密類型，您可以使用伺服器端加密 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS)、使用金鑰的雙層伺服器端加密 (DSSE-KMS)，或使用客戶提供的金 AWS KMS 鑰 (SSE-C) 進行伺服器端加密。若您想在目的地儲存貯體中設定不同的預設加密組態，您可以使用 SSE-KMS 或 DSSE-KMS。

若要使用物件建立 REST API，用以設定伺服器端加密，請務必提供 `x-amz-server-side-encryption` 要求標頭。如需 REST APIs 的相關資訊，請參閱使用 REST API。

下列 Amazon S3 API 支援此標頭。

- PUT 操作 — 使用 PUT API 上傳資料時，請指定要求標頭。如需詳細資訊，請參閱 PUT 物件。
- 啟動分段上傳 — 使用分段上傳 API 操作上傳大型物件時，您可於起始要求時，指定這些標頭。如需詳細資訊，請參閱啟動分段上傳。
- COPY 操作 — 當您複製物件時，要有來源物件與目標物件。如需詳細資訊，請參閱 PUT 物件 - 複製。

> **Note**
>
> 使用 POST 操作上傳物件時，請您提供與表單欄位中相同的資訊，而非提供要求標頭。如需詳細資訊，請參閱 POST 物件。

AWS SDK 也提供包裝 API，您可以用來要求伺服器端加密。您也可以使用上 AWS Management Console 載物件並要求伺服器端加密。

如需更多一般資訊，請參閱《AWS Key Management Service 開發人員指南》中的 AWS KMS 概念。

主題

- 使用 Amazon S3 受管金鑰 (SSE-S3) 指定伺服器端加密

使用 Amazon S3 受管金鑰 (SSE-S3) 指定伺服器端加密

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

根據預設，所有 Amazon S3 儲存貯體都設定了加密，所有上傳到 S3 儲存貯體的新物件都會在靜態時自動加密。伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 是 Amazon S3 中每個儲存貯體的預設加密組態。若要使用不同類型的加密，您可以指定 S3 PUT 請求中要使用的伺服器端加密類型，也可以在目的地儲存貯體中設定預設加密組態。

如果您想要在要PUT求中指定不同的加密類型，您可以使用伺服器端加密 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS)、使用金鑰的雙層伺服器端加密 (DSSE-KMS)，或使用客戶提供的金 AWS KMS 鑰 (SSE-C) 進行伺服器端加密。若您想在目的地儲存貯體中設定不同的預設加密組態，您可以使用 SSE-KMS 或 DSSE-KMS。

您可以使用 S3 主控台、REST API、 AWS 開發套件和 AWS Command Line Interface (AWS CLI) 來指定 SSE-S3。如需詳細資訊，請參閱「對 Amazon S3 儲存貯體設定預設伺服器端加密行為」。

使用 S3 主控台

本主題說明如何使用 AWS Management Console設定或變更物件所使用的加密類型。當您使用主控台複製物件時，Amazon S3 會依原樣複製物件。這表示，若來源物件已加密，則目標物件也會加密。您可以使用主控台來新增或變更物件的加密。

> **Note**
>
> - 如果您變更物件的加密，則會建立新物件來取代舊物件。如果啟用 S3 版本控制，則系統會建立物件的新版本，且現有物件會變成較舊的版本。變更屬性的角色也會成為新物件 (或物件版本) 的擁有者。
> - 如果您變更具有使用者定義標籤之物件的加密類型，您必須擁有`s3:GetObjectTagging`權限。如果您要變更沒有使用者定義標籤但大小超過 16 MB 的物件的加密類型，您也必須擁有`s3:GetObjectTagging`權限。
>
> 如果目的地儲存貯體政策拒絕該`s3:GetObjectTagging`動作，則會更新物件的加密類型，但使用者定義的標籤會從物件中移除，而且您會收到錯誤訊息。

變更物件的加密

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇包含該物件的儲存貯體名稱。
4. 在 Object (物件) 清單中，選擇您希望新增或變更加密的物件名稱。

   系統會顯示物件的詳細資訊頁面，其中包含數個區段，顯示物件的屬性。
5. 選擇屬性索引標籤。
6. 向下捲動至伺服器端加密設定區段，然後選擇編輯。
7. 在加密設定底下，選擇使用儲存貯體預設加密設定或覆寫儲存貯體預設加密設定。
8. 若您選擇覆寫預設加密的儲存貯體設定，請設定下列加密設定。
   - 在加密類型之下，選擇 Amazon S3 受管金鑰 (SSE-S3)。SSE-S3 使用目前最強大的其中一種區塊加密法，也就是 256 位元進階加密標準 (AES-256)，來加密每個物件。如需詳細資訊，請參閱 使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密。
9. 選擇 Save changes (儲存變更)。

> **Note**
>
> 此動作會將加密套用至所有指定的物件。加密資料夾時，請等待儲存作業完成，然後再將新物件新增至資料夾。

使用 REST API

建立物件時 (也就是當您上傳新的物件或複製現有物件時)，可以指定是否在請求中新增 `x-amz-server-side-encryption` 標頭，讓 Amazon S3 以 Amazon S3 受管金鑰 (SSE-S3) 加密您的資料。將標頭值設為 Amazon S3 支援的加密演算法 `AES256`。Amazon S3 會傳回回應標頭 `x-amz-server-side-encryption`，確認已使用 SSE-S3 存放物件。

下列 REST 上傳 API 操作，接受 `x-amz-server-side-encryption` 要求標頭。

- PUT 物件
- PUT 物件 - 複製
- POST 物件
- 啟動分段上傳

使用分段上傳 API 操作上傳大型物件時，可以對啟動分段上傳要求新增 `x-amz-server-side-encryption` 標頭，指定伺服器端加密。複製現有物件時，除非明確地要求伺服器端加密，否則無論來源物件是否經過加密，都不會加密目標物件。

使用 SSE-S3 存放物件時，下列 REST API 操作的回應標頭會傳回 `x-amz-server-side-encryption` 標頭。

- PUT 物件
- PUT 物件 - 複製
- POST 物件
- 啟動分段上傳
- 上傳片段
- 上傳片段 - 複製
- 完成分段上傳
- Get 物件

- Head 物件

> **Note**
>
> 如果物件使用 SSE-S3，請勿為 GET 請求與 HEAD 請求傳送加密請求標頭，否則您會收到 HTTP 狀態碼 400 (錯誤的請求) 錯誤。

使用 AWS 軟體開發套件

使用 AWS 開發套件時，您可以請求 Amazon S3 將伺服器端加密與 Amazon S3 受管加密金鑰搭配使用 (SSE-S3)。本節提供以多種語言使用 AWS SDK 的範例。如需其他 SDK 的資訊，請前往範例程式碼與程式庫。

Java

當您使用上 AWS SDK for Java 載物件時，您可以使用 SSE-S3 加密物件。若要求伺服器端加密，可使用 `ObjectMetadata`，`PutObjectRequest`的屬性，設定 `x-amz-server-side-encryption` 要求標頭。當您呼叫 `AmazonS3Client` 的，`putObject()` 方法時，Amazon S3 會加密和儲存資料。

使用分段上傳 API 操作時，您也可以要求 SSE-S3 加密。

- 使用高階分段上傳 API 操作，您可以使用 `TransferManager` 方法，應用於上傳伺服器端物件加密。您可使用任一接受 `ObjectMetadata` 為參數的上傳方法。如需詳細資訊，請參閱 使用分段上傳來上傳物件。
- 使用低階分段上傳 API 操作，可以在啟動分段上傳時，指定伺服器端加密。呼叫 `ObjectMetadata` 的方法，新增 `InitiateMultipartUploadRequest.setObjectMetadata()` 屬性。如需詳細資訊，請參閱 使用 AWS 軟體開發套件 (低階 API)。

您無法直接更改物件的加密狀態 (加密未加密的物件或解密加密的物件)。要更改物件的加密狀態，請複製該物件，指定所需複本的加密狀態，然後刪除原始物件。只有在您明確請求伺服器端加密時，Amazon S3 才會加密複製的物件。使用 `ObjectMetadata` 屬性，在 `CopyObjectRequest` 中透過 API 指定伺服器端加密，要求進行物件複本加密。

Example 範例

下列範例示範如何使用 AWS SDK for Java設定伺服器端加密。說明如何執行以下任務：

- 使用 SSE-S3 上傳新物件。
- 透過製作物件複本，更改物件的加密狀態 (在此範例中，加密先前未加密過的物件)。
- 確認物件加密狀態。

如需伺服器端加密的詳細資訊，請參閱「使用 REST API」。如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.internal.SSEResultBase;
import com.amazonaws.services.s3.model.*;

import java.io.ByteArrayInputStream;

public class SpecifyServerSideEncryption {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyNameToEncrypt = "*** Key name for an object to upload and encrypt
 ***";
        String keyNameToCopyAndEncrypt = "*** Key name for an unencrypted object to
 be encrypted by copying ***";
        String copiedObjectKeyName = "*** Key name for the encrypted copy of the
 unencrypted object ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Upload an object and encrypt it with SSE.
            uploadObjectWithSSEEncryption(s3Client, bucketName, keyNameToEncrypt);

            // Upload a new unencrypted object, then change its encryption state
            // to encrypted by making a copy.
```

```
            changeSSEEncryptionStatusByCopying(s3Client,
                    bucketName,
                    keyNameToCopyAndEncrypt,
                    copiedObjectKeyName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    private static void uploadObjectWithSSEEncryption(AmazonS3 s3Client, String
bucketName, String keyName) {
        String objectContent = "Test object encrypted with SSE";
        byte[] objectBytes = objectContent.getBytes();

        // Specify server-side encryption.
        ObjectMetadata objectMetadata = new ObjectMetadata();
        objectMetadata.setContentLength(objectBytes.length);

objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
        PutObjectRequest putRequest = new PutObjectRequest(bucketName,
                keyName,
                new ByteArrayInputStream(objectBytes),
                objectMetadata);

        // Upload the object and check its encryption status.
        PutObjectResult putResult = s3Client.putObject(putRequest);
        System.out.println("Object \"" + keyName + "\" uploaded with SSE.");
        printEncryptionStatus(putResult);
    }

    private static void changeSSEEncryptionStatusByCopying(AmazonS3 s3Client,
            String bucketName,
            String sourceKey,
            String destKey) {
        // Upload a new, unencrypted object.
        PutObjectResult putResult = s3Client.putObject(bucketName, sourceKey,
"Object example to encrypt by copying");
        System.out.println("Unencrypted object \"" + sourceKey + "\" uploaded.");
```

```
        printEncryptionStatus(putResult);

        // Make a copy of the object and use server-side encryption when storing the
        // copy.
        CopyObjectRequest request = new CopyObjectRequest(bucketName,
                sourceKey,
                bucketName,
                destKey);
        ObjectMetadata objectMetadata = new ObjectMetadata();

 objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
        request.setNewObjectMetadata(objectMetadata);

        // Perform the copy operation and display the copy's encryption status.
        CopyObjectResult response = s3Client.copyObject(request);
        System.out.println("Object \"" + destKey + "\" uploaded with SSE.");
        printEncryptionStatus(response);

        // Delete the original, unencrypted object, leaving only the encrypted copy
 in
        // Amazon S3.
        s3Client.deleteObject(bucketName, sourceKey);
        System.out.println("Unencrypted object \"" + sourceKey + "\" deleted.");
    }

    private static void printEncryptionStatus(SSEResultBase response) {
        String encryptionStatus = response.getSSEAlgorithm();
        if (encryptionStatus == null) {
            encryptionStatus = "Not encrypted with SSE";
        }
        System.out.println("Object encryption status is: " + encryptionStatus);
    }
}
```

.NET

當您上傳物件時，可指示 Amazon S3 加密物件。若要變更現有物件的加密狀態，請複製該物件並刪除來源物件。依預設值，除非您明確地要求對目標物件進行伺服器端加密，否則複製作業不會加密目標。若要在 CopyObjectRequest 中指定 SSE-S3，請新增下列項目：

```
ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256
```

如需如何複製物件的工作範例，請參閱「使用 AWS 軟體開發套件」。

下列範例會上傳一個物件。在請求中，此範例會指示 Amazon S3 加密物件。然後，示範擷取物件中繼資料，驗證使用的加密方法。如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SpecifyServerSideEncryptionTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for object created ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            WritingAnObjectAsync().Wait();
        }

        static async Task WritingAnObjectAsync()
        {
            try
            {
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    ContentBody = "sample text",
                    ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256
                };

                var putResponse = await client.PutObjectAsync(putRequest);
```

```
                // Determine the encryption state of an object.
                GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };
                GetObjectMetadataResponse response = await
 client.GetObjectMetadataAsync(metadataRequest);
                ServerSideEncryptionMethod objectEncryption =
 response.ServerSideEncryptionMethod;

                Console.WriteLine("Encryption method used: {0}",
 objectEncryption.ToString());
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
 an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

本主題說明如何使用第 3 版的類別，將 SSE-S3 新增 AWS SDK for PHP 至您上傳到 Amazon S3 的物件。有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

若要將物件上傳至 Amazon S3，請使用 Aws\S3\S3Client::putObject() 方法。若要將 `x-amz-server-side-encryption` 請求標頭新增至上傳請求，則需使用 `AES256` 值來指定 `ServerSideEncryption` 參數，如以下程式碼範例所示。如需伺服器端加密要求的詳細資訊，請參閱使用 REST API。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;
```

```
$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

// $filepath should be an absolute path to a file on disk.
$filepath = '*** Your File Path ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Upload a file with server-side encryption.
$result = $s3->putObject([
    'Bucket'               => $bucket,
    'Key'                  => $keyname,
    'SourceFile'           => $filepath,
    'ServerSideEncryption' => 'AES256',
]);
```

在回應時，Amazon S3 會傳回 `x-amz-server-side-encryption` 標頭，值為用來加密物件資料的加密演算法。

當您使用分段上傳 API 操作上傳大型物件時，可以為要上傳的物件指定 SSE-S3，如下所示：

- 當您使用低階多部分上傳 API 作業時，請在呼叫 AWS\ S3\ S3Client:: createMultipartUpload () 方法時指定伺服器端加密。若要將 `x-amz-server-side-encryption` 請求標頭新增至請求，則需使用 `AES256` 值來指定 `array` 參數的 `ServerSideEncryption` 金鑰。如需低階分段上傳 API 操作的詳細資訊，請參閱 使用 AWS 軟體開發套件 (低階 API)。
- 當您使用高階分段上傳 API 作業時，請使用 CreateMultipartUploadAPI 作業的`ServerSideEncryption`參數指定伺服器端加密。如需使用 `setOption()` 方法搭配高階分段上傳 API 操作的範例，請參閱 使用分段上傳來上傳物件。

若要判斷現有物件的加密狀態，請呼叫 Aws\S3\S3Client::headObject() 方法來擷取物件中繼資料，如下列 PHP 程式碼範例所示。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
```

```
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Check which server-side encryption algorithm is used.
$result = $s3->headObject([
    'Bucket' => $bucket,
    'Key'    => $keyname,
]);
echo $result['ServerSideEncryption'];
```

若要變更現有物件的加密狀態，請使用 Aws\S3\S3Client::copyObject() 方法建立物件複本，並刪除原始物件。除非您使用值為 AES256 的 ServerSideEncryption 參數來明確請求目的地物件的伺服器端加密，否則 copyObject() 預設不會加密目標。下列 PHP 程式碼範例會建立物件複本，並將伺服器端加密新增至複製的物件。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$sourceBucket = '*** Your Source Bucket Name ***';
$sourceKeyname = '*** Your Source Object Key ***';

$targetBucket = '*** Your Target Bucket Name ***';
$targetKeyname = '*** Your Target Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Copy an object and add server-side encryption.
$s3->copyObject([
    'Bucket'               => $targetBucket,
    'Key'                  => $targetKeyname,
    'CopySource'           => "$sourceBucket/$sourceKeyname",
    'ServerSideEncryption' => 'AES256',
]);
```

如需詳細資訊，請參閱下列主題：

- AWS SDK for PHP 對於 Amazon S3 AWS\ S3\ S3 客戶端類
- AWS SDK for PHP 文件

Ruby

使用上 AWS SDK for Ruby 載物件時，您可以指定使用 SSE-S3 加密儲存物件。當您讀回物件時，會自動解密。

以下第 3 AWS SDK for Ruby 版範例示範如何指定上傳到 Amazon S3 的檔案靜態加密。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"
```

```
  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

下列程式碼範例示範如何判斷現有物件的加密狀態。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data
```

```
  encryption = obj_data.server_side_encryption.nil? ? "no" :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

若存放在 Amazon S3 的物件未使用伺服器端加密，該方法會傳回 null。

若要變更現有物件的加密狀態，請複製物件並刪除來源物件。根據預設，除非明確地要求進行伺服器端加密，否則複製方法依預設不會加密目標。您可以在雜湊引數選項中指定 server_side_encryption 值，以此請求加密目標物件，如下列 Ruby 程式碼範例所示。該程式碼範例示範如何以 SSE-S3 複製物件及加密複本。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the target
 key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
```

```
  end
end

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key, target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

使用 AWS CLI

若要在使用上載物件時指定 SSE-S3 AWS CLI，請使用下列範例。

```
aws s3api put-object --bucket example-s3-bucket1 --key object-key-name --server-side-
encryption AES256  --body file path
```

如需詳細資訊，請參閱 AWS CLI 參考中的 put-object。若要在使用複製物件時指定 SSE-S3 AWS CLI，請參閱複製物件。

使用 AWS CloudFormation

如需設定加密方式的範例 AWS CloudFormation，請參閱使用AWS CloudFormation 者指南`AWS::S3::Bucket ServerSideEncryptionRule`主題中的使用預設加密建立儲存貯體和使用 AWS KMS 伺服器端加密搭配 S3 儲存貯體金鑰建立儲存貯體。

使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

伺服器端加密是指接收資料的應用程式或服務在目的地將資料加密。

Amazon S3 會自動為新上傳的物件，以 Amazon S3 受管金鑰 (SSE-S3) 啟用伺服器端加密。

除非您另行指定，否則根據預設，儲存貯體會使用 SSE-S3 來加密物件。不過，您可以選擇設定值區，改為使用 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS) 的伺服器端加密。如需詳細資訊，請參閱 使用 AWS KMS (SSE-KMS) 指定伺服器端加密。

AWS KMS 是一種結合安全、高可用性硬體和軟體的服務，提供專為雲端調整規模的金鑰管理系統。Amazon S3 使用伺服器端加密搭配 AWS KMS (SSE-KMS) 來加密您的 S3 物件資料。此外，當對物件要求 SSE-KMS 時，S3 總和檢查碼 (做為物件中繼資料的一部分) 會以加密形式儲存。如需總和檢查的詳細資訊，請參閱 檢查物件完整性。

如果您使用 KMS 金鑰，您可以 AWS KMS 透過AWS Management Console或 AWS KMS API 使用來執行下列動作：

- 集中建立、檢視、編輯、監控、啟用或停用、輪換和排程 KMS 金鑰的刪除。
- 定義可控制 KMS 金鑰使用方式和使用者的政策。
- 稽核其使用情況，以證明其使用方式正確無誤。AWS KMS API 支援稽核，但 AWS KMSAWS Management Console 不支援。

中的安全控制 AWS KMS 可協助您符合加密相關的合規性要求。您可以使用這些 KMS 金鑰來保護 Amazon S3 儲存貯體中的資料。當您搭配 S3 儲存貯體使用 SSE-KMS 加密時，該儲存貯體 AWS KMS keys 必須位於與儲存貯體相同的區域。

使用需支付額外費用 AWS KMS keys。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的 AWS KMS key 概念和 AWS KMS 定價。

許可

若要將使用加密的物件上傳 AWS KMS key 到 Amazon S3，您需要金鑰的`kms:GenerateDataKey`許可。若要下載使用加密的物件 AWS KMS key，您需要`kms:Decrypt`權限。如需有關分段上傳所需 AWS KMS 權限的資訊，請參閱分段上傳 API 與許可。

> **Important**
>
> 請仔細檢閱 KMS 金鑰原則中授與的權限。一律將客戶管理的 KMS 金鑰政策許可限制為必須存取相關金 AWS KMS 鑰動作的 IAM 主體和 AWS 服務。如需詳細資訊，請參閱中的主要原則 AWS KMS。

主題

- AWS KMS keys
- Amazon S3 儲存貯體金鑰
- 要求伺服器端加密
- 加密內容
- 傳送 AWS KMS 加密物件的要求
- 使用 AWS KMS (SSE-KMS) 指定伺服器端加密
- 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本

AWS KMS keys

將伺服器端加密與 AWS KMS (SSE-KMS) 搭配使用時，您可以使用預設的AWS 受管金鑰，也可以指定您已建立的客戶受管金鑰。 AWS KMS 支持信封加密。S3 使用包絡加密 AWS KMS 功能來進一步保護您的資料。封套加密是使用資料金鑰來加密純文字資料，然後再透過另一個 KMS 金鑰來加密資料金鑰的實務做法。如需封套加密的詳細資訊，請參閱 AWS Key Management Service 開發人員指南中的封套加密。

如果您未指定客戶受管金鑰，Amazon S3 會 AWS 受管金鑰 在您 AWS 帳戶 第一次將使用 SSE-KMS 加密的物件新增至儲存貯體時自動建立。根據預設，Amazon S3 會將此 KMS 金鑰用於 SSE-KMS。

> **Note**
>
> 使用 SSE-KMS 搭配 AWS 受管金鑰 加密的物件無法跨帳戶共用。如果您需要跨帳戶共用 SSE-KMS 資料，則必須使用來自的客戶管理金鑰。 AWS KMS

如果您想要針對 SSE-KMS 使用客戶受管金鑰，請在設定 SSE-KMS 之前建立對稱加密客戶受管金鑰。然後，當您為儲存貯體設定 SSE-KMS 時，請指定現有客戶受管金鑰。如需對稱加密的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的對稱加密 KMS 金鑰。

建立客戶受管金鑰可為您提供更多的靈活性與控制。例如，您可以建立、輪換和停用客戶受管金鑰。您也可以定義存取控制，並稽核用來保護資料的客戶受管金鑰。如需有關客戶受管和受 AWS 管金鑰的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的客戶 AWS 金鑰和金鑰。

> **Note**
>
> 當您搭配存放在外部金鑰存放區中的客戶受管金鑰使用伺服器端加密時，與標準 KMS 金鑰不同，您有責任確保金鑰材料的可用性和耐久性。如需外部金鑰存放區及其如何轉移共用責任模型的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的外部金鑰存放區。

針對跨帳戶操作使用 SSE-KMS 加密

對跨帳戶操作使用加密時，請注意下列事項：

- 如果在請求時或透過儲存貯體的預設加密組態未提供 AWS KMS key Amazon 資源名稱 (ARN) 或別名，則會使用 AWS 受管金鑰 (`aws/s3`)。
- 如果您使用與 KMS 金鑰相同 AWS 帳戶 的 AWS Identity and Access Management (IAM) 主體上傳或存取 S3 物件，則可以使用 AWS 受管金鑰 (`aws/s3`)。
- 如果您想要授予 S3 物件跨帳戶存取權，請使用客戶受管金鑰。您可以設定客戶受管金鑰的政策，以允許從另一個帳戶存取的權限。
- 如果您要指定客戶受管 KMS 金鑰，建議您使用完整的 KMS 金鑰 ARN。如果您改用 KMS 金鑰別名，請 AWS KMS 解析要求者帳戶內的金鑰。此行為可能會導致資料使用屬於申請者 (而不是儲存貯體擁有者) 的 KMS 金鑰來加密。
- 您必須指定已授予您 (要求者) 已獲授予 `Encrypt` 許可的金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的允許金鑰使用者在密碼編譯操作中使用 KMS 金鑰。

如需何時使用客戶受管金鑰和受 AWS 管 KMS 金鑰的詳細資訊，請參閱我應該使用 AWS 受管金鑰 還是客戶受管金鑰來加密 Amazon S3 中的物件？

SSE-KMS 加密工作流程

如果您選擇使用 AWS 受管金鑰 或客戶受管金鑰加密資料 AWS KMS ，Amazon S3 會執行下列信封加密動作：

1. Amazon S3 請求純文字資料金鑰和在指定的 KMS 金鑰下加密的金鑰複本。
2. AWS KMS 產生資料金鑰，在 KMS 金鑰下加密，然後將純文字資料金鑰和加密的資料金鑰傳送到 Amazon S3。
3. Amazon S3 使用資料金鑰來加密資料，並在使用後盡快從記憶體中移除純文字金鑰。
4. Amazon S3 將加密的資料金鑰以中繼資料形式跟加密資料一起存放。

當您要求解密資料時，Amazon S3 並 AWS KMS 執行下列動作：

1. Amazon S3 會 AWS KMS 在`Decrypt`請求中將加密的資料金鑰傳送到。
2. AWS KMS 使用相同的 KMS 金鑰解密加密的資料金鑰，然後將純文字資料金鑰傳回 Amazon S3。
3. Amazon S3 使用純文字資料金鑰來解密加密的資料，然後盡快從記憶體移除純文字資料金鑰。

> **Important**
>
> 當您在 Amazon S3 中使 AWS KMS key 用伺服器端加密時，您必須選擇對稱加密 KMS 金鑰。Amazon S3 只支援對稱加密 KMS 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的對稱加密 KMS 金鑰。

稽核 SS-KMS 加密

若要識別指定 SSE-KMS 的請求，您可以在 Amazon S3 Storage Lens 指標中使用所有 SSE-KMS 請求和 % 所有 SSE-KMS 請求指標。S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。您也可以使用啟用 SSE-KMS 的儲存貯體計數和啟用 SSE-KMS 的% 值區，以瞭解預設值區加密的儲存貯體 (SSE-KMS) 計數。如需詳細資訊，請參閱使用 S3 Storage Lens 評估儲存活動和用量。如需完整的指標清單，請參閱 S3 Storage Lens 指標詞彙表。

若要稽核 SSE-KMS 加密資料金 AWS KMS 鑰的使用情況，您可以使用 AWS CloudTrail 記錄檔。您可以深入瞭解您的密碼編譯作業，例如GenerateDataKey和Decrypt。 CloudTrail 支援許多屬性值來篩選搜尋，包括事件名稱、使用者名稱和事件來源。

Amazon S3 儲存貯體金鑰

使用 AWS KMS (SSE-KMS) 設定伺服器端加密時，可以將儲存貯體設定為針對 SSE-KMS 使用 S3 儲存貯體金鑰。針對 SSE-KMS 使用儲存貯體層級金鑰，可將 Amazon S3 的 AWS KMS 請求流量減少到最多 99%，藉此將請求成本降低多達 99%。 AWS KMS

當您將儲存貯體設定為在新物件上使用 SSE-KMS 的 S3 儲存貯體金鑰時， AWS KMS 會產生一個儲存貯體層級金鑰，用於為儲存貯體中的物件建立唯一的資料金鑰。此 S3 儲存貯體金鑰會在 Amazon S3 中使用一段時間限制，進一步減少 Amazon S3 提出要求以完成 AWS KMS 加密操作的需求。如需使用 S3 儲存貯體金鑰的詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本。

要求伺服器端加密

若要在特定 Amazon S3 儲存貯體中要求所有物件的伺服器端加密，您可以使用儲存貯體政策。例如，如果請求不包含要求含 SSE-KMS 之伺服器端加密的 `x-amz-server-side-encryption-aws-kms-key-id` 標頭，則下列儲存貯體政策會拒絕向所有人上傳物件 (`s3:PutObject`) 的許可。

```
{
   "Version":"2012-10-17",
   "Id":"PutObjectPolicy",
   "Statement":[{
         "Sid":"DenyObjectsThatAreNotSSEKMS",
         "Effect":"Deny",
         "Principal":"*",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::example-s3-bucket1/*",
         "Condition":{
            "Null":{
               "s3:x-amz-server-side-encryption-aws-kms-key-id":"true"
            }
         }
      }
   ]
}
```

若要要求使用特 AWS KMS key 定物件來加密值區中的物件，您可以使用`s3:x-amz-server-side-encryption-aws-kms-key-id`條件金鑰。若要指定 KMS 金鑰，您必須使

用`arn:aws:kms:region:acct-id:key/key-id`格式為金鑰的 Amazon 資源名稱 (ARN)。 AWS Identity and Access Management 不驗證的字串是否`s3:x-amz-server-side-encryption-aws-kms-key-id`存在。

> **Note**
>
> 上傳物件時，您可以使用 `x-amz-server-side-encryption-aws-kms-key-id` 標頭指定 KMS 金鑰。如果請求中沒有該標頭，Amazon S3 會假設您想要使用 AWS 受管金鑰。無論如何，Amazon S3 用於物件加密的 AWS KMS 金鑰 ID 都必須與政策中的 AWS KMS 金鑰 ID 相符，否則 Amazon S3 會拒絕該請求。

如需 Amazon S3 特定條件金鑰的完整清單，請參閱服務授權參考中的 Amazon S3 條件金鑰。

加密內容

加密內容是一組金鑰值對，其中包含資料的其他相關內容資訊。加密內容不被加密。為加密操作指定加密內容時，Amazon S3 必須指定與解密操作相同的加密內容。否則，解密會失敗。 AWS KMS 使用加密內容作為其他驗證資料 (AAD) 來支援已驗證的加密。如需有關加密內容的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的加密內容。

根據預設，Amazon S3 會使用物件或儲存貯體 Amazon Resource Name (ARN) 作為加密內容對：

- 如果在未啟用 S3 儲存貯體金鑰的情況下使用 SSE-KMS，則使用物件 ARN 作為加密內容。

```
arn:aws:s3:::object_ARN
```

- 如果使用 SSE-KMS 並啟用 S3 儲存貯體金鑰，則使用儲存貯體 ARN 作為加密內容。如需 S3 儲存貯體金鑰的詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本。

```
arn:aws:s3:::bucket_ARN
```

您可以選擇性地使用 s3: PutObject 請求中的`x-amz-server-side-encryption-context`標頭來提供額外的加密內容對。不過，加密內容未加密，因此請確保其中不包含敏感資訊。Amazon S3 會一起存放此附加金鑰對與預設加密內容。當其處理 PUT 請求時，Amazon S3 會將 `aws:s3:arn` 的預設加密內容附加至您提供的加密內容。

您可以使用加密內容來識別和分類密碼編譯操作。您也可以使用預設加密內容 ARN 值來追蹤中 AWS CloudTrail 的相關請求，方法是檢視哪個 Amazon S3 ARN 與哪個加密金鑰搭配使用。

在 CloudTrail 記錄檔的`requestParameters`欄位中，加密內容看起來類似下列內容。

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::example-s3-bucket1/file_name"
}
```

當您搭配選用 S3 儲存貯體金鑰功能使用 SSE-KMS 時，加密內容值是儲存貯體的 ARN。

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::example-s3-bucket1"
}
```

傳送 AWS KMS 加密物件的要求

> **Important**
>
> 所有GET加 AWS KMS 密物件的PUT要求必須使用安全通訊端層 (SSL) 或傳輸層安全性 (TLS) 進行。請求也必須使用有效的認證進行簽名，例如 AWS 簽名版本 4（或 AWS 簽名版本 2）。

AWS 簽名版本 4 是將身份驗證信息添加到由 HTTP 發送的 AWS 請求的過程。為了安全起見，大多數的請求都 AWS 必須使用訪問密鑰進行簽名，該訪問密鑰包括訪問密鑰 ID 和秘密訪問密鑰。這兩種金鑰通常稱為您的安全憑證。如需詳細資訊，請參閱身分驗證請求 (AWS Signature 第 4 版) 和 Signature 第 4 版簽章程序。

> **Important**
>
> 如果您的物件使用 SSE-KMS，請不要傳送 GET 請求與 HEAD 請求的加密請求標頭。否則，您會收到 HTTP 400 Bad Request (HTTP 400 錯誤的請求) 錯誤。

主題

- 使用 AWS KMS (SSE-KMS) 指定伺服器端加密
- 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本

使用 AWS KMS (SSE-KMS) 指定伺服器端加密

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

根據預設，所有 Amazon S3 儲存貯體都設定了加密，所有上傳到 S3 儲存貯體的新物件都會在靜態時自動加密。伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 是 Amazon S3 中每個儲存貯體的預設加密組態。若要使用不同類型的加密，您可以指定 S3 PUT 請求中要使用的伺服器端加密類型，也可以在目的地儲存貯體中設定預設加密組態。

如果您想要在要PUT求中指定不同的加密類型，您可以使用伺服器端加密 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS)、使用金鑰的雙層伺服器端加密 (DSSE-KMS)，或使用客戶提供的金 AWS KMS 鑰 (SSE-C) 進行伺服器端加密。若您想在目的地儲存貯體中設定不同的預設加密組態，您可以使用 SSE-KMS 或 DSSE-KMS。

上傳新物件或複製現有物件時，您都可以套用加密。

您可以使用 Amazon S3 主控台、REST API 操作、 AWS 開發套件和 () 來指定 SSE-KMS。 AWS Command Line Interface AWS CLI如需詳細資訊，請參閱下列主題。

> **Note**
>
> 您可以 AWS KMS keys 在 Amazon S3 中使用多區域。但是，Amazon S3 目前將多區域金鑰視為單區域金鑰，並且不使用金鑰的多區域功能。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的使用多區域金鑰。

> **Note**
>
> 如果您想要使用其他帳戶所擁有的 KMS 金鑰，您必須擁有使用金鑰的權限。如需詳細了解 KMS 金鑰跨帳戶權限，請參閱《AWS Key Management Service 開發人員指南》中的建立其他帳戶可使用的 KMS 金鑰。

使用 S3 主控台

本主題說明如何設定或變更物件的加密類型，以使用 Amazon S3 主控台搭配 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 使用伺服器端加密。

> **Note**
>
> - 如果您變更物件的加密，則會建立新物件來取代舊物件。如果啟用 S3 版本控制，則系統會建立物件的新版本，且現有物件會變成較舊的版本。變更屬性的角色也會成為新物件 (或物件版本) 的擁有者。
> - 如果您變更具有使用者定義標籤之物件的加密類型，您必須擁有`s3:GetObjectTagging`權限。如果您要變更沒有使用者定義標籤但大小超過 16 MB 的物件的加密類型，您也必須擁有`s3:GetObjectTagging`權限。
>
> 如果目的地儲存貯體政策拒絕該`s3:GetObjectTagging`動作，則會更新物件的加密類型，但使用者定義的標籤會從物件中移除，而且您會收到錯誤訊息。

新增或變更物件的加密

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇包含該物件的儲存貯體名稱。
4. 在 Object (物件) 清單中，選擇您希望新增或變更加密的物件名稱。

   系統會顯示物件的詳細資訊頁面，其中包含數個區段，顯示物件的屬性。
5. 選擇屬性索引標籤。
6. 向下捲動至伺服器端加密設定區段，然後選擇編輯。

輯伺服器端加密頁面隨即開啟。

7. 在伺服器端加密設定底下，針對加密設定，選擇覆寫預設加密儲存貯體設定。
8. 在 [加密類型] 下，選擇 [伺服器端使用金 AWS Key Management Service 鑰加密 (SSE-KMS)]。

> **Important**
>
> 如果您針對預設加密組態使用 SSE-KMS 選項，則受到 AWS KMS的每秒請求數目 (RPS) 配額限制。如需 AWS KMS 配額以及如何請求提高配額的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的配額。

9. 在 AWS KMS 金鑰之下，執行下列其中一個動作，以選擇 KMS 金鑰：

   - 若要從可用的 KMS 金鑰清單中選擇，請選擇從 AWS KMS keys中選擇，然後從可用金鑰清單中選擇您的 KMS 金鑰。

     AWS 受管金鑰 (aws/s3) 和您的客戶管理金鑰都會出現在此清單中。如需詳細了解客戶受管金鑰，請參閱《AWS Key Management Service 開發人員指南》中的客戶金鑰和 AWS 金鑰。
   - 若要輸入 KMS 金鑰 ARN，請選擇 [輸入 AWS KMS key ARN]，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。
   - 若要在 AWS KMS 主控台中建立新的客戶管理金鑰，請選擇 [建立 KMS 金鑰]。

     如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。

> **Important**
>
> 您只能使用與值區相同 AWS 區域 的 KMS 金鑰。Amazon S3 主控台僅會列出與儲存貯體位於相同區域的前 100 個 KMS 金鑰。若要使用未列出的 KMS 金鑰，必須輸入 KMS 金鑰 ARN。若您想要使用其他帳戶的 KMS 金鑰，您必須先具有該金鑰的使用權限，然後輸入 KMS 金鑰 ARN。
>
> Amazon S3 僅支援對稱加密 KMS 金鑰，而不支援非對稱 KMS 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的識別對稱和非對稱 KMS 金鑰。

10. 選擇 Save changes (儲存變更)。

> **Note**
>
> 此動作會將加密套用至所有指定的物件。加密資料夾時，請等待儲存作業完成，然後再將新物件新增至資料夾。

使用 REST API

建立物件 (即上傳新物件或複製現有物件) 時，可以指定使用伺服器端加密和 AWS KMS keys (SSE-KMS) 來加密資料。若要執行這項操作，請將 `x-amz-server-side-encryption` 標頭新增至要求。將標頭的值設為加密演算法 `aws:kms`。Amazon S3 會傳回回應標頭 `x-amz-server-side-encryption`，確認已使用 SSE-KMS 存放物件。

如果您使用 `aws:kms` 的值指定 `x-amz-server-side-encryption` 標頭，也可以使用下列要求標頭：

- `x-amz-server-side-encryption-aws-kms-key-id`
- `x-amz-server-side-encryption-context`
- `x-amz-server-side-encryption-bucket-key-enabled`

主題

- 支援 SSE-KMS 的 Amazon S3 REST API 操作
- 加密內容 (x-amz-server-side-encryption-context)
- AWS KMS 金鑰識別碼 (x-amz-server-side-encryption-aws-kms-key-id)
- S3 儲存貯體金鑰 (x-amz-server-side-encryption-aws-bucket-key-enabled)

支援 SSE-KMS 的 Amazon S3 REST API 操作

下列 REST API 操作接受 `x-amz-server-side-encryption`、`x-amz-server-side-encryption-aws-kms-key-id` 和 `x-amz-server-side-encryption-context` 請求標頭。

- PutObject – 使用 PUT API 操作上傳資料時，您可以指定這些請求標頭。
- CopyObject – 複製物件時，您會同時有來源物件與目標物件。當您將 SSE-KMS 標頭與`CopyObject`作業一起傳遞時，它們只會套用至目標物件。當您複製現有物件時，無論來源物件是否加密，除非您明確要求伺服器端加密，否則目標物件都不會加密。

- POST Object— 當您使用POST作業上載物件時，而不是要求標頭，您會在表單欄位中提供相同的資訊。
- CreateMultipartUpload— 當您使用多部分上傳 API 作業上載大型物件時，您可以指定這些標頭。您可以在CreateMultipartUpload要求中指定這些標頭。

使用伺服器端加密存放物件時，下列 REST API 操作的回應標頭會傳回 x-amz-server-side-encryption 標頭。

- PutObject
- CopyObject
- POST Object
- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- GetObject
- HeadObject

> **Important**
>
> - 如果您 AWS KMS 未使用安全通訊端層 (SSL)、傳輸層安全性 (TLS) 或簽章版本 4 來提出這些要求，則所有物件GET和PUT受到失敗保護的要求。
> - 如果您的物件使用 SSE-KMS，請勿針對要GET求和HEAD要求傳送加密要求標頭，否則會收到 HTTP 400 BadRequest 錯誤訊息。

加密內容 (**x-amz-server-side-encryption-context**)

如果您指定 x-amz-server-side-encryption:aws:kms，則 Amazon S3 API 支援具有 x-amz-server-side-encryption-context 標頭的加密內容。加密內容是一組金鑰值對，其中包含資料的其他相關內容資訊。

Amazon S3 會自動使用物件或儲存貯體 Amazon Resource Name (ARN) 作為加密內容對。如果您在未啟用 S3 儲存貯體金鑰的情況下使用 SSE-KMS，則您使用物件 ARN 做為加密內容，例如

`arn:aws:s3:::object_ARN`。不過，如果您使用 SSE-KMS 並啟用 S3 儲存貯體金鑰，則會將儲存貯體 ARN 用於加密內容，例如 `arn:aws:s3:::bucket_ARN`。

您可以使用 `x-amz-server-side-encryption-context` 標頭來提供其他加密內容對。但是，由於加密上下文未加密，因此請確保其中不包含敏感信息。Amazon S3 會一起存放此附加金鑰對與預設加密內容。

如需 Amazon S3 中加密內容的相關資訊，請參閱 加密內容。如需有關加密內容的更多資訊，請參閱《AWS Key Management Service 開發人員指南》中的 AWS Key Management Service 概念：加密內容。

AWS KMS 金鑰識別碼 (**`x-amz-server-side-encryption-aws-kms-key-id`**)

您可以使用 `x-amz-server-side-encryption-aws-kms-key-id` 標頭來指定用來保護資料之客戶受管金鑰的 ID。如果您指定 `x-amz-server-side-encryption:aws:kms` 標頭，但未提供 `x-amz-server-side-encryption-aws-kms-key-id` 標頭，則 Amazon S3 會使用 AWS 受管金鑰 (`aws/s3`) 來保護資料。如果您想要使用客戶受管的金鑰，則必須提供客戶受管金鑰的 `x-amz-server-side-encryption-aws-kms-key-id` 標頭。

> **Important**
>
> 當您在 Amazon S3 中使 AWS KMS key 用伺服器端加密時，您必須選擇對稱加密 KMS 金鑰。Amazon S3 只支援對稱加密 KMS 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的對稱加密 KMS 金鑰。

S3 儲存貯體金鑰 (**`x-amz-server-side-encryption-aws-bucket-key-enabled`**)

您可以使用`x-amz-server-side-encryption-aws-bucket-key-enabled`請求標頭在物件層級啟用或停用 S3 儲存貯體金鑰。S3 儲存貯體金鑰可將 Amazon S3 的請求流量減少到 AWS KMS。AWS KMS 如需詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本。

如果您指定 `x-amz-server-side-encryption:aws:kms` 標頭但未提供 `x-amz-server-side-encryption-aws-bucket-key-enabled` 標頭，則您的物件使用目的地儲存貯體存儲桶的 S3 儲存貯體金鑰設定來加密您的物件。如需詳細資訊，請參閱 在物件層級設定 S3 儲存貯體金鑰 。

使用 AWS CLI

若要使用下列範例 AWS CLI 指令，請以您自己*`user input placeholders`*的資訊取代。

當您上傳新物件或複製現有物件時，您可以指定使用伺服器端加密 AWS KMS 金鑰來加密資料。若要執行這項操作，請將 `--server-side-encryption aws:kms` 標頭新增至要求。使用新`--ssekms-key-id` *`example-key-id`*增您建立的客戶管理 AWS KMS 金鑰。如果您指定`--server-side-encryption aws:kms`但未提供 AWS KMS 金鑰 ID，Amazon S3 將使用受 AWS 管金鑰。

```
aws s3api put-object --bucket example-s3-bucket --key example-object-key --server-side-encryption aws:kms --ssekms-key-id example-key-id --ssekms-encryption-context example-encryption-context --body filepath
```

您可以透過新增`--bucket-key-enabled`或在您的或`copy-object`作業上啟用`put-object`或停用 S3 儲存貯體金鑰`--no-bucket-key-enabled`。S3 儲存貯體金鑰可以透過將 Amazon S3 的 AWS KMS 請求流量減少到，從而降低請求成本 AWS KMS。如需詳細資訊，請參閱使用 S3 儲存貯體金鑰降低 SSE-KMS 的成本。

```
aws s3api put-object --bucket example-s3-bucket --key example-object-key --server-side-encryption aws:kms --bucket-key-enabled --body filepath
```

您可以將物件從來源儲存貯體複製到新值區，並指定 SSE-KMS 加密。

```
aws s3api copy-object --copy-source example-s3-bucket/example-object-key --
bucket example-s3-bucket2 --key example-object-key --server-side-encryption aws:kms --
sse-kms-key-id example-key-id --ssekms-encryption-context example-encryption-context
```

使用 AWS 軟體開發套件

使用 AWS 開發套件時，您可以請求 Amazon S3 用 AWS KMS keys 於伺服器端加密。下列範例說明如何使用 SSE-KMS 搭配 Java 和 .NET 的 AWS 開發套件。如需其他 SDK 的相關資訊，請參閱 AWS 開發人員中心上的範例程式碼和程式庫。

> **Important**
>
> 當您在 Amazon S3 中使 AWS KMS key 用伺服器端加密時，您必須選擇對稱加密 KMS 金鑰。Amazon S3 只支援對稱加密 KMS 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的對稱加密 KMS 金鑰。

`CopyObject` 操作

複製物件時，請新增相同的請求屬性 (`ServerSideEncryptionMethod` 與 `ServerSideEncryptionKeyManagementServiceKeyId`)，以請求 Amazon S3 使用 AWS KMS key。如需複製物件的詳細資訊，請參閱「複製、移動和重新命名物件」。

PUT 操作

Java

使用上傳物件時 AWS SDK for Java，您可以透過新增`SSEAwsKeyManagementParams`屬性來請求 Amazon S3 使用，如下列請求所示： AWS KMS key

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
   keyName, file).withSSEAwsKeyManagementParams(new SSEAwsKeyManagementParams());
```

在這種情況下，Amazon S3 使用 AWS 受管金鑰 （ `aws/s3` ）。如需詳細資訊，請參閱 使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)。您可以選擇性地建立對稱加密 KMS 金鑰，並在要求中指定該金鑰，如下列範例所示：

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
   keyName, file).withSSEAwsKeyManagementParams(new
 SSEAwsKeyManagementParams(keyID));
```

如需有關建立客戶受管金鑰的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的程式設計 AWS KMS API。

如需上傳物件的可行程式碼範例，請參閱下列主題。若要使用這些範例，您必須更新這些程式碼範例並提供加密資訊，如上述程式碼片段所示。

- 如需以單一操作上傳物件，請參閱 上傳物件。
- 如需使用高階或低階分段上傳 API 作業的多部分上傳，請參閱。使用分段上傳來上傳物件

.NET

使用上傳物件時 AWS SDK for .NET，您可以透過新增`ServerSideEncryptionMethod`屬性來請求 Amazon S3 使用，如下列請求所示： AWS KMS key

```
PutObjectRequest putRequest = new PutObjectRequest
```

```
 {
     BucketName = example-s3-bucket,
     Key = keyName,
     // other properties
     ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS
 };
```

在這種情況下，Amazon S3 使用 AWS 受管金鑰. 如需詳細資訊，請參閱 使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)。您可以選擇性地建立自己的對稱加密客戶管理金鑰，並在要求中指定該金鑰，如下列範例所示：

```
PutObjectRequest putRequest1 = new PutObjectRequest
{
  BucketName = example-s3-bucket,
  Key = keyName,
  // other properties
  ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS,
  ServerSideEncryptionKeyManagementServiceKeyId = keyId
};
```

如需有關建立客戶受管金鑰的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的程式設計 AWS KMS API。

如需上傳物件的可行程式碼範例，請參閱下列主題。若要使用這些範例，您必須更新這些程式碼範例並提供加密資訊，如上述程式碼片段所示。

- 如需以單一操作上傳物件，請參閱 上傳物件。
- 如需使用高階或低階分段上傳 API 作業的多部分上傳，請參閱。使用分段上傳來上傳物件

預先簽章的 URL

Java

為使用加密的物件建立預先簽署的 URL 時 AWS KMS key，您必須明確指定簽章版本 4，如下列範例所示：

```
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setSignerOverride("AWSS3V4SignerType");
AmazonS3Client s3client = new AmazonS3Client(
        new ProfileCredentialsProvider(), clientConfiguration);
```

```
...
```

如需程式碼範例，請參閱「使用預先簽章的 URL 來共用物件」。

.NET

為使用加密的物件建立預先簽署的 URL 時 AWS KMS key，您必須明確指定簽章版本 4，如下列範例所示：

```
AWSConfigs.S3Config.UseSignatureVersion4 = true;
```

如需程式碼範例，請參閱「使用預先簽章的 URL 來共用物件」。

使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本

Amazon S3 儲存貯體金鑰使用 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS) 降低 Amazon S3 伺服器端加密的成本。針對 SSE-KMS 使用儲存貯體層級金鑰，可將 Amazon S3 的 AWS KMS 請求流量減少到最多 99% 的請求成本。 AWS KMS只要在 AWS Management Console中點擊幾下，而無需對用戶端應用程式進行任何變更，您就可以設定儲存貯體，在新物件上使用 S3 儲存貯體金鑰進行 SSE-KMS 加密。

> **Note**
>
> 使用 AWS Key Management Service () 金鑰 (DSSE-KMS AWS KMS) 進行雙層伺服器端加密時，不支援 S3 儲存貯體金鑰。

適用於 SSE-KMS 的 S3 儲存貯體金鑰

存取使用 SSE-KMS 加密的數百萬或數十億個物件的工作負載，可能會產生大量的要求。 AWS KMS 當您使用 SSE-KMS 在沒有 S3 儲存貯體金鑰的情況下保護資料時，Amazon S3 會針對每個物件使用個別的 AWS KMS 資料金鑰。在這種情況下，Amazon S3 會在 AWS KMS 每次針對 KMS 加密物件發出請求時進行呼叫。如需 SSE-KMS 如何運作的相關資訊，請參閱 使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)。

當您將儲存貯體設定為針對 SSE-KMS 使用 S3 儲存貯體金鑰時， AWS 會從中產生短暫的儲存貯體層級金鑰 AWS KMS，然後暫時將其保留在 S3 中。此儲存貯體層級金鑰會其生命週期中建立資料金鑰。S3 儲存貯體金鑰會在 Amazon S3 中使用有限的時間段，減少 S3 提出要求完成 AWS KMS 加密

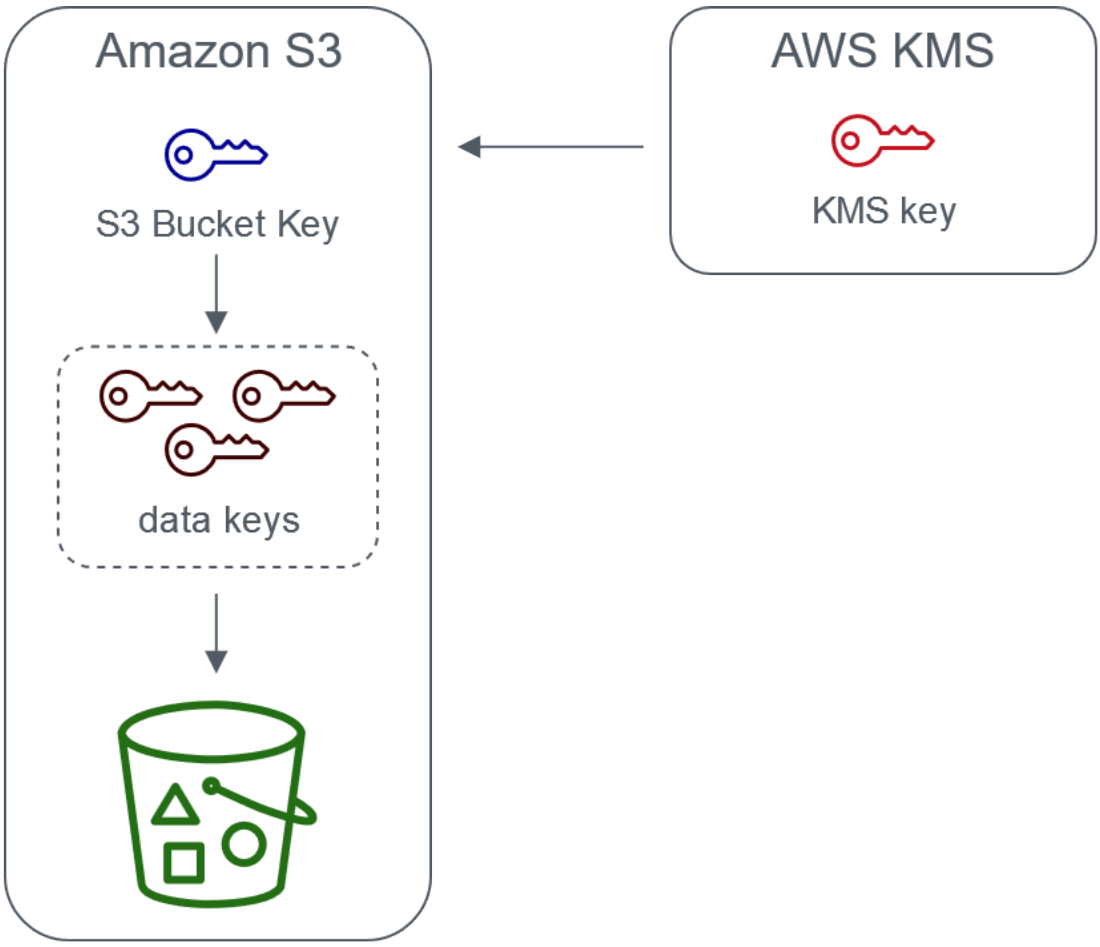
操作的需求。如此可將 S3 的流量減少到 AWS KMS，讓您以先前成本的一小部分存取 Amazon S3 中的 AWS KMS加密物件。

每個請求者至少會擷取一次唯一的儲存貯體層級金鑰，以確保在事件中擷取要求者對金鑰的存取權。AWS KMS CloudTrail 當來電者使用不同的角色或帳戶時，Amazon S3 會將來電者視為不同的請求者，或使用不同範圍政策的相同角色。 AWS KMS 節省請求反映了請求者的數量，請求模式以及請求對象的相對年齡。例如，較少的請求者數量、在有限的時間範圍內請求多個物件，以及使用相同的儲存貯體層級金鑰加密，會節省更多成本。

> **Note**
>
> 使用 S3 儲存貯體金鑰可讓您透過使用儲存貯體層級金鑰將要 AWS KMS 求減少到`EncryptGenerateDataKey`、和`Decrypt`操作，以節省請求成本。 AWS KMS 根據設計，利用此儲存貯體層級金鑰的後續要求不會產生 AWS KMS API 要求或根據金鑰原則驗證存取權。 AWS KMS

當您設定 S3 儲存貯體金鑰時，儲存貯體中已存在的物件不會使用 S3 儲存貯體金鑰。若要為現有物件設定 S3 儲存貯體金鑰，您可以使用 `CopyObject` 操作。如需詳細資訊，請參閱 在物件層級設定 S3 儲存貯體金鑰 。

Amazon S3 只會針對相同 AWS KMS key加密的物件共用 S3 儲存貯體金鑰。S3 儲存貯體金鑰與由建立的 KMS 金鑰 AWS KMS、匯入的金鑰材料以及由自訂金鑰存放區支援的金鑰材料相容。



Server-side encryption with AWS Key Management service using an S3 Bucket Key

設定 S3 儲存貯體金鑰

您可以透過 Amazon S3 主控台、 AWS 開發套件或 REST API，將儲存貯體設定為針對新物件使用 SSE-KMS 的 S3 儲存貯體金鑰。 AWS CLI在儲存貯體上啟用 S3 儲存貯體金鑰後，使用不同所指定 SSE-KMS 金鑰上傳的物件將使用自己的 S3 儲存貯體金鑰。無論您的 S3 儲存貯體金鑰設定為何，您都可以在請求中包含具有 `true` 或 `false` 值的 `x-amz-server-side-encryption-bucket-key-enabled` 標頭，以覆寫儲存貯體設定。

在您將儲存貯體設定為使用 S3 儲存貯體金鑰之前，請先檢閱 啟用 S3 儲存貯體金鑰之前，要注意的變更。

使用 Amazon S3 主控台設定 S3 儲存貯體金鑰

當您建立新儲存貯體時，您可以將儲存貯體設定為在新物件上使用 SSE-KMS 的 S3 儲存貯體金鑰。您也可以透過更新儲存貯體屬性，將現有儲存貯體設定為在新物件上使用 SSE-KMS 的 S3 儲存貯體金鑰。

如需詳細資訊，請參閱 設定您的儲存貯體以使用具有 SSE-KMS 的 S3 儲存貯體來獲取新物件。

S3 儲存貯體金鑰的 REST AWS API 和開發套件支援 AWS CLI

您可以使用 REST API 或 AWS SDK 將儲存貯體設定為在新物件上使用適用於 SSE-KMS 的 S3 儲存貯體金鑰。 AWS CLI您也可以在物件層級啟用 S3 儲存貯體金鑰。

如需詳細資訊，請參閱下列內容：

- 在物件層級設定 S3 儲存貯體金鑰
- 設定您的儲存貯體以使用具有 SSE-KMS 的 S3 儲存貯體來獲取新物件

下列 API 操作支援 SSE-KMS 的 S3 儲存貯體金鑰：

- PutBucketEncryption
  - `ServerSideEncryptionRule`接受啟用和停用 S3 儲存貯體金鑰的 `BucketKeyEnabled` 參數。
- GetBucketEncryption
  - `ServerSideEncryptionRule` 會傳回 `BucketKeyEnabled` 的設定。
- PutObject、CopyObjectCreateMultipartUpload、和 POST 物件
  - `x-amz-server-side-encryption-bucket-key-enabled` 請求標頭啟用或停用在物件層級的 S3 儲存貯體金鑰。
- HeadObject、GetObjectUploadPartCopy、UploadPart、和 CompleteMultipartUpload
  - `x-amz-server-side-encryption-bucket-key-enabled` 回應標頭表示是否為物件啟用或停用 S3 儲存貯體金鑰。

使用 AWS CloudFormation

在中 AWS CloudFormation，資`AWS::S3::Bucket`源包含稱為的加密屬性`BucketKeyEnabled`，您可以用來啟用或停用 S3 儲存貯體金鑰。

如需詳細資訊，請參閱 使用 AWS CloudFormation。

啟用 S3 儲存貯體金鑰之前，要注意的變更

啟用 S3 儲存貯體金鑰之前，請注意下列相關變更：

IAM 或 AWS KMS 金鑰政策

如果您現有的 AWS Identity and Access Management (IAM) 政策或金 AWS KMS 鑰政策使用您的物件 Amazon 資源名稱 (ARN) 做為加密內容，以精簡或限制對 KMS 金鑰的存取，則這些政策將無法與 S3 儲存貯體金鑰搭配使用。S3 儲存貯體金鑰使用儲存貯體 ARN 作為加密內容。啟用 S3 儲存貯體金鑰之前，請更新您的 IAM 政策或 AWS KMS 金鑰政策，以使用儲存貯體 ARN 做為加密內容。

如需加密內容與 S3 儲存貯體金鑰的詳細資訊，請參閱 加密內容。

CloudTrail 的活動 AWS KMS

啟用 S3 儲存貯體金鑰後，您的 AWS KMS CloudTrail 事件會記錄儲存貯體 ARN，而不是物件 ARN。此外，您會在記錄中看到 SSE-KMS 物件的 KMS CloudTrail 事件較少。 由於 Amazon S3 中的金鑰材料是有時間限制的，所以 AWS KMS要求的請求較少。

使用 S3 儲存貯體金鑰與複寫

您可以將 S3 儲存貯體金鑰與相同區域複寫 (SRR) 和跨區域複寫 (CRR) 搭配使用。

Amazon S3 複寫加密物件時，通常會在目的地儲存貯體中保留複本物件的加密設定。不過，如果來源物件未加密，且目的地儲存貯體使用預設加密或 S3 儲存貯體金鑰，Amazon S3 會使用目的地儲存貯體的組態加密物件。

下列範例說明 S3 儲存貯體金鑰如何與複寫搭配使用。如需詳細資訊，請參閱 複寫加密的物件 (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)。

Example 範例 1 — 來源物件使用 S3 儲存貯體金鑰，目的地儲存貯體使用預設加密

如果您的來源物件使用 S3 儲存貯體金鑰，但目的地儲存貯體使用預設加密搭配 SSE-KMS，則複本物件會在目的地儲存貯體中維護其 S3 儲存貯體金鑰加密設定。目的地儲存貯體仍使用預設加密搭配 SSE-KMS。

Example 範例 2 — 來源物件未加密，目的地儲存貯體使用 S3 儲存貯體金鑰搭配 SSE-KMS

如果您的來源物件未加密，且目的地儲存貯體使用 SSE-KMS 的 S3 儲存貯體金鑰，則複本物件會使用目的地儲存貯體中的 SSE-KMS，使用 S3 儲存貯體金鑰加密。這會導致來源物件的 ETag 與複本物件的 ETag 不同。您必須更新可使用 ETag 的應用程式來容納這項差異。

使用 S3 儲存貯體金鑰

如需啟用及使用 S3 儲存貯體金鑰的詳細資訊，請參閱下列章節：

- 設定您的儲存貯體以使用具有 SSE-KMS 的 S3 儲存貯體來獲取新物件
- 在物件層級設定 S3 儲存貯體金鑰
- 檢視 S3 儲存貯體金鑰的設定

設定您的儲存貯體以使用具有 SSE-KMS 的 S3 儲存貯體來獲取新物件

使用 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS) 設定伺服器端加密時，可以將儲存貯體設定為在新物件上使用 SSE-KMS 的 S3 儲存貯體金鑰。S3 儲存貯體金鑰可將 Amazon S3 的請求流量減少到 AWS KMS 並降低 SSE-KMS 的成本。如需詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本。

您可以使用 Amazon S3 主控台、REST API、 AWS 開發套件 AWS Command Line Interface (AWS CLI) 或，將儲存貯體設定為在新物件上使用 SSE-KMS 的 S3 儲存貯體金鑰。 AWS CloudFormation如果您想要啟用或停用現有物件的 S3 儲存貯體金鑰，可以使用 `CopyObject` 操作。如需詳細資訊，請參閱 在物件層級設定 S3 儲存貯體金鑰 與 使用 S3 批次操作以透過 S3 儲存貯體金鑰來加密物件。

針對來源或目的地儲存貯體啟用 S3 儲存貯體金鑰時，加密內容將是儲存貯體 Amazon Resource Name (ARN)，而不是物件 ARN，例如 `arn:aws:s3:::`*`bucket_ARN`*。您需要更新 IAM 政策，才能將儲存貯體 ARN 用於加密內容。如需詳細資訊，請參閱 S3 儲存貯體金鑰和複寫。

下列範例說明 S3 儲存貯體金鑰如何與複寫搭配使用。如需詳細資訊，請參閱 複寫加密的物件 (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)。

必要條件

在您將儲存貯體設定為使用 S3 儲存貯體金鑰之前，請先檢閱 啟用 S3 儲存貯體金鑰之前，要注意的變更。

使用 S3 主控台

在 S3 主控台中，您可以啟用或停用新的或現有儲存貯體的 S3 儲存貯體金鑰。S3 主控台中的物件會從儲存貯體組態繼承其 S3 儲存貯體金鑰設定。當您為儲存貯體啟用 S3 儲存貯體金鑰時，上傳至儲存貯體的新物件使用 S3 儲存貯體金鑰以進行 SSE-KMS。

在啟用 S3 儲存貯體金鑰的儲存貯體中上傳、複製或修改物件

如果您在已啟用 S3 儲存貯體金鑰的儲存貯體中上傳、修改或複製物件，則該物件的 S3 儲存貯體金鑰設定可能會更新以符合儲存貯體組態。

如果物件已啟用 S3 儲存貯體金鑰，則在您複製或修改物件時，該物件的 S3 儲存貯體金鑰設定不會變更。但是，如果您修改或複製未啟用 S3 儲存貯體金鑰的物件，且目的地儲存貯體具有 S3 儲存貯體

金鑰組態，則物件會繼承目的地儲存貯體的 S3 儲存貯體金鑰設定。例如，如果您的來源物件沒有啟用 S3 儲存貯體金鑰，但目的地儲存貯體已啟用 S3 儲存貯體金鑰，則將為該物件啟用 S3 儲存貯體金鑰。

在建立新儲存貯體時啟用 S3 儲存貯體金鑰

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇建立儲存貯體。
4. 輸入您的儲存貯體名稱，然後選擇您的 AWS 區域。
5. 在預設加密之下，針對加密金鑰類型選擇 AWS Key Management Service 金鑰 (SSE-KMS)。
6. 在 AWS KMS 金鑰之下，執行下列其中一個動作，以選擇 KMS 金鑰：

   - 若要從可用 KMS 金鑰清單中選擇，請選擇 [從您的] 中選擇 AWS KMS keys，然後從可用金鑰清單中選擇您的 KMS 金鑰。

     AWS 受管金鑰 (aws/s3) 和您的客戶管理金鑰都會出現在此清單中。如需有關客戶受管金鑰的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的客戶 AWS 金鑰和金鑰。
   - 若要輸入 KMS 金鑰 ARN，請選擇輸入 AWS KMS key ARN，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。
   - 若要在 AWS KMS 主控台中建立新的客戶管理金鑰，請選擇 [建立 KMS 金鑰]。

     如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。
7. 在 Bucket Key (儲存貯體金鑰) 底下，選擇 Enable (啟用)。
8. 選擇 Create bucket (建立儲存貯體)。

   Amazon S3 會在啟用 S3 儲存貯體金鑰的情況下建立您的儲存貯體。您上傳到儲存貯體的新物件將使用 S3 儲存貯體金鑰。

   若要停用 S3 儲存貯體金鑰，請依照上述步驟執行，然後選擇 Disable (停用)。

為現有儲存貯體啟用 S3 儲存貯體金鑰

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。

3. 在 Buckets (儲存貯體) 清單中，選擇您要啟用 S3 儲存貯體金鑰的儲存貯體。
4. 選擇屬性索引標籤。
5. 在 Default encryption (預設加密) 底下，選擇 Edit (編輯)。
6. 在預設加密之下，針對加密金鑰類型選擇 AWS Key Management Service 金鑰 (SSE-KMS)。
7. 在 AWS KMS 金鑰之下，執行下列其中一個動作，以選擇 KMS 金鑰：

   - 若要從可用 KMS 金鑰清單中選擇，請選擇 [從您的] 中選擇 AWS KMS keys，然後從可用金鑰清單中選擇您的 KMS 金鑰。

     AWS 受管金鑰 (aws/s3) 和您的客戶管理金鑰都會出現在此清單中。如需有關客戶受管金鑰的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的客戶 AWS 金鑰和金鑰。
   - 若要輸入 KMS 金鑰 ARN，請選擇輸入 AWS KMS key ARN，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。
   - 若要在 AWS KMS 主控台中建立新的客戶管理金鑰，請選擇 [建立 KMS 金鑰]。

     如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。
8. 在 Bucket Key (儲存貯體金鑰) 底下，選擇 Enable (啟用)。
9. 選擇 Save changes (儲存變更)。

   Amazon S3 為新增至儲存貯體的新物件啟用 S3 儲存貯體金鑰。現有物件不使用 S3 儲存貯體金鑰。若要為現有物件設定 S3 儲存貯體金鑰，您可以使用 `CopyObject` 操作。如需詳細資訊，請參閱 在物件層級設定 S3 儲存貯體金鑰 。

   若要停用 S3 儲存貯體金鑰，請依照上述步驟執行，然後選擇 Disable (停用)。

使用 REST API

您可以使用PutBucketEncryption為儲存貯體啟用或停用 S3 儲存貯體金鑰。 若要使用設定 S3 儲存貯體金鑰`PutBucketEncryption`，請使用ServerSideEncryptionRule資料類型，其中包括使用 SSE-KMS 的預設加密。您也可以指定客戶受管金鑰的 KMS 金鑰 ID，選擇性地使用客戶受管金鑰。

如需詳細資訊和語法範例，請參閱PutBucketEncryption。

使用適用於 Java 的 AWS 開發套件

下列範例使用 AWS SDK for Java，以 SSE-KMS 和 S3 儲存貯體金鑰啟用預設儲存貯體加密。

Java

```
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .build();

ServerSideEncryptionByDefault serverSideEncryptionByDefault = new
 ServerSideEncryptionByDefault()
    .withSSEAlgorithm(SSEAlgorithm.KMS);
ServerSideEncryptionRule rule = new ServerSideEncryptionRule()
    .withApplyServerSideEncryptionByDefault(serverSideEncryptionByDefault)
    .withBucketKeyEnabled(true);
ServerSideEncryptionConfiguration serverSideEncryptionConfiguration =
    new ServerSideEncryptionConfiguration().withRules(Collections.singleton(rule));

SetBucketEncryptionRequest setBucketEncryptionRequest = new
 SetBucketEncryptionRequest()
    .withServerSideEncryptionConfiguration(serverSideEncryptionConfiguration)
    .withBucketName(bucketName);

s3client.setBucketEncryption(setBucketEncryptionRequest);
```

使用 AWS CLI

下列範例使用 AWS CLI，以 SSE-KMS 和 S3 儲存貯體金鑰啟用預設儲存貯體加密。以您自己的資訊取代 *user input placeholders*。

```
aws s3api put-bucket-encryption --bucket example-s3-bucket --server-side-encryption-
configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
                "BucketKeyEnabled": true
            }
        ]
    }'
```

使用 AWS CloudFormation

如需使用設定 S3 儲存貯體金鑰的詳細資訊 AWS CloudFormation，請參閱AWS CloudFormation 使用者指南AWS::S3::Bucket ServerSideEncryptionRule中的。

在物件層級設定 S3 儲存貯體金鑰

當您使用 REST API、 AWS 軟體開發套件執行 PUT 或 COPY 作業時 AWS CLI，您可以在物件層級啟用或停用 S3 儲存貯體金鑰，方法是使用`true`或`false`值新增`x-amz-server-side-encryption-bucket-key-enabled`要求標頭。S3 儲存貯體金鑰透過將 Amazon S3 的請求流量減少到使用 AWS Key Management Service (AWS KMS) (SSE-KMS) 來降低伺服器端加密的成本。 AWS KMS如需詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本。

當您使用 PUT 或 COPY 操作為物件設定 S3 儲存貯體金鑰時，Amazon S3 只會更新該物件的設定。目的地儲存貯體的 S3 儲存貯體金鑰設定不會變更。如果您將 KMS 加密物件的 PUT 或 COPY 請求提交至啟用 S3 儲存貯體金鑰的儲存貯體，除非您停用請求標頭中的金鑰，否則物件層級操作將自動使用 S3 儲存貯體金鑰。如果您沒有為物件指定 S3 儲存貯體金鑰，Amazon S3 會將目的地儲存貯體的 S3 儲存貯體金鑰設定套用至物件。

必要條件：

將物件設定為使用 S3 儲存貯體金鑰之前，請先檢閱 啟用 S3 儲存貯體金鑰之前，要注意的變更。

主題

- Amazon S3 批次操作
- 使用 REST API
- 使用 AWS 開 SDK for Java (PutObject)
- 使用 AWS CLI （ PutObject ）

Amazon S3 批次操作

若要加密現有的 Amazon S3 物件，您可以使用 Amazon S3 批次操作。請提供 S3 批次操作可操作的物件清單，批次操作就會呼叫個別 API 來執行指定的操作。

您可以使用 S3 批次操作複製操作來複製現有的未加密物件，並在相同的儲存貯體中寫入新的加密物件。單一批次操作任務可在數十億個物件上執行指定的操作。如需詳細資訊，請參閱「在 Amazon S3 物件上執行大規模批次操作」和使用 Amazon S3 批次操作來加密物件。

使用 REST API

當您使用 SSE-KMS 時，可以使用下列 API 操作為物件啟用 S3 儲存貯體金鑰：

- PutObject— 上傳物件時，您可以指定x-amz-server-side-encryption-bucket-key-enabled要求標頭，以在物件層級啟用或停用 S3 儲存貯體金鑰。
- CopyObject— 複製物件並設定 SSE-KMS 時，您可以指定x-amz-server-side-encryption-bucket-key-enabled要求標頭來啟用或停用物件的 S3 儲存貯體金鑰。
- POST 物件 — 當您使用 POST 操作上傳物件並設定 SSE-KMS 時，您可以使用 x-amz-server-side-encryption-bucket-key-enabled 表單欄位來啟用或停用物件的 S3 儲存貯體金鑰。
- CreateMultipartUpload— 當您使用 CreateMultipartUpload API 作業上傳大型物件並設定 SSE-KMS 時，您可以使用x-amz-server-side-encryption-bucket-key-enabled要求標頭為物件啟用或停用 S3 儲存貯體金鑰。

若要在物件層級啟用 S3 儲存貯體金鑰，請包含 x-amz-server-side-encryption-bucket-key-enabled 請求標頭。如需有關 SSE-KMS 和 REST API 的詳細資訊，請參閱 使用 REST API。

使用 AWS 開 SDK for Java (PutObject)

您可以使用下列範例，使用 AWS SDK for Java在物件層級設定 S3 儲存貯體金鑰。

Java

```
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .build();

String bucketName = "DOC-EXAMPLE-BUCKET1";
String keyName = "key name for object";
String contents = "file contents";

PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, keyName,
 contents)
    .withBucketKeyEnabled(true);

s3client.putObject(putObjectRequest);
```

使用 AWS CLI （ PutObject ）

您可以使用下列 AWS CLI 範例在物件層級設定 S3 儲存貯體金鑰，做為PutObject請求的一部分。

```
aws s3api put-object --bucket example-s3-bucket --key object key name --server-side-
encryption aws:kms --bucket-key-enabled --body filepath
```

檢視 S3 儲存貯體金鑰的設定

您可以使用 Amazon S3 主控台、REST API、 AWS Command Line Interface (AWS CLI) 或 AWS 開發套件，在儲存貯體或物件層級檢視 S3 儲存貯體金鑰的設定。

S3 儲存貯體金鑰可減少 Amazon S3 的請求流量， AWS KMS 並使用 AWS Key Management Service (SSE-KMS) 降低伺服器端加密的成本。如需詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本。

若要檢視儲存貯體或已從儲存貯體組態繼承 S3 儲存貯體金鑰設定之物件的 S3 儲存貯體金鑰設定，您需要執行 s3:GetEncryptionConfiguration 動作的許可。如需詳細資訊，請參閱 Amazon 簡易儲存服務 API 參考GetBucketEncryption中的。

使用 S3 主控台

在 S3 主控台中，您可以查看儲存貯體或物件的 S3 儲存貯體金鑰設定。除非來源物件已設定 S3 儲存貯體金鑰，否則會從儲存貯體組態繼承 S3 儲存貯體金鑰設定。

同一儲存貯體中的物件和資料夾可以具有不同的 S3 儲存貯體金鑰設定。例如，如果您使用 REST API 上傳物件並為物件啟用 S3 儲存貯體金鑰，則物件會在目的地儲存貯體中保留其 S3 儲存貯體金鑰設定，即使已在目的地儲存貯體中停用 S3 儲存貯體金鑰。另一個範例，如果您為現有儲存貯體啟用 S3 儲存貯體金鑰，則儲存貯體中已存在的物件不會使用 S3 儲存貯體金鑰。但是，新物件啟用了 S3 儲存貯體金鑰。

檢視儲存佇體的 S3 儲存貯體金鑰設定

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇您要啟用 S3 儲存貯體金鑰的儲存貯體。
4. 選擇 Properties (屬性)。
5. 在預設加密區段，儲存貯體金鑰底下，您會看到儲存貯體的 S3 儲存貯體金鑰設定。

如果您看不到 S3 儲存貯體金鑰設定，您可能沒有執行 `s3:GetEncryptionConfiguration` 動作的許可。如需詳細資訊，請參閱 Amazon 簡易儲存服務 API 參考GetBucketEncryption中的。

檢視物件的 S3 儲存貯體金鑰設定

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇您要啟用 S3 儲存貯體金鑰的儲存貯體。
3. 在 Object (物件) 清單中，選擇您的物件名稱。
4. 在 Details (詳細資訊)標籤的 Server-side encrption settings (伺服器端加密設定) 底下，選擇 Edit (編輯)。

   在儲存貯體金鑰之下，您會看到物件的 S3 儲存貯體金鑰設定。您無法編輯此設定。

使用 AWS CLI

傳回儲存貯體層級 S3 儲存貯體金鑰設定

若要使用此範例，請以您自己的資訊取代每個 *user input placeholder*。

```
aws s3api get-bucket-encryption --bucket example-s3-bucket1
```

若要取得更多資訊，請參閱《指AWS CLI 令參考》get-bucket-encryption中的。

傳回 S3 儲存貯體金鑰的物件層級設定

若要使用此範例，請以您自己的資訊取代每個 *user input placeholder*。

```
aws s3api head-object --bucket example-s3-bucket1 --key my_images.tar.bz2
```

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 head-object。

使用 REST API

傳回儲存貯體層級 S3 儲存貯體金鑰設定

若要傳回儲存貯體的加密資訊，包括 S3 儲存貯體金鑰的設定，請使用 `GetBucketEncryption` 操作。在具有 `BucketKeyEnabled` 設定的 `ServerSideEncryptionConfiguration` 元素中，會在

回應內文傳回 S3 儲存貯體金鑰。如需詳細資訊，請GetBucketEncryption參閱 Amazon S3 API 參考中的。

傳回 S3 儲存貯體金鑰的物件層級設定

若要傳回物件的 S3 儲存貯體金鑰狀態，請使用 `HeadObject` 操作。`HeadObject` 傳回 `x-amz-server-side-encryption-bucket-key-enabled` 回應標頭以顯示是否以為物件啟用或停用 S3 儲存貯體金鑰。如需詳細資訊，請HeadObject參閱 Amazon S3 API 參考中的。

如果已針對物件設定 S3 儲存貯體金鑰，則下列 API 操作也會傳回 `x-amz-server-side-encryption-bucket-key-enabled` 回應標頭：

- PutObject
- PostObject
- CopyObject
- CreateMultipartUpload
- UploadPartCopy
- UploadPart
- CompleteMultipartUpload
- GetObject

搭配 AWS KMS 金鑰 (DSSE-KMS) 使用雙層伺服器端加密

搭配 () 金鑰 AWS Key Management Service (DSSE-KMS AWS KMS) 使用雙層伺服器端加密，會在物件上傳到 Amazon S3 時，將兩層加密套用至物件。DSSE-KMS 可協助您更輕鬆地達成合規標準，這些標準會要求您將多層加密套用至資料，並完全掌控您的加密金鑰。

當您將 DSSE-KMS 與 Amazon S3 儲存貯體搭配使用時，金 AWS KMS 鑰必須與儲存貯體位於相同的區域。此外，當物件要求 DSSE-KMS 時，做為物件中繼資料一部分的 S3 總和檢查會以加密形式存放。如需總和檢查的詳細資訊，請參閱 檢查物件完整性。

使用 DSSE-KMS 和需要支付額外費用。 AWS KMS keys如需詳細了解 DSSE-KMS 定價，請參閱《AWS Key Management Service 開發人員指南》中的 AWS KMS key 概念和 AWS KMS 定價。

> **Note**
>
> DSSE-KMS 不支援 S3 儲存貯體金鑰。

需要使用 AWS KMS keys (DSSE-KMS) 進行雙層伺服器端加密

若要在特定 Amazon S3 儲存貯體中要求所有物件的雙層伺服器端加密，您可以使用儲存貯體政策。例如，如果請求不包含要求含 DSSE-KMS 之伺服器端加密的 `x-amz-server-side-encryption` 標頭，則下列儲存貯體政策會拒絕向所有人上傳物件 (`s3:PutObject`) 的許可。

```
{
            "Version":"2012-10-17",
            "Id":"PutObjectPolicy",
            "Statement":[{
                  "Sid":"DenyUnEncryptedObjectUploads",
                  "Effect":"Deny",
                  "Principal":"*",
                  "Action":"s3:PutObject",
                  "Resource":"arn:aws:s3:::example-s3-bucket1/*",
                  "Condition":{
                     "StringNotEquals":{
                        "s3:x-amz-server-side-encryption":"aws:kms:dsse"
                     }
                  }
               }
            ]
         }
```

主題

- 搭配 AWS KMS 金鑰 (DSSE-KMS) 指定雙層伺服器端加密

搭配 AWS KMS 金鑰 (DSSE-KMS) 指定雙層伺服器端加密

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

根據預設，所有 Amazon S3 儲存貯體都設定了加密，所有上傳到 S3 儲存貯體的新物件都會在靜態時自動加密。伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 是 Amazon S3 中每個儲存貯體的預設加

密組態。若要使用不同類型的加密，您可以指定 S3 PUT 請求中要使用的伺服器端加密類型，也可以在目的地儲存貯體中設定預設加密組態。

如果您想要在要PUT求中指定不同的加密類型，您可以使用伺服器端加密 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS)、使用金鑰的雙層伺服器端加密 (DSSE-KMS)，或使用客戶提供的金 AWS KMS 鑰 (SSE-C) 進行伺服器端加密。若您想在目的地儲存貯體中設定不同的預設加密組態，您可以使用 SSE-KMS 或 DSSE-KMS。

上傳新物件或複製現有物件時，您都可以套用加密。

您可以使用 Amazon S3 主控台、Amazon S3 REST API 和 AWS Command Line Interface (AWS CLI) 來指定 DSSE-KMS。如需詳細資訊，請參閱下列主題。

> **Note**
>
> 您可以 AWS KMS keys 在 Amazon S3 中使用多區域。但是，Amazon S3 目前將多區域金鑰視為單區域金鑰，並且不使用金鑰的多區域功能。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的使用多區域金鑰。

> **Note**
>
> 若您想要使用其他帳戶的 KMS 金鑰，您必須具有該金鑰的使用權限。如需詳細了解 KMS 金鑰跨帳戶權限，請參閱《AWS Key Management Service 開發人員指南》中的建立其他帳戶可使用的 KMS 金鑰。

使用 S3 主控台

本節說明如何使用 Amazon S3 主控台設定或變更物件的加密類型，以使用含 AWS Key Management Service (AWS KMS) 金鑰 (DSSE-KMS) 的雙層伺服器端加密。

> **Note**
>
> - 如果您變更物件的加密方式，則會建立新物件來取代舊物件。如果啟用 S3 版本控制，則系統會建立物件的新版本，且現有物件會變成較舊的版本。變更屬性的角色也會成為新物件 (或物件版本) 的擁有者。

- 如果您變更具有使用者定義標籤之物件的加密類型，您必須擁有`s3:GetObjectTagging`權限。如果您要變更沒有使用者定義標籤但大小超過 16 MB 的物件的加密類型，您也必須擁有`s3:GetObjectTagging`權限。

  如果目的地儲存貯體政策拒絕該`s3:GetObjectTagging`動作，則會更新物件的加密類型，但使用者定義的標籤會從物件中移除，而且您會收到錯誤訊息。

新增或變更物件的加密

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在儲存貯體清單中，選擇儲存貯體的名稱，其中包含您要加密的物件。
4. 在物件清單中，選取您欲新增或變更加密的物件核取方塊。

   系統會顯示物件的詳細資訊頁面，其中包含數個區段，顯示物件的屬性。
5. 選擇屬性索引標籤。
6. 向下捲動至預設加密區段，然後選擇編輯。

   編輯預設加密頁面隨即開啟。
7. 在 [加密類型] 下，選擇 [含 AWS Key Management Service 金鑰的雙層伺服器端加密 (DSSE-KMS)]。
8. 在 AWS KMS 金鑰之下，執行下列其中一個動作，以選擇 KMS 金鑰：

   - 若要從可用的 KMS 金鑰清單中選擇，請選擇從 AWS KMS keys中選擇，然後從可用金鑰清單中選擇您的 KMS 金鑰。

     AWS 受管金鑰 (`aws/s3`) 和您的客戶管理金鑰都會出現在此清單中。如需詳細了解客戶受管金鑰，請參閱《AWS Key Management Service 開發人員指南》中的客戶金鑰和 AWS 金鑰。
   - 若要輸入 KMS 金鑰 ARN，請選擇 [輸入 AWS KMS key ARN]，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。
   - 若要在 AWS KMS 主控台中建立新的客戶管理金鑰，請選擇 [建立 KMS 金鑰]。

     如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。

> **Important**
>
> 您只能使用在與儲存貯體相同的 AWS 區域 中可用的 KMS 金鑰。Amazon S3 主控台僅會列出與儲存貯體位於相同區域的前 100 個 KMS 金鑰。若要使用未列出的 KMS 金鑰，必須輸入 KMS 金鑰 ARN。若您想要使用其他帳戶的 KMS 金鑰，您必須先具有該金鑰的使用權限，然後輸入 KMS 金鑰 ARN。
>
> Amazon S3 僅支援對稱加密 KMS 金鑰，而不支援非對稱 KMS 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的識別非對稱 KMS 金鑰。

9. 在儲存貯體金鑰下，選擇停用。DSSE-KMS 不支援 S3 儲存貯體金鑰。
10. 選擇 Save changes (儲存變更)。

> **Note**
>
> 此動作會將加密套用至所有指定的物件。加密資料夾時，請等待儲存作業完成，然後再將新物件新增至資料夾。

使用 REST API

當您建立物件 (亦即，當您上傳新物件或複製現有物件時) 時，您可以指定使用雙層伺服器端加密 AWS KMS keys (DSSE-KMS) 來加密資料。若要執行這項操作，請將 `x-amz-server-side-encryption` 標頭新增至要求。將標頭的值設為加密演算法 `aws:kms:dsse`。Amazon S3 會傳回回應標頭 `x-amz-server-side-encryption`，確認已使用 DSSE-KMS 加密存放物件。

如果您使用 `aws:kms:dsse` 的值指定 `x-amz-server-side-encryption` 標頭，也可以使用下列要求標頭：

- `x-amz-server-side-encryption-aws-kms-key-id: SSEKMSKeyId`
- `x-amz-server-side-encryption-context: SSEKMSEncryptionContext`

主題

- 支援 DSSE-KMS 的 Amazon S3 REST API 操作
- 加密內容 (x-amz-server-side-encryption-context)
- AWS KMS 金鑰識別碼 (x-amz-server-side-encryption-aws-kms-key-id)

支援 DSSE-KMS 的 Amazon S3 REST API 操作

下列 REST API 操作接受 `x-amz-server-side-encryption`、`x-amz-server-side-encryption-aws-kms-key-id` 和 `x-amz-server-side-encryption-context` 請求標頭。

- PutObject – 使用 PUT API 操作上傳資料時，您可以指定這些請求標頭。
- CopyObject – 複製物件時，您會同時有來源物件與目標物件。當您使用 `CopyObject` 操作傳遞 DSSE-KMS 標頭時，這些標頭只會套用至目標物件。複製現有物件時，除非明確地要求伺服器端加密，否則無論來源物件是否經過加密，都不會加密目標物件。
- POST 物件 – 使用 `POST` 操作上傳物件時，您會提供與表單欄位中相同的資訊，而不是請求標頭。
- CreateMultipartUpload – 透過分段上傳來上傳大型物件時，您可以在 `CreateMultipartUpload` 請求中指定這些標頭。

使用伺服器端加密存放物件時，下列 REST API 操作的回應標頭會傳回 `x-amz-server-side-encryption` 標頭。

- PutObject
- CopyObject
- POST 物件
- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- GetObject
- HeadObject

> **Important**
>
> - 如果您 AWS KMS 未使用安全通訊端層 (SSL)、傳輸層安全性 (TLS) 或簽章版本 4 來建立受到保護的物件，則所有物件`GET`和`PUT`要求受到失敗保護的物件。
> - 如果您的物件使用 DSSE-KMS，請勿傳送 `GET` 請求與 `HEAD` 請求的加密請求標頭，否則您會收到 HTTP 400 (錯誤的請求) 錯誤。

加密內容 (**x-amz-server-side-encryption-context**)

如果您指定 x-amz-server-side-encryption:aws:kms:dsse，則 Amazon S3 API 支援具有 x-amz-server-side-encryption-context 標頭的加密內容。加密內容是一組金鑰值對，其中包含資料的其他相關內容資訊。

Amazon S3 會自動使用物件的 Amazon Resource Name (ARN) 作為加密內容配對，例如 arn:aws:s3:::*object_ARN*。

您可以使用 x-amz-server-side-encryption-context 標頭來提供其他加密內容對。不過，加密內容未加密，因此請確保其中不包含敏感資訊。Amazon S3 會一起存放此附加金鑰對與預設加密內容。

如需 Amazon S3 中加密內容的相關資訊，請參閱 加密內容。如需有關加密內容的更多資訊，請參閱《AWS Key Management Service 開發人員指南》中的 AWS Key Management Service 概念：加密內容。

AWS KMS 金鑰識別碼 (**x-amz-server-side-encryption-aws-kms-key-id**)

您可以使用 x-amz-server-side-encryption-aws-kms-key-id 標頭來指定用來保護資料之客戶受管金鑰的 ID。如果您指定x-amz-server-side-encryption:aws:kms:dsse標頭但未提供標x-amz-server-side-encryption-aws-kms-key-id頭，Amazon S3 會使用 AWS 受管金鑰 (aws/s3) 來保護資料。如果您想要使用客戶受管的金鑰，則必須提供客戶受管金鑰的 x-amz-server-side-encryption-aws-kms-key-id 標頭。

> **Important**
>
> 當您在 Amazon S3 中使 AWS KMS key 用伺服器端加密時，您必須選擇對稱加密 KMS 金鑰。Amazon S3 只支援對稱加密 KMS 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的對稱加密 KMS 金鑰。

使用 AWS CLI

當上傳新物件或複製現有物件時，您可以指定使用 DSSE-KMS 來加密資料。若要執行這項操作，請將 --server-side-encryption aws:kms:dsse 參數新增至請求。使用 --ssekms-key-id *example-key-id* 參數來新增您已建立的客戶受管 AWS KMS 金鑰。如果您指定--server-side-encryption aws:kms:dsse但未提供 AWS KMS 金鑰 ID，則 Amazon S3 將使用受 AWS 管金鑰 (aws/s3)。

```
aws s3api put-object --bucket DOC-EXAMPLE-BUCKET --key example-object-key --server-
side-encryption aws:kms:dsse --ssekms-key-id example-key-id --body filepath
```

您可以將未加密的物件加密以使用 DSSE-KMS，方法是將物件複製回原位。

```
aws s3api copy-object --bucket DOC-EXAMPLE-BUCKET --key example-object-key --
body filepath --bucket DOC-EXAMPLE-BUCKET --key example-object-key --sse aws:kms:dsse
 --sse-kms-key-id example-key-id --body filepath
```

搭配客戶提供的金鑰 (SSE-C) 使用伺服器端加密

伺服器端加密是有關保護靜態資料。伺服器端加密只會加密物件資料，非物件中繼資料。透過使用伺服器端加密搭配客戶提供的金鑰 (SSE-C)，您可以儲存使用自己的加密金鑰加密的資料。如果您提供的加密金鑰作為請求的一部分，Amazon S3 會在資料寫入磁碟時管理資料加密，以及在您存取物件時管理資料解密。因此，您不需要維護任何程式碼來執行資料加密與解密。您只需要管理自己提供的加密金鑰即可。

當您上傳物件時，Amazon S3 會使用您提供的加密金鑰將 AES-256 加密套用至您的資料。然後，Amazon S3 會從記憶體中移除加密金鑰。當您擷取物件時，您必須在要求中提供相同的加密金鑰。Amazon S3 會先驗證您提供的加密金鑰是否相符，然後對物件進行解密，再將物件資料傳回給您。

使用 SSE-C 無須額外付費。不過，請求設定和使用 SSE-C 會產生標準的 Amazon S3 請求費用。如需定價的資訊，請參閱 Amazon S3 定價。

> **Note**
>
> Amazon S3 不會存放您提供的加密金鑰。相反地，它會存放加密金鑰的隨機雜湊訊息驗證碼 (HMAC) Salt 值，以驗證未來的請求。HMAC Salt 值不可用來衍生加密金鑰的值，或用來對加密物件的內容進行解密。換句話說，如果您遺失加密金鑰，則會遺失物件。

S3 複寫支援使用 SSE-C 加密的物件。如需複寫加密物件的詳細資訊，請參閱 the section called “複寫加密的物件”。

如需有關 SSE-C 的詳細資訊，請參閱以下主題。

主題

- SSE-C 概觀

- SSE-C 的要求和限制
- 已預先簽章的 URL 和 SSE-C
- 使用客戶提供金鑰 (SSC-C) 指定伺服器端加密

SSE-C 概觀

本節提供 SSE-C 的概觀。使用 SSE-C 時，請謹記下列考量。

- 您必須使用 HTTPS。

> **Important**
>
> 使用 SSE-C 時，Amazon S3 會拒絕所有透過 HTTP 提出的請求。為安全起見，建議任何錯誤地透過 HTTP 傳送的金鑰皆視為已遭洩漏。請捨棄該金鑰，並適當地輪換。

- 回應中的實體標籤 (ETag) 不是物件資料的 MD5 雜湊。
- 您會管理哪個加密金鑰用來加密哪個物件的對應。Amazon S3 不會存放加密金鑰。您會負責追蹤針對哪個物件提供哪個加密金鑰。
  - 若已對儲存貯體啟用版本控制，使用此功能上傳的每個物件版本都會有自己的加密金鑰。您會負責追蹤針對哪個物件版本使用了哪個加密金鑰。
  - 由於您是在用戶端管理加密金鑰，因此您會在用戶端管理任何額外的保護措施，例如金鑰輪替。

> **Warning**
>
> 如果您遺失加密金鑰，針對沒有其加密金鑰的物件所提出的任何 GET 要求都會失敗，而且您會遺失物件。

SSE-C 的要求和限制

若要針對特定 Amazon S3 儲存貯體中的所有物件要求 SSE-C，您可以使用儲存貯體政策。

例如，下列儲存貯體政策拒絕所有不包含請求 SSE-C 之 `x-amz-server-side-encryption-customer-algorithm` 標題請求的上傳物件 (`s3:PutObject`) 許可。

```
{
    "Version": "2012-10-17",
    "Id": "PutObjectPolicy",
```

```
    "Statement": [
        {
            "Sid": "RequireSSECObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::example-s3-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "true"
                }
            }
        }
    ]
}
```

您也可以使用政策，限制特定 Amazon S3 儲存貯體中所有物件的伺服器端加密。例如，如果請求包含請求 SSE-KMS 的 `x-amz-server-side-encryption-customer-algorithm` 標頭，則下列儲存貯體政策會拒絕向所有人上傳物件 (`s3:PutObject`) 的許可。

```
{
    "Version": "2012-10-17",
    "Id": "PutObjectPolicy",
    "Statement": [
        {
            "Sid": "RestrictSSECObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::example-s3-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "false"
                }
            }
        }
    ]
}
```

> **Important**
>
> 如果您使用儲存貯體政策要求開啟 SSE-Cs3:PutObject，則必須在所有分段上傳要求 (CreateMultipartUpload UploadPart、和) 中包含x-amz-server-side-encryption-customer-algorithm標頭。CompleteMultipartUpload

已預先簽章的 URL 和 SSE-C

您可以產生預先簽章的 URL，其可以用於上傳新的物件、擷取現有的物件或擷取物件中繼資料等操作。預先簽章的 URL 支援如下的 SSE-C：

- 建立預先簽章的 URL 時，您必須在簽章計算中使用 `x-amz-server-side-encryption-customer-algorithm` 標頭來指定演算法。
- 使用預先簽章的 URL 來上傳新的物件、擷取現有的物件或只擷取物件中繼資料時，您必須提供用戶端應用程式請求中的所有加密標頭。

> **Note**
>
> 針對非 SSE-C 物件，您可以產生預先簽章的 URL，並將該 URL 直接貼入至瀏覽器以存取資料。
>
> 不過，您無法針對 SSE-C 物件執行此操作，因為除了預先簽章的 URL 之外，您亦須包含 SSE-C 物件專屬的 HTTP 標頭。因此，您只能以程式設計方式針對 SSE-C 物件使用預先簽章的 URL。

如需預先簽章的 URL 詳細資訊，請參閱the section called “使用預先簽章的 URL”。

使用客戶提供金鑰 (SSC-C) 指定伺服器端加密

在使用 REST API 建立物件時，您可以使用客戶提供的金鑰 (SSE-C) 來指定伺服器端加密。當您使用 SSE-C 時，必須使用下列要求標頭來提供加密金鑰資訊。

名稱	描述
`x-amz-server-side-encryption`	使用此標頭可指定加密演算法。標頭值必須為 AES256。

名稱	描述
`-customer-algorithm`	
`x-amz-server-side-encryption-customer-key`	使用此標頭可提供 256 位元 base64 編碼加密金鑰，讓 Amazon S3 用來對資料進行加密或解密。
`x-amz-server-side-encryption-customer-key-MD5`	使用此標頭可提供採用 RFC 1321 之 base64 編碼 128 位元 MD5 Digest 的加密金鑰。Amazon S3 使用此標頭來進行訊息完整性檢查，以確保加密金鑰傳輸無誤。

您可以使用 AWS SDK 包裝函式庫將這些標頭新增至您的要求。如果需要，您可以直接在應用程式中進行 Amazon S3 REST API 呼叫。

> **Note**
>
> 您無法使用 Amazon S3 主控台上傳物件及請求 SSE-C。您也無法使用主控台來更新使用 SSE-C 存放的現有物件 (例如變更儲存方案或新增中繼資料)。

使用 REST API

支援 SSE-C 的 Amazon S3 REST API

下列 Amazon S3 API 支援伺服器端加密搭配客戶提供的加密金鑰 (SSE-C)。

- GET 操作 – 使用 GET API 擷取物件時 (請參閱 GET 物件)，您可以指定這些請求標頭。
- HEAD 操作 – 若要使用 HEAD API 擷取物件中繼資料 (請參閱 HEAD 物件)，您可以指定這些請求標頭。
- PUT 操作 – 使用 PUT API 上傳資料時 (請參閱 PUT 物件)，您可以指定這些請求標頭。
- 分段上傳 – 使用分段上傳 API 上傳大型物件時，您可以指定這些標頭。您可以在啟動請求 (請參閱啟動分段上傳) 及每個後續片段上傳請求 (請參閱上傳片段或上傳片段 - 複製) 中指定這些標頭。每個部分上傳要求的加密資訊，必須與您在啟動分段上傳要求中所提供的加密資訊相同。
- POST 操作 – 使用 POST 操作上傳物件時 (請參閱 POST 物件)，請提供與表單欄位中相同的資訊，而不是請求標頭。

- 複製操作 – 當您複製物件時 (請參閱 PUT 物件 - 複製)，您要同時有來源物件與目標物件。
  - 如果您想要使用具有 AWS 受管理金鑰的伺服器端加密來加密目標物件，則必須提供要x-amz-server-side-encryption 求標頭。
  - 如果您想要使用 SSE-C 對目標物件進行加密，您必須使用上一個表格中所述的三個標頭來提供加密資訊。
  - 如果來源物件是使用 SSE-C 加密，您必須使用下列標頭來提供加密金鑰資訊，讓 Amazon S3 可以先解密物件，再進行複製。

名稱	描述
x-amz-copy-source-server-side-encryption-customer-algorithm	包含此標頭可指定 Amazon S3 應該用來解密來源物件的演算法。此值必須為 AES256。
x-amz-copy-source-server-side-encryption-customer-key	包含此標頭可提供 base64 編碼加密金鑰，讓 Amazon S3 用來對來源物件進行解密。此加密金鑰必須是您建立來源物件時提供給 Amazon S3 的加密金鑰。否則，Amazon S3 無法解密物件。
x-amz-copy-source-server-side-encryption-customer-key-MD5	包含此標頭可提供採用 RFC 1321 之 base64 編碼 128 位元 MD5 摘要的加密金鑰。

使用開 AWS 發套件為 PUT、GET、標頭和複製作業指定 SSE-C

下列範例說明如何用客戶提供金鑰 (SSE-C)，要求為物件進行伺服器端加密。這些範例會執行下列操作。每個操作示範如何在要求中指定 SSE-C 相關標頭：

- 放置物件 – 使用客戶提供的加密金鑰上傳物件並請求伺服器端加密。
- 取得物件 – 下載前一個步驟中所上傳的物件。在此請求中，請您提供在您上傳物件時所提供的相同加密資訊。Amazon S3 需要此資訊來解密物件，才能將物件傳回給您。
- 取得物件中繼資料 – 擷取物件的中繼資料。請您提供物件建立時，使用的加密資訊。

- 複製物件 – 建立先前上傳物件的複本。因為來源物件是使用 SSE-C 所存放，所以您必須在複製要求中提供其加密資訊。根據預設，只有在您明確請求加密時，Amazon S3 才會加密物件的複本。此範例指示 Amazon S3 存放加密的物件複本。

Java

> **Note**
>
> 此範例示範如何以單一操作上傳物件。使用分段上傳 API 上傳大型物件時，請您提供如此範例一樣的加密資訊。如需使用分段上傳的範例 AWS SDK for Java，請參閱使用分段上傳來上傳物件。

新增要求加密資訊，您可包含 `SSECustomerKey` 在您的要求中。如需有關 `SSECustomerKey` 類別的詳細資訊，請參閱 REST API 一節。

如需有關 SSE-C 的詳細資訊，請參閱 搭配客戶提供的金鑰 (SSE-C) 使用伺服器端加密。如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

Example

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import javax.crypto.KeyGenerator;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

public class ServerSideEncryptionUsingClientSideEncryptionKey {
    private static SSECustomerKey SSE_KEY;
    private static AmazonS3 S3_CLIENT;
```

```
   private static KeyGenerator KEY_GENERATOR;

   public static void main(String[] args) throws IOException,
NoSuchAlgorithmException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String uploadFileName = "*** File path ***";
        String targetKeyName = "*** Target key name ***";

        // Create an encryption key.
        KEY_GENERATOR = KeyGenerator.getInstance("AES");
        KEY_GENERATOR.init(256, new SecureRandom());
        SSE_KEY = new SSECustomerKey(KEY_GENERATOR.generateKey());

        try {
            S3_CLIENT = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Upload an object.
            uploadObject(bucketName, keyName, new File(uploadFileName));

            // Download the object.
            downloadObject(bucketName, keyName);

            // Verify that the object is properly encrypted by attempting to
retrieve it
            // using the encryption key.
            retrieveObjectMetadata(bucketName, keyName);

            // Copy the object into a new object that also uses SSE-C.
            copyObject(bucketName, keyName, targetKeyName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
   }
```

```
   private static void uploadObject(String bucketName, String keyName, File file) {
       PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName,
file).withSSECustomerKey(SSE_KEY);
       S3_CLIENT.putObject(putRequest);
       System.out.println("Object uploaded");
   }

   private static void downloadObject(String bucketName, String keyName) throws
IOException {
       GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName,
keyName).withSSECustomerKey(SSE_KEY);
       S3Object object = S3_CLIENT.getObject(getObjectRequest);

       System.out.println("Object content: ");
       displayTextInputStream(object.getObjectContent());
   }

   private static void retrieveObjectMetadata(String bucketName, String keyName) {
       GetObjectMetadataRequest getMetadataRequest = new
GetObjectMetadataRequest(bucketName, keyName)
               .withSSECustomerKey(SSE_KEY);
       ObjectMetadata objectMetadata =
S3_CLIENT.getObjectMetadata(getMetadataRequest);
       System.out.println("Metadata retrieved. Object size: " +
objectMetadata.getContentLength());
   }

   private static void copyObject(String bucketName, String keyName, String
targetKeyName)
           throws NoSuchAlgorithmException {
       // Create a new encryption key for target so that the target is saved using
       // SSE-C.
       SSECustomerKey newSSEKey = new SSECustomerKey(KEY_GENERATOR.generateKey());

       CopyObjectRequest copyRequest = new CopyObjectRequest(bucketName, keyName,
bucketName, targetKeyName)
               .withSourceSSECustomerKey(SSE_KEY)
               .withDestinationSSECustomerKey(newSSEKey);

       S3_CLIENT.copyObject(copyRequest);
       System.out.println("Object copied");
   }
```

```
    private static void displayTextInputStream(S3ObjectInputStream input) throws
 IOException {
        // Read one line at a time from the input stream and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        String line;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
}
```

.NET

> **Note**
>
> 如需更說使用分段上傳 API 大型物件的範例，請參閱 使用分段上傳來上傳物件 和 使用 AWS 軟體開發套件 (低階 API)。

如需有關 SSE-C 的詳細資訊，請參閱 搭配客戶提供的金鑰 (SSE-C) 使用伺服器端加密。如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSEClientEncryptionKeyObjectOperationsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for new object created ***";
        private const string copyTargetKeyName = "*** key name for object copy ***";
        // Specify your bucket region (an example region is shown).
```

```
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            ObjectOpsUsingClientEncryptionKeyAsync().Wait();
        }
        private static async Task ObjectOpsUsingClientEncryptionKeyAsync()
        {
            try
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // 1. Upload the object.
                PutObjectRequest putObjectRequest = await
UploadObjectAsync(base64Key);
                // 2. Download the object and verify that its contents matches what
you uploaded.
                await DownloadObjectAsync(base64Key, putObjectRequest);
                // 3. Get object metadata and verify that the object uses AES-256
encryption.
                await GetObjectMetadataAsync(base64Key);
                // 4. Copy both the source and target objects using server-side
encryption with
                //    a customer-provided encryption key.
                await CopyObjectAsync(aesEncryption, base64Key);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }
```

```
        private static async Task<PutObjectRequest> UploadObjectAsync(string
base64Key)
        {
            PutObjectRequest putObjectRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };
            PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
            return putObjectRequest;
        }
        private static async Task DownloadObjectAsync(string base64Key,
PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                // Provide encryption information for the object stored in Amazon
S3.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (String.Compare(putObjectRequest.ContentBody, content) == 0)
                    Console.WriteLine("Object content is same as we uploaded");
                else
                    Console.WriteLine("Error...Object content is not same.");

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
```

```
                    Console.WriteLine("Object encryption method is AES256, same as
we set");
                else
                    Console.WriteLine("Error...Object encryption method is not the
same as AES256 we set");

                // Assert.AreEqual(putObjectRequest.ContentBody, content);
                // Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getResponse.ServerSideEncryptionCustomerMethod);
            }
        }
        private static async Task GetObjectMetadataAsync(string base64Key)
        {
            GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
            Console.WriteLine("The object metadata show encryption method used is:
{0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
            // Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }
        private static async Task CopyObjectAsync(Aes aesEncryption, string
base64Key)
        {
            aesEncryption.GenerateKey();
            string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

            CopyObjectRequest copyRequest = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = keyName,
                DestinationBucket = bucketName,
```

```
                DestinationKey = copyTargetKeyName,
                // Information about the source object's encryption.
                CopySourceServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,
                // Information about the target object's encryption.
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = copyBase64Key
            };
            await client.CopyObjectAsync(copyRequest);
        }
    }
}
```

使用 AWS SDK 為多部分上傳指定 SSE-C

上節中的範例示範如何以 PUT、GET、Head 和 Copy 操作要求經由客戶提供加密金鑰的伺服器端加密 (SSE-C)。本節說明其他支援 SSE-C 的 Amazon S3 API。

Java

若要上傳大型物件，您可以使用分段上傳 API (請參閱「使用分段上傳來上傳和複製物件」)。您可以使用高階或低階 API 來上傳大型物件。這些 API 支援要求中的加密相關標頭。

- 使用高階 `TransferManager` API 時，請您在 `PutObjectRequest` (請參閱 使用分段上傳來上傳物件) 所提供加密特定標頭。
- 使用低階 API 時，請您在 `InitiateMultipartUploadRequest` 提供加密關聯資訊，遵照 `UploadPartRequest` 中每一個的相同加密資訊。您不需要在 `CompleteMultipartUploadRequest` 提供任何加密特定標頭。如需範例，請參閱 使用 AWS 軟體開發套件 (低階 API)。

下列範例使用 `TransferManager` 建立物件，並示範如何提供 SSE-C 相關資訊。此範例執行下列操作：

- 使用 `TransferManager.upload()` 方法建立物件。在 `PutObjectRequest` 執行個體中，請您提供加密金鑰資訊以進行請求。Amazon S3 會使用客戶提供的金鑰來加密物件。
- 呼叫 `TransferManager.copy()` 方法，以建立物件的複本。此範例指示 Amazon S3 使用新的 `SSECustomerKey` 來加密物件複本。因為來源物件使用 SSE-C 加密，所以

CopyObjectRequest 也會提供來源物件的加密金鑰，讓 Amazon S3 可以先解密物件，再進行複製。

Example

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.SSECustomerKey;
import com.amazonaws.services.s3.transfer.Copy;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import javax.crypto.KeyGenerator;
import java.io.File;
import java.security.SecureRandom;

public class ServerSideEncryptionCopyObjectUsingHLwithSSEC {

    public static void main(String[] args) throws Exception {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String fileToUpload = "*** File path ***";
        String keyName = "*** New object key name ***";
        String targetKeyName = "*** Key name for object copy ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();
            TransferManager tm = TransferManagerBuilder.standard()
                    .withS3Client(s3Client)
                    .build();

            // Create an object from a file.
```

```
            PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName,
keyName, new File(fileToUpload));

            // Create an encryption key.
            KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
            keyGenerator.init(256, new SecureRandom());
            SSECustomerKey sseCustomerEncryptionKey = new
SSECustomerKey(keyGenerator.generateKey());

            // Upload the object. TransferManager uploads asynchronously, so this
call
            // returns immediately.
            putObjectRequest.setSSECustomerKey(sseCustomerEncryptionKey);
            Upload upload = tm.upload(putObjectRequest);

            // Optionally, wait for the upload to finish before continuing.
            upload.waitForCompletion();
            System.out.println("Object created.");

            // Copy the object and store the copy using SSE-C with a new key.
            CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucketName,
keyName, bucketName, targetKeyName);
            SSECustomerKey sseTargetObjectEncryptionKey = new
SSECustomerKey(keyGenerator.generateKey());
            copyObjectRequest.setSourceSSECustomerKey(sseCustomerEncryptionKey);

copyObjectRequest.setDestinationSSECustomerKey(sseTargetObjectEncryptionKey);

            // Copy the object. TransferManager copies asynchronously, so this call
returns
            // immediately.
            Copy copy = tm.copy(copyObjectRequest);

            // Optionally, wait for the upload to finish before continuing.
            copy.waitForCompletion();
            System.out.println("Copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
```

```
        }
    }
}
```

.NET

若要上傳大型物件，您可以使用多部分上傳 API (請參閱使用分段上傳來上傳和複製物件)。 AWS SDK for .NET 提供高階或低階 API 來上傳大型物件。這些 API 支援要求中的加密相關標頭。

- 使用高階 `Transfer-Utility` API 時，您在 `TransferUtilityUploadRequest` 所提供的加密特定標頭，如下所示。如需程式碼範例，請參閱「使用分段上傳來上傳物件」。

```
TransferUtilityUploadRequest request = new TransferUtilityUploadRequest()
{
    FilePath = filePath,
    BucketName = existingBucketName,
    Key = keyName,
    // Provide encryption information.
    ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
    ServerSideEncryptionCustomerProvidedKey = base64Key,
};
```

- 使用低階 API 時，請提供您在啟動分段上傳要求中的加密相關資訊，後面接著後續分段上傳要求中的相同加密資訊。您不需要在完整的分段上傳要求中提供任何加密特定標頭。如需範例，請參閱 使用 AWS 軟體開發套件 (低階 API)。

  以下為建立現有大型物件複本的低階分段上傳範例。在此範例中，要複製的物件會使用 SSE-C 存放在 Amazon S3 中，而您也想要使用 SSE-C 儲存目標物件。在此範例中，您要執行以下動作：

  - 提供加密金鑰與相關資訊，以啟動分段上傳要求。
  - 在 `CopyPartRequest` 中提供來源與目標物件加密金鑰以及相關資訊。
  - 擷取物件中繼資料，以取得要複製之來源物件的大小。
  - 將物件分成 5 MB 部分來上傳。

  Example

```
using Amazon;
using Amazon.S3;
```

```
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSECLowLevelMPUcopyObjectTest
    {
        private const string existingBucketName = "*** bucket name ***";
        private const string sourceKeyName      = "*** source object key name
 ***";
        private const string targetKeyName      = "*** key name for the target
 object ***";
        private const string filePath           = @"*** file path ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CopyObjClientEncryptionKeyAsync().Wait();
        }

        private static async Task CopyObjClientEncryptionKeyAsync()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);

            await CreateSampleObjUsingClientEncryptionKeyAsync(base64Key,
 s3Client);

            await CopyObjectAsync(s3Client, base64Key);
        }
        private static async Task CopyObjectAsync(IAmazonS3 s3Client, string
 base64Key)
        {
            List<CopyPartResponse> uploadResponses = new List<CopyPartResponse>();
```

```
            // 1. Initialize.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = targetKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // 2. Upload Parts.
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB
            long firstByte = 0;
            long lastByte = partSize;

            try
            {
                // First find source object size. Because object is stored
encrypted with
                // customer provided key you need to provide encryption
information in your request.
                GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest()
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                    ServerSideEncryptionCustomerProvidedKey = base64Key // " *
**source object encryption key ***"
                };

                GetObjectMetadataResponse getObjectMetadataResponse = await
s3Client.GetObjectMetadataAsync(getObjectMetadataRequest);

                long filePosition = 0;
                for (int i = 1; filePosition <
getObjectMetadataResponse.ContentLength; i++)
                {
                    CopyPartRequest copyPartRequest = new CopyPartRequest
```

```
                    {
                        UploadId = initResponse.UploadId,
                        // Source.
                        SourceBucket = existingBucketName,
                        SourceKey = sourceKeyName,
                        // Source object is stored using SSE-C. Provide encryption
information.
                        CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        CopySourceServerSideEncryptionCustomerProvidedKey =
base64Key, //"***source object encryption key ***",
                        FirstByte = firstByte,
                        // If the last part is smaller then our normal part size
then use the remaining size.
                        LastByte = lastByte >
getObjectMetadataResponse.ContentLength ?
                            getObjectMetadataResponse.ContentLength - 1 :
lastByte,

                        // Target.
                        DestinationBucket = existingBucketName,
                        DestinationKey = targetKeyName,
                        PartNumber = i,
                        // Encryption information for the target object.
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key
                    };
                    uploadResponses.Add(await
s3Client.CopyPartAsync(copyPartRequest));
                    filePosition += partSize;
                    firstByte += partSize;
                    lastByte += partSize;
                }

                // Step 3: complete.
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = targetKeyName,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(uploadResponses);
```

```
                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine("Exception occurred: {0}", exception.Message);
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = targetKeyName,
                    UploadId = initResponse.UploadId
                };
                s3Client.AbortMultipartUpload(abortMPURequest);
            }
        }
        private static async Task
CreateSampleObjUsingClientEncryptionKeyAsync(string base64Key, IAmazonS3
s3Client)
        {
            // List to store upload part responses.
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

            // 1. Initialize.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            InitiateMultipartUploadResponse initResponse =
               await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // 2. Upload Parts.
            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
```

```
            {
                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                    {
                        BucketName = existingBucketName,
                        Key = sourceKeyName,
                        UploadId = initResponse.UploadId,
                        PartNumber = i,
                        PartSize = partSize,
                        FilePosition = filePosition,
                        FilePath = filePath,
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key
                    };

                    // Upload part and add response to our list.
                    uploadResponses.Add(await
s3Client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                // Step 3: complete.
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                    //PartETags = new List<PartETag>(uploadResponses)

                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);

            }
            catch (Exception exception)
            {
                Console.WriteLine("Exception occurred: {0}", exception.Message);
```

```
                AbortMultipartUploadRequest abortMPURequest = new
 AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId
                };
                await s3Client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
    }
}
```

使用用戶端加密保護資料

用戶端加密是在本機加密的動作，以協助確保其在傳輸和靜態中的安全性。若要在將物件傳送到 Amazon S3 之前進行加密，請使用 Amazon S3 加密用戶端。當您的物件以這種方式加密時，您的物件不會暴露給任何第三方，包括 AWS. Amazon S3 得知您的物件已經加密；Amazon S3 不負責加密或解密物件。您可以同時使用 Amazon S3 加密用戶端和伺服器端加密來加密資料。將加密物件傳送到 Amazon S3 時，Amazon S3 只會偵測典型物件，無法將物件識別為已加密。

Amazon S3 加密用戶端可做為您與 Amazon S3 之間的媒介。執行個體化 Amazon S3 加密用戶端之後，物件會做為部分 Amazon S3 `PutObject` 和 `GetObject` 請求，自動加密和解密。您的物件皆使用不重複的資料金鑰來加密。即使您指定 KMS 金鑰做為包裝金鑰，Amazon S3 加密用戶端也不會使用儲存貯體金鑰或與其互動。

《Amazon S3 加密用戶端開發人員指南》著重於 Amazon S3 加密用戶端的 3.0 版及更新版本。如需詳細資訊，請參閱《Amazon S3 加密用戶端開發人員指南》中的什麼是 Amazon S3 用戶端加密。

如需舊版 Amazon S3 加密用戶端的詳細資訊，請參閱適用於您程式設計語言的 AWS SDK 開發人員指南。

- AWS SDK for Java
- AWS SDK for .NET
- AWS SDK for Go
- AWS SDK for PHP
- AWS SDK for Ruby
- AWS SDK for C++

網際網路流量隱私權

本主題描述 Amazon S3 如何保護從服務到其他位置的連線。

服務和內部部署用戶端與應用程式之間的流量。

下列連線可結合使用， AWS PrivateLink 以提供私人網路與私人網路之間的連線 AWS：

- AWS Site-to-Site VPN 連線。如需詳細資訊，請參閱什麼是 AWS Site-to-Site VPN？
- 一個 AWS Direct Connect 連接。如需詳細資訊，請參閱什麼是 AWS Direct Connect？

透過網路存取 Amazon S3 是透過 AWS 已發佈的 API 進行的。用戶端必須支援 Transport Layer Security (TLS) 1.2。我們建議使用 TLS 1.3。用戶端也必須支援具備完整轉寄密碼 (PFS) 的密碼套件，例如暫時性 Diffie-Hellman (DHE) 或橢圓曲線 Diffie-Hellman Ephemeral (ECDHE)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。此外，您必須使用存取金鑰 ID，以及與 IAM 委託人相關聯的私密存取金鑰來簽署請求，或者您可以使用 AWS Security Token Service (STS) 來產生臨時安全憑證來簽署請求。

同一地區 AWS 資源之間的流量

適用於 Amazon S3 的 Virtual Private Cloud (VPC) 端點是 VPC 中的邏輯實體，僅允許連線到 Amazon S3。VPC 會將請求路由至 Amazon S3，並將回應路由回 VPC。如需詳細資訊，請參閱《VPC 使用者指南》中的 VPC 端點。如需您可用來從 VPC 端點控制 S3 儲存貯體存取的儲存貯體政策範例，請參閱使用儲存貯體政策控制來自 VPC 端點的存取。

AWS PrivateLink 適用於 Amazon S3

使 AWS PrivateLink 用 Amazon S3，您可以在虛擬私有雲 (VPC) 中佈建界面 VPC 端點 (界面端點)。這些端點可直接從內部部署的應用程式透過 VPN 存取 AWS Direct Connect，或透 AWS 區域 過 VPC 對等互連的其他應用程式存取。

介面端點由一個或多個彈性網路介面 (ENI) 來表示，這些是在 VPC 的子網路中指派的私有 IP 地址。透過界面端點傳送到 Amazon S3 的請求會保留在 Amazon 網路。您也可以透過 AWS Direct Connect 或 AWS Virtual Private Network (AWS VPN) 從內部部署應用程式存取 VPC 中的介面端點。如需有關如何將 VPC 與內部部署網路連線的詳細資訊，請參閱AWS Direct Connect 《 使用者指南》和AWS Site-to-Site VPN 《 使用者指南》。

如需有關介面端點的一般資訊，請參閱《AWS PrivateLink指南》中的介面 VPC 端點 (AWS PrivateLink)。

主題

適用於 Amazon S3 的 VPC 端點類型

您可以使用兩種類型的 VPC 端點存取 Amazon S3：閘道端點和界面端點 (使用 AWS PrivateLink)。閘道端點是您在路由表中指定的閘道，可透過 AWS 網路從 VPC 存取 Amazon S3。界面端點使用私有 IP 地址，透過使用 VPC 對等或從 VPC 內部、內部部署或從另一個 VPC 中的 VPC 路 AWS 區域 由請求到 Amazon S3，從而擴展閘道端點的功能。 AWS Transit Gateway如需詳細資訊，請參閱什麼是 VPC 對等互連？和 Transit Gateway 與 VPC 對等互連。

介面端點與閘道端點相容。如果 VPC 中具有現有的閘道端點，則可以在同一 VPC 中使用兩種類型的端點。

適用於 Amazon S3 的閘道端點	適用於 Amazon S3 的介面端點
在這兩種情況下，您的網路流量都會保留在 AWS 網路上。	
使用 Amazon S3 公有 IP 地址	使用 VPC 中的私有 IP 地址來存取 Amazon S3
使用相同的 Amazon S3 DNS 名稱	需要端點特定 Amazon S3 DNS 名稱
不允許從內部部署存取	允許從內部部署存取
不允許從其他人存取 AWS 區域	使用 VPC 對等互連，允許從另一個 VPC 中的 VPC AWS 區域 進行存取 AWS Transit Gateway
不計費	計費

如需有關閘道端點的詳細資訊，請參閱《AWS PrivateLink 指南》中的閘道 VPC 端點。

Amazon S3 的 AWS PrivateLink 限制和限制

VPC 限制適用 AWS PrivateLink 於 Amazon S3。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的界面端點考量事項和 AWS PrivateLink 配額。此外，適用下列限制。

AWS PrivateLink 對於 Amazon S3 不支持以下內容：

- 聯邦資訊處理標準 (FIPS) 端點
- 網站端點
- 舊版全域端點
- S3 Dash 區域端點
- Amazon S3 雙堆疊端點
- 在不同的桶UploadPartCopy之間使用CopyObject或之間 AWS 區域
- Transport Layer Security (TLS) 1.1

建立一個 VPC 端點

如要建立 VPC 介面端點，請參閱《AWS PrivateLink 指南》中的建立 VPC 端點。

存取 Amazon S3 介面端點

在您建立介面端點時，Amazon S3 會產生兩種類型的端點特定 S3 DNS 名稱：區域和地區。

- 地區 DNS 名稱包括唯一的 VPC 端點識別碼、服務識別碼 AWS 區域、和名稱`vpce.amazonaws.com`中的。例如，針對 VPC 端點 ID*`vpce-1a2b3c4d`*，產生的 DNS 名稱可能類似於 *`vpce-1a2b3c4d-5e6f`*`.s3.us-east-1.vpce.amazonaws.com`。
- 地區 DNS 名稱包含可用區域，例如 *`vpce-1a2b3c4d-5e6f`*`-us-east-1a.s3.us-east-1.vpce.amazonaws.com`。如果您的架構可隔離可用區域，則可以使用此選項。例如，您可以將其用於故障遏止或降低區域資料傳輸成本。

端點特定 S3 DNS 名稱可以從 S3 公有 DNS 網域解析。

私有 DNS

VPC 界面端點的私有 DNS 選項會簡化透過 VPC 端點的 S3 流量路由，並協助您利用應用程式可用成本最低的網路路徑。您可以使用私有 DNS 選項路由區域 S3 流量，而無需更新 S3 用戶端，以使用界面端點的端點特定 DNS 名稱或管理 DNS 基礎設施。啟用私有 DNS 名稱後，區域 S3 DNS 查詢會針對下列端點解析 AWS PrivateLink 為的私有 IP 位址：

- 區域儲存貯體端點 (例如，`s3.us-east-1.amazonaws.com`)
- 控制端點 (例如，`s3-control.us-east-1.amazonaws.com`)
- 存取點端點 (例如，`s3-accesspoint.us-east-1.amazonaws.com`)

若您的 VPC 中有閘道端點，您可以在界面端點，透過現有 S3 閘道端點和內部部署請求，自動路由 VPC 內請求。這種方法可讓您免費使用 VPC 內流量的閘道端點，針對 VPC 內流量來最佳化網路成本。您的內部部署應用程式可 AWS PrivateLink 以在輸入解析器端點的協助下使用。Amazon 會為您的 VPC 提供 DNS 伺服器，名為 Route 53 Resolver。傳入 Resolver 端點會將 DNS 查詢從內部部署網路轉送至 Route 53 Resolver。

> **Important**
>
> 若要在使用僅針對輸入端點啟用私有 DNS 時利用成本最低的網路路徑，VPC 中必須有閘道端點。選取僅針對輸入端點啟用私有 DNS 時，閘道端點的存在有助於確保 VPC 內流量皆是透過 AWS 私有網路路由。選取僅針對輸入端點啟用私有 DNS 選項時，必須維護此閘道端點。若要刪除閘道端點，必須先清除僅針對輸入端點啟用私有 DNS。
> 如果您想要將現有界面端點更新為僅針對輸入端點啟用私有 DNS，請先確認您的 VPC 具有 S3 閘道端點。如需有關閘道端點和管理私有 DNS 名稱的詳細資訊，請參閱《AWS PrivateLink 指南》中的閘道 VPC 端點和管理 DNS 名稱。

只有支援閘道端點的服務才能使用僅針對輸入端點啟用私有 DNS 選項。

如需詳細了解使用僅針對輸入端點啟用私有 DNS建立的 VPC 端點，請參閱《AWS PrivateLink 指南》中的建立界面端點。

使用 VPC 主控台

在主控台中，您有兩個選項：啟用 DNS 名稱和僅針對輸入端點啟用私有 DNS。啟用 DNS 名稱是支援的選項 AWS PrivateLink。透過使用啟用 DNS 名稱選項，您可以使用 Amazon 與 Amazon S3 的私有

連線，同時向預設公有端點 DNS 名稱傳送請求。啟用此選項後，客戶可以利用其應用程式可用的最低成本網路路徑。

在 Amazon S3 的現有或新 VPC 界面端點上啟用私有 DNS 名稱時，根據預設，系統會選取僅針對輸入端點啟用私有 DNS 選項。如果選取此選項，針對內部部署流量，您的應用程式只會使用界面端點。此 VPC 內流量會自動使用成本較低的閘道端點。或者，您可以清除僅針對傳入端點啟用私有 DNS，以透過界面端點路由所有 S3 請求。

使用 AWS CLI

如果您不指定 `PrivateDnsOnlyForInboundResolverEndpoint` 的值，則會預設為 `true`。但是，在 VPC 套用您的設定之前，會先執行檢查以確定 VPC 中存在閘道端點。如果 VPC 中存在閘道端點，則表示呼叫成功。若無，您看到以下錯誤訊息：

若要設定 PrivateDnsOnlyForInboundResolverEndpoint 為 true，VPC *vpce_id* 必須具有該服務的閘道端點。

針對新的 VPC 界面端點

使用 `private-dns-enabled` 和 `dns-options` 屬性透過命令列啟用私有 DNS。`dns-options` 屬性中的 `PrivateDnsOnlyForInboundResolverEndpoint` 選項必須設定為 `true`。以您自己的資訊取代 *user input placeholders*。

```
aws ec2 create-vpc-endpoint \
--region us-east-1 \
--service-name s3-service-name \
--vpc-id client-vpc-id \
--subnet-ids client-subnet-id \
--vpc-endpoint-type Interface  \
--private-dns-enabled  \
--ip-address-type ip-address-type \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=true \
--security-group-ids client-sg-id
```

針對現有 VPC 端點

如果您想對現有 VPC 端點使用私有 DNS，請使用下列範例命令，並以您自己的資訊取代 *user input placeholders*。

```
aws ec2 modify-vpc-endpoint \
--region us-east-1 \
--vpc-endpoint-id client-vpc-id \
--private-dns-enabled \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=false
```

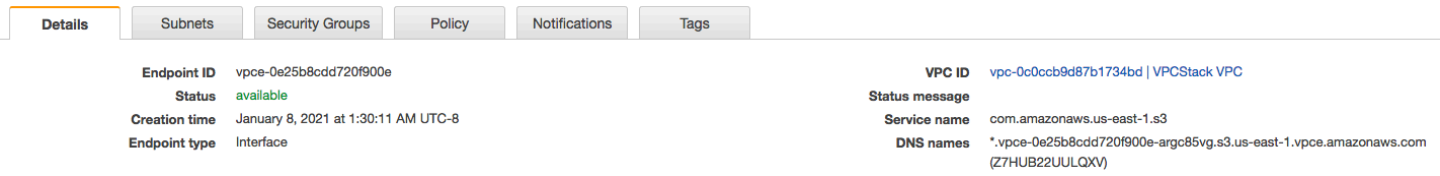
如果您想要更新現有 VPC 端點，以僅針對傳入 Resolver 啟用私有 DNS，請使用下列範例，並以您自己的值取代範例值。

```
aws ec2 modify-vpc-endpoint \
--region us-east-1 \
--vpc-endpoint-id client-vpc-id \
--private-dns-enabled \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=true
```

從 S3 界面端點存取儲存貯體、存取點和 Amazon S3 控制 API 操作

您可以使用 AWS CLI 或 AWS 開發套件透過 S3 界面端點存取儲存貯體、S3 存取點和 Amazon S3 控制 API 操作。

下圖顯示了 VPC 主控台詳細資訊標籤，您可以在其中找到 VPC 端點的 DNS 名稱。在此範例中，VPC 端點 ID (vpce-id) 為 `vpce-0e25b8cdd720f900e`，DNS 名稱為 `*.vpce-0e25b8cdd720f900e-argc85vg.s3.us-east-1.vpce.amazonaws.com`。



使用 DNS 名稱存取資源時，請以適當的值取代 *。取代 * 的適當值如下：

- `bucket`
- `accesspoint`

- `control`

例如，若要存取儲存貯體，請使用類似以下的 DNS 名稱：

`bucket.vpce-0e25b8cdd720f900e-argc85vg.s3.us-east-1.vpce.amazonaws.com`

如需相關範例，了解如何使用 DNS 名稱存取儲存貯體、存取點和 Amazon S3 控制 API 操作，請參閱以下 AWS CLI 例子 和 AWS SDK 範例 章節。

如需如何檢視端點特定 DNS 名稱的詳細資訊，請參閱《VPC 使用者指南》中的檢視端點服務私有 DNS 名稱組態。

AWS CLI 例子

若要透過 AWS CLI 命令中的 S3 界面端點存取 S3 儲存貯體、S3 存取點或 Amazon S3 控制 API 操作，請使用`--region`和`--endpoint-url`參數。

範例：使用端點 URL 列出儲存貯體中的物件

在下列範例中，將儲存貯體名稱 *my-bucket*、區域 *us-east-1* 和 VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 的 DNS 名稱取代為您自己的資訊。

```
aws s3 ls s3://my-bucket/ --region us-east-1 --endpoint-url
 https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com
```

範例：使用端點 URL 從存取點列出物件

- 方法 1 — 搭配使用存取點的 Amazon Resource Name (ARN) 與存取點端點

  將 ARN *us-east-1:123456789012:accesspoint/accesspointexamplename*、區域 *us-east-1* 和 VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 取代為您的資訊。

  ```
  aws s3api list-objects-v2 --bucket arn:aws:s3:us-east-1:123456789012:accesspoint/
  accesspointexamplename --region us-east-1 --endpoint-url
   https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com
  ```

  如果您無法成功執行命令，請更新 AWS CLI 至最新版本，然後再試一次。如需更新相關指示，請參閱《AWS Command Line Interface 使用者指南》中的安裝或更新 AWS CLI的最新版本。

- 方法 2 — 將存取點的別名與區域儲存貯體端點搭配使用

  在下列範例中，將存取點別名 *accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias*、區域 *us-east-1* 和 VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 取代為您的資訊。

  ```
  aws s3api list-objects-v2 --
  bucket accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias
   --region us-east-1 --endpoint-url https://bucket.vpce-1a2b3c4d-5e6f.s3.us-
  east-1.vpce.amazonaws.com
  ```

- 方法 3 — 將存取點的別名與存取點端點搭配使用

  首先，若要使用包含在主機名稱中的儲存貯體來建構 S3 端點，請將 `aws s3api` 的地址樣式設定為 `virtual` 以供使用。如需 `AWS configure` 相關資訊，請參閱《AWS Command Line Interface 使用者指南》中的組態和憑證檔案設定。

  ```
  aws configure set default.s3.addressing_style virtual
  ```

  接著，在下列範例中，將存取點別名 *accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias*、區域 *us-east-1* 和 VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 取代為您的資訊。如需存取點別名的詳細資訊，請參閱針對您的 S3 儲存貯體存取點使用儲存貯體樣式別名。

  ```
  aws s3api list-objects-v2 --
  bucket accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias --
  region us-east-1 --endpoint-url https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
  east-1.vpce.amazonaws.com
  ```

範例：使用端點 URL 列出具有 S3 控制 API 操作的任務

在下列範例中，將區域 *us-east-1*、VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 和 帳戶 ID *12345678* 取代為您的資訊。

```
aws s3control --region us-east-1 --endpoint-url
 https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com list-jobs --
account-id 12345678
```

AWS SDK 範例

若要在使用 AWS 開發套件時透過 S3 介面端點存取 S3 儲存貯體、S3 存取點或 Amazon S3 控制 API 操作，請將您的開發套件更新為最新版本。接著，設定您的用戶端使用端點 URL，以透過 S3 界面端點來存取儲存貯體、存取點或 Amazon S3 控制 API 操作。

SDK for Python (Boto3)

範例：使用端點 URL 存取 S3 儲存貯體

在下列範例中，將區域 *us-east-1* 和 VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 取代為您的資訊。

```
s3_client = session.client(
service_name='s3',
region_name='us-east-1',
endpoint_url='https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com'
)
```

範例：使用端點 URL 存取 S3 存取點

在下列範例中，將區域 *us-east-1* 和 VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 取代為您的資訊。

```
ap_client = session.client(
service_name='s3',
region_name='us-east-1',
endpoint_url='https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com'
)
```

範例：使用端點 URL 存取 Amazon S3 控制 API

在下列範例中，將區域 *us-east-1* 和 VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 取代為您的資訊。

```
control_client = session.client(
service_name='s3control',
region_name='us-east-1',
endpoint_url='https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com'
)
```

SDK for Java 1.x

範例：使用端點 URL 存取 S3 儲存貯體

在下列範例中，將 VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 取代為您的資訊。

```
// bucket client
final AmazonS3 s3 = AmazonS3ClientBuilder.standard().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
List<Bucket> buckets = s3.listBuckets();
```

範例：使用端點 URL 存取 S3 存取點

在下列範例中，將 VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 和 ARN *us-east-1:123456789012:accesspoint/prod* 取代為您的資訊。

```
// accesspoint client
final AmazonS3 s3accesspoint =
 AmazonS3ClientBuilder.standard().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
ObjectListing objects = s3accesspoint.listObjects("arn:aws:s3:us-
east-1:123456789012:accesspoint/prod");
```

範例：使用端點 URL 存取 Amazon S3 控制 API 操作

在下列範例中，將 VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 取代為您的資訊。

```
// control client
final AWSS3Control s3control =
 AWSS3ControlClient.builder().withEndpointConfiguration(
```

```
        new AwsClientBuilder.EndpointConfiguration(
                "https://control.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
final ListJobsResult jobs = s3control.listJobs(new ListJobsRequest());
```

SDK for Java 2.x

範例：使用端點 URL 存取 S3 儲存貯體

在下列範例中，將 VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 和區域 *Region.US_EAST_1* 取代為您的資訊。

```
// bucket client
Region region = Region.US_EAST_1;
s3Client = S3Client.builder().region(region)

 .endpointOverride(URI.create("https://bucket.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com"))
                   .build()
```

範例：使用端點 URL 存取 S3 存取點

在下列範例中，將 VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 和區域 *Region.US_EAST_1* 取代為您的資訊。

```
// accesspoint client
Region region = Region.US_EAST_1;
s3Client = S3Client.builder().region(region)

 .endpointOverride(URI.create("https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com"))
                   .build()
```

範例：使用端點 URL 存取 Amazon S3 控制 API

在下列範例中，將 VPC 端點 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 和區域 *Region.US_EAST_1* 取代為您的資訊。

```
// control client
```

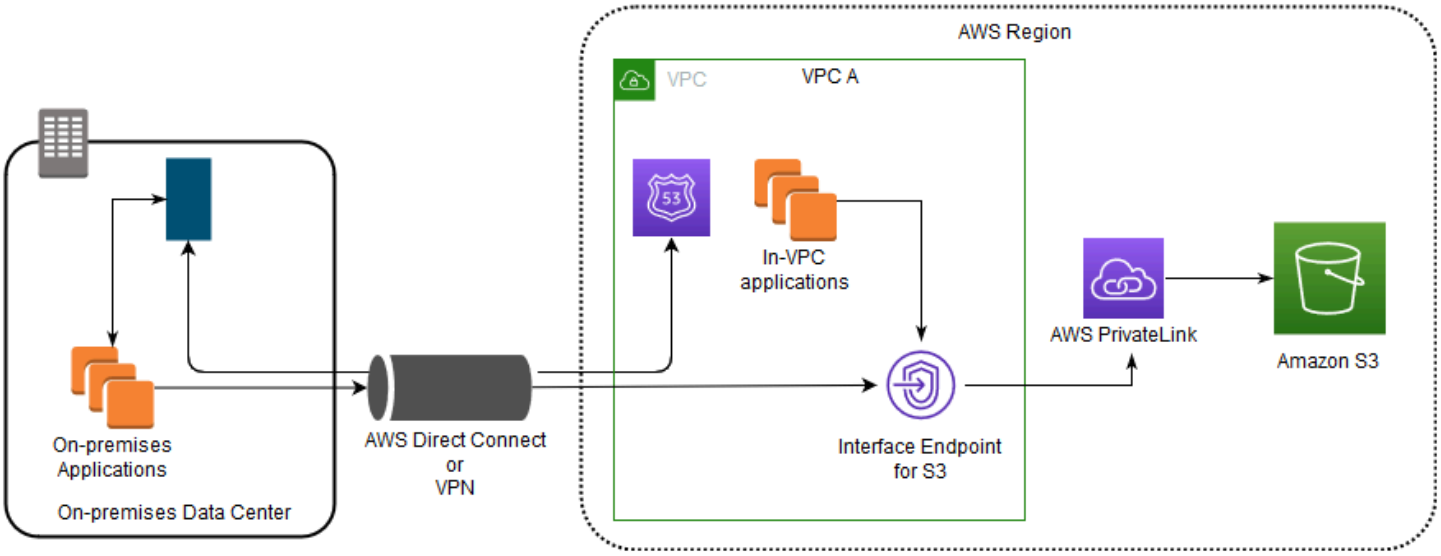
```
Region region = Region.US_EAST_1;
s3ControlClient = S3ControlClient.builder().region(region)

 .endpointOverride(URI.create("https://control.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com"))
                                    .build()
```

更新內部部署 DNS 組態

使用端點特定 DNS 名稱來存取 Amazon S3 的介面端點時，您不必更新內部部署 DNS 解析器。您可以使用公有 Amazon S3 DNS 網域中介面端點的私有 IP 地址，來解析端點特定的 DNS 名稱。

使用介面端點來存取 Amazon S3，而不需要 VPC 中的閘道端點或網際網路閘道

VPC 中的介面端點可以透過 Amazon 網路，將 VPC 內應用程式和內部部署應用程式路由至 Amazon S3，如下圖所示。

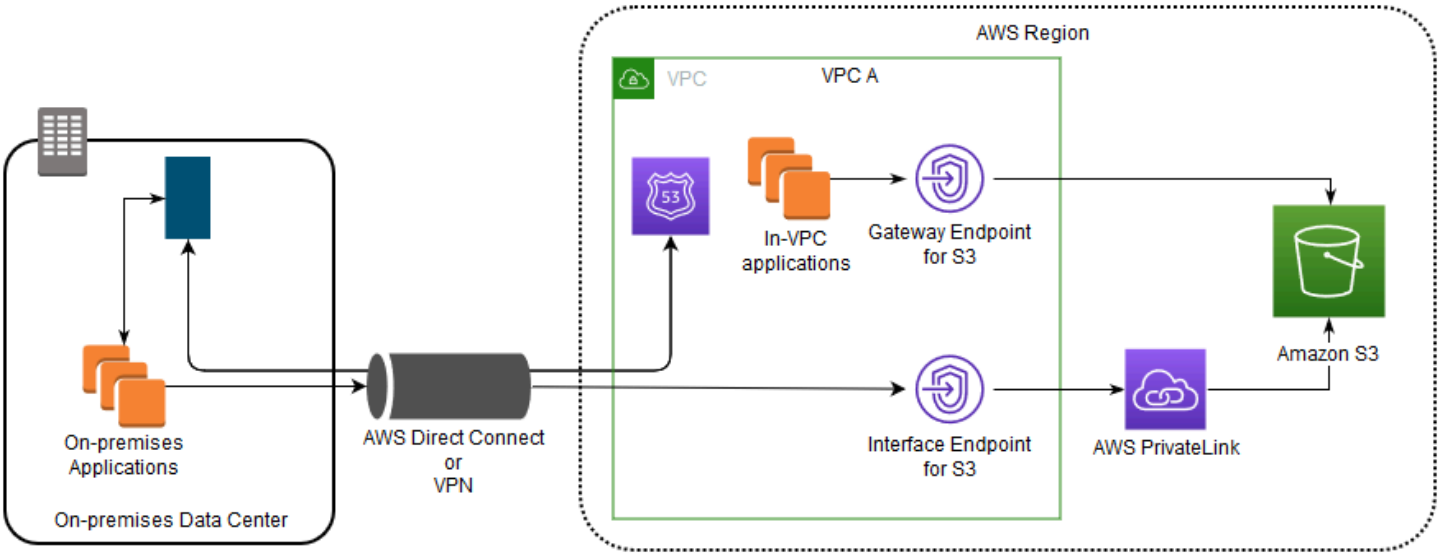


此圖展示了以下要點：

- 您的內部部署網路使用 AWS Direct Connect 或連線 AWS VPN 至 VPC A。
- 您的內部部署和 VPC A 中的應用程式使用端點特定的 DNS 名稱，透過 S3 介面端點存取 Amazon S3。
- 內部部署應用程式透過 AWS Direct Connect (或 AWS VPN) 將資料傳送到 VPC 中的介面端點。 AWS PrivateLink 透過 AWS 網路將資料從介面端點移至 Amazon S3。
- 虛擬私人雲端應用程式也會將流量傳送至介面端點。 AWS PrivateLink 透過 AWS 網路將資料從介面端點移至 Amazon S3。

在同一 VPC 中使用閘道端點和介面端點來存取 Amazon S3

您可以建立介面端點，並將現有的閘道端點保留在同一 VPC 中，如下圖所示。使用這個方式，您可以允許 VPC 內應用程式繼續透過閘道端點來存取 Amazon S3，而無需支付費用。然後，只有您的內部部署應用程式才會使用控制 API 端點來存取 Amazon S3。若要以這種方式存取 Amazon S3，您必須更新內部部署應用程式，以使用 Amazon S3 端點特定 DNS 名稱。



此圖展示了以下要點：

- 內部部署應用程式使用端點特定的 DNS 名稱，透過 AWS Direct Connect (或) 將資料傳送至 VPC 內的介面端點。 AWS VPN AWS PrivateLink 透過 AWS 網路將資料從介面端點移至 Amazon S3。
- VPC 擬私人雲端應用程式使用預設區域 Amazon S3 名稱，將資料傳送到透過網路連接到 Amazon S3 的閘道端點。 AWS

如需有關閘道端點的詳細資訊，請參閱《VPC 使用者指南》中的閘道 VPC 端點。

為 Amazon S3 建立 VPC 端點政策

您可以將端點政策連接至控制 Amazon S3 存取權的 VPC 端點。此政策會指定下列資訊：

- 可以執行動作的 AWS Identity and Access Management (IAM) 主體
- 可執行的動作
- 可在其中執行動作的資源

您還可以利用 Amazon S3 儲存貯體政策，使用儲存貯體政策中的 `aws:sourceVpce` 條件，來限制特定 VPC 端點對特定儲存貯體的存取。下列範例顯示了限制儲存貯體或端點存取權的政策。

主題

範例：限制從 VPC 端點對特定儲存貯體的存取

您可以建立端點政策，以限制只存取特定 Amazon S3 儲存貯體。如果您的 VPC AWS 服務 中有其他使用值區的策略，則此類型的策略非常有用。下列儲存貯體政策限制只存取 *example-s3-bucket1*。若要使用此端點政策，請將 *example-s3-bucket1* 取代為您的儲存貯體名稱。

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909151",
  "Statement": [
    { "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::example-s3-bucket1",
                   "arn:aws:s3:::example-s3-bucket1/*"]
    }
  ]
}
```

範例：限制從 VPC 端點對特定帳戶中儲存貯體的存取

您可以建立端點政策，限制只存取特定 S3 儲存貯體中的 S3 儲存貯體 AWS 帳戶。若要防止 VPC 內的用戶端存取您未擁有的儲存貯體，請在端點政策中使用下列陳述式。下列範例陳述式會建立政策，限制對單一 AWS 帳戶 ID *111122223333* 所擁有資源的存取。

```
{
  "Statement": [
    {
      "Sid": "Access-to-bucket-in-specific-account-only",
      "Principal": "*",
      "Action": [
```

```
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Deny",
      "Resource": "arn:aws:s3:::*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

> **Note**
>
> 若要指定要存取之資源的 AWS 帳戶 ID，您可以在 IAM 政策中使用`aws:ResourceAccount`或`s3:ResourceAccount`金鑰。但是，請注意，有些人 AWS 服務 依賴對 AWS 託管存儲桶的訪問。因此，在 IAM 政策使用 `aws:ResourceAccount` 或者 `s3:ResourceAccount` 金鑰也可能影響對這些資源的存取。

範例：限制對 S3 儲存貯體政策中特定 VPC 端點的存取

範例：限制對 S3 儲存貯體政策中特定 VPC 端點的存取

下列 Amazon S3 儲存貯體政策允許僅從 VPC 端點 *vpce-1a2b3c4d* 存取特定儲存貯體 *example-s3-bucket2*。如果未使用指定的端點，政策會拒絕所有對儲存貯體的存取。`aws:sourceVpce` 條件會指定端點，且不需要 VPC 端點資源的 Amazon Resource Name (ARN)，只需要端點 ID。若要使用此儲存貯體政策，請將 *example-s3-bucket2* 和 *vpce-1a2b3c4d* 取代為您的儲存貯體名稱與端點。

> **Important**
>
> - 套用下列 Amazon S3 儲存貯體政策以將存取限制為特定 VPC 端點時，您可能會意外封鎖對儲存貯體的存取。來自您 VPC 端點，旨在特別限制儲存貯體存取連線的儲存貯體政策，可能會封鎖所有對儲存貯體的連線。如需有關如何修復此問題的資訊，請參閱我的儲存貯體政策有錯誤的 VPC 或 VPC 端點 ID。我該如何修復政策，讓我可以存取儲存貯體？(位於 AWS Support 知識中心)。

- 使用下列範例政策之前，請以適合您使用案例的適當值取代 VPC 端點 ID。否則，您將無法存取儲存貯體。
- 此政策會停用對指定儲存貯體的主控台存取，因為主控台要求不是來自指定的 VPC 端點。

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909152",
  "Statement": [
    { "Sid": "Access-to-specific-VPCE-only",
      "Principal": "*",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::example-s3-bucket2",
                   "arn:aws:s3:::example-s3-bucket2/*"],
      "Condition": {"StringNotEquals": {"aws:sourceVpce": "vpce-1a2b3c4d"}}
    }
  ]
}
```

如需更多政策範例，請參閱《VPC 使用者指南》中的適用於 Amazon S3 的端點。

如需 VPC 連線的詳細資訊，請參閱 AWS 白皮書 Amazon Virtual Pri vate Cloud 連線選項中的網路到 VPC 連線選項。

存取管理

在中 AWS，資源是您可以使用的實體。在 Amazon Simple Storage Service (S3) 中，儲存貯體和物件是原始的 Amazon S3 資源。每個 S3 客戶可能都有包含物件的儲存貯體。隨著 S3 新增功能，也會新增其他資源，但並非每個客戶都會使用這些特定功能的資源。如需 Amazon S3 資源的詳細資訊，請參閱S3 資源。

根據預設，所有 Amazon S3 資源均為私有資源。根據預設，在 AWS 帳戶 該帳戶內建立資源 (資源擁有者) 的根使用者和 IAM 使用者 (擁有必要權限) 可以存取他們建立的資源。資源擁有者決定誰可以存取資源，以及允許其他人對資源執行的動作。S3 提供各種存取管理工具，您可以用來授與其他人存取您 S3 資源的權限。

以下各節提供 S3 資源的概觀、可用的 S3 存取管理工具，以及每個存取管理工具的最佳使用案例。這些章節中的清單旨在提供全面性，並包括所有 S3 資源、存取管理工具和常見的存取管理使用案例。同

時，這些部分被設計為可引導您找到所需技術詳細信息的目錄。如果您對以下某些主題有很好的了解，可以跳到適用於您的部分。

主題

S3 資源

原始的 Amazon S3 資源是儲存貯體及其包含的物件。隨著 S3 新增功能，也會新增資源。以下是 S3 資源及其各自功能的完整清單。

資源類型	Amazon S3 功能	描述
`bucket`	核心功能	儲存貯體是物件的容器。若要將物件存放在 S3 中，請建立儲存貯體，然後將一或多個物件上傳至儲存貯體。如需詳細資訊，請參閱 建立、設定和使用 Amazon S3 儲存貯體。
`object`		物件可以是檔案，也可以是描述該檔案的任何中繼資料。當物件位於值區中時，您可以開啟物件、下載並移動物件。如需詳細資訊，請參閱 在 Amazon S3 中上傳、下載和使用物件。

資源類型	Amazon S3 功能	描述
accesspoint	存取點	存取點是指連接至儲存貯體的網路端點，可用來執行 Amazon S3 物件操作，例如GetObject 和PutObject 。每個存取點都有不同的權限、網路控制以及自訂的存取點原則，可與附加至基礎值區的值區政策搭配使用。您可以將任何存取點設定為僅接受來自虛擬私有雲 (VPC) 的要求，或為每個存取點設定自訂封鎖公用存取設定。如需詳細資訊，請參閱 使用 Amazon S3 存取點管理資料存取。
objectlambdaaccesspoint		物件 Lambda 存取點是儲存貯體的存取點，該儲存貯體也與 Lambda 函數相關聯。使用物件 Lambda 存取點，您可以將自己的程式碼新增至 Amazon S3 GETLIST，以及HEAD請求在資料傳回至應用程式時修改和處理資料。如需詳細資訊，請參閱 建立 Object Lambda 存取點。
multiregionaccesspoint		多區域存取點提供了一個全球端點，應用程式可用來滿足來自多個區 AWS 域之 Amazon S3 儲存貯體的請求。您可以使用多區域存取點，進而使用與單一區域中使用的相同架構來建置多區域應用程式，然後在全球任何地方執行這些應用程式。對多區域存取點全球端點發出的應用程式請求不會透過擁擠的公用網際網路傳送請求，而是會自動透過 AWS 全球網路路由到最近的 Amazon S3 儲存貯體。如需詳細資訊，請參閱 Amazon S3 中的多區域存取點。
job	S3 批次操作	工作是 S3 Batch 操作功能的資源。您可以使用 S3 Batch 操作對您指定的 Amazon S3 物件清單執行大規模批次操作。Amazon S3 會追蹤批次操作任務的進度、傳送通知並存放所有動作的詳細完成報告，為您提供全受管、可稽核和無伺服器體驗。如需詳細資訊，請參閱 在 Amazon S3 物件上執行大規模批次操作。

資源類型	Amazon S3 功能	描述
storagelensconfiguration	S3 Storage Lens	S3 儲存鏡頭組態可跨帳戶收集整個組織的儲存指標和使用者資料。S3 Storage Lens 為管理員提供組織中數百個甚至數千個帳戶的物件儲存使用情況和活動的單一檢視，並提供詳細資訊，以便在多個彙總層級產生洞察。如需詳細資訊，請參閱 使用 Amazon S3 Storage Lens 評估儲存活動和使用量。
storagelensgroup		S3 儲存鏡頭群組使用以物件中繼資料為基礎的自訂篩選器來彙總指標。S3 Storage Lens 群組可協助您調查資料的特徵，例如按年齡分佈物件、最常見的檔案類型等。如需詳細資訊，請參閱 使用 Amazon S3 Storage Lens 群組。
accessgrantsinstance	S3 Access Grants	S3 存取授與執行個體是您建立之 S3 授權的容器。使用 S3 存取授權，您可以針對帳戶內的 IAM 身分、其他帳戶中的 IAM 身分 (跨帳戶)，以及 AWS IAM Identity Center 從公司目錄新增到的目錄身分建立授權給 Amazon S3 資料。如需 S3 存取授與的詳細資訊，請參閱使用 S3 Access Grants 管理存取。
accessgrantslocation		存取授與位置是儲存貯體、儲存貯體內的前綴，或您在 S3 Access Grants 執行個體中註冊的物件。您必須先在 S3 Access Asfer 執行個體中註冊位置，才能建立該位置的授權。然後，透過 S3 Access Grants，您可以授與存取權存取權給帳戶中 IAM 身分的儲存貯體、前綴或物件、其他帳戶中的 IAM 身分 (跨帳戶)，以及 AWS IAM Identity Center 從公司目錄新增至的目錄身分。如需 S3 存取授與的詳細資訊，請參閱 使用 S3 Access Grants 管理存取
accessgrant		存取授與是對您的 Amazon S3 資料的個別授權。使用 S3 存取授權，您可以針對帳戶內的 IAM 身分、其他帳戶中的 IAM 身分 (跨帳戶)，以及 AWS IAM Identity Center 從公司目錄新增到的目錄身分建立授權給 Amazon S3 資料。如需 S3 存取授與的詳細資訊，請參閱 使用 S3 Access Grants 管理存取

儲存貯體

Amazon S3 儲存貯體有兩種類型：一般用途儲存貯體和目錄儲存貯體。

- 一般用途儲存貯體是原始 S3 儲存貯體類型，建議用於大多數使用案例和存取模式。一般用途儲存貯體也允許跨所有儲存類別儲存的物件，但 S3 Express One Zone 除外。如需 S3 儲存類別的詳細資訊，請參閱使用 Amazon S3 儲存體方案。
- 目錄儲存貯體使用 S3 Express One Zone 儲存類別，如果您的應用程式對效能敏感且受益於 10 毫PUT秒和延遲，則建議使用此類別。GET如需詳細資訊，請參閱目錄值區、什麼是 S3 Express One Zone？及AWS Identity and Access Management 適用於 S3 快遞單一區域的 (IAM)。

分類 S3 資源

Amazon S3 提供分類和組織 S3 資源的功能。對資源進行分類不僅對組織資源很有用，還可以根據資源類別設定存取管理規則。特別是，前置詞和標記是兩個儲存組織功能，您可以在設定存取管理權限時使用這些功能。

> **Note**
>
> 下列資訊適用於一般用途的值區。目錄值區不支援標記，且具有前置字元限制。如需詳細資訊，請參閱 AWS Identity and Access Management 適用於 S3 快遞單一區域的 (IAM)。

- 前綴 — Amazon S3 中的前綴是物件金鑰名稱開頭的一串字元，用於組織存放在 S3 儲存貯體中的物件。您可以使用分隔符號字元，例如正斜線 (/)，指出物件索引鍵名稱中前置詞的結尾。例如，您的物件索引鍵名稱可能以`engineering/`首碼開頭，或以前置詞開頭的物件索引鍵名`marketing/campaigns/`稱。在前置詞結尾使用分隔符號 (例如正斜線字元) 可/模擬資料夾和檔案命名慣例。但是，在 S3 中，前綴是對象密鑰名稱的一部分。在一般用途 S3 儲存貯體中，沒有實際的資料夾階層。

  Amazon S3 支援使用物件的前置詞來組織和分組物件。您也可以透過物件的首碼來管理物件的存取權限。例如，您可以限制只存取名稱以特定前置詞開頭的物件。

  如需詳細資訊，請參閱 使用字首整理物件。S3 主控台使用資料夾的概念，在一般用途的儲存貯體中，這些概念基本上是附加在物件金鑰名稱之前的前置詞。如需詳細資訊，請參閱 在 Amazon S3 主控台中使用資料夾整理物件。
- 標籤 — 每個標籤都是您指派給資源的索引鍵值配對。例如，您可以使用標籤標記某些資源`topicCategory=engineering`。您可以使用標記來協助進行成本分配、分類和組織以及存取控制。時段標記僅用於成本分配。您可以標記物件、S3 儲存鏡頭、任務和 S3 存取授權，以進行組織

或用於存取控制。在 S3 存取授權中，您也可以使用標記來分配成本。使用資源的標籤來控制資源存取的範例，您只能共用具有特定標籤或標籤組合的物件。

如需詳細AWS 資訊，請參閱 IAM 使用者指南中的使用資源標籤控制資源的存取。

身分

在 Amazon S3 中，資源擁有者是建立資源的身分，例如儲存貯體或物件。根據預設，只有在具有所需權限的帳戶內建立資源和 IAM 身分的帳戶的根使用者才能存取 S3 資源。資源擁有者可以授予其他身分存取其 S3 資源的權限。

不擁有資源的身分識別可以要求存取該資源。對資源的請求是經過驗證或未經驗證的。已驗證的要求必須包含可驗證要求寄件者的簽章值，但未經驗證的要求不需要簽章。我們建議您僅將存取權授與已驗證的使用者。如需請求身分驗證的詳細資訊，請參閱「提出要求」。

> **Important**
>
> 我們建議您不要使用 AWS 帳戶 root 使用者認證來發出驗證的要求。而是建立 IAM 角色，然後授予該角色完整的存取。我們稱擁有此角色的使用者為管理員使用者。您可以使用指派給系統管理員角色的認證 (而非 AWS 帳戶 root 使用者認證) 與其互動 AWS 並執行工作，例如建立值區、建立使用者以及授與權限。如需詳細資訊，請參閱中的AWS 帳戶 根使用者登入資料和 IAM 使用者登入資料 AWS 一般參考，並參閱 IAM 使用者指南中的 IAM 中的安全性最佳實務。

在 Amazon S3 中存取資料的身分可以是下列其中一種：

AWS 帳戶 owner

建 AWS 帳戶 立資源的。例如，建立值區的帳戶。此帳號擁有資源。如需詳細資訊，請參閱AWS 帳戶根使用者。

AWS 帳戶 擁有者相同帳戶中的 IAM 身分

為需要 S3 存取權的新團隊成員設定帳戶時， AWS 帳戶 擁有者可以使用 AWS Identity and Access Management (IAM) 建立使用者、群組和角色。然後， AWS 帳戶 擁有者可以使用這些 IAM 身分共用資源。帳戶擁有者也可以指定授與 IAM 身分的許可，以允許或拒絕可在共用資源上執行的動作。

IAM 身分提供更多功能，包括要求使用者在存取共用資源之前輸入登入認證的能力。透過使用 IAM 身分，您可以實作 IAM 多因素身分驗證 (MFA) 形式，以支援強大的身分基礎。IAM 最佳做法是建立用於

存取管理的角色，而不是將權限授予每個個別使用者。您可以將個別使用者指定給適當的角色。如需更多詳細資訊，請參閱 IAM 中的安全最佳實務。

其他 AWS 帳戶擁有者及其 IAM 身分 (跨帳戶存取)

AWS 帳戶 擁有者也可以將資源存取權授予其他 AWS 帳戶擁有者或屬於其他 AWS 帳戶的 IAM 身分。

> **Note**
>
> 權限委派 — 如果 AWS 帳戶 擁有資源，它可以將這些權限授與另一個資源 AWS 帳戶。然後，該帳戶可以將這些權限或其子集委派給相同帳戶中的使用者。這稱為委派許可。但是，從另一個帳戶接收權限的帳戶無法將這些權限「跨帳戶」委託給另一個 AWS 帳戶帳戶。

匿名使用者 (公開存取)

AWS 帳戶 擁有者可以將資源設為公開。將資源設為公開技術上會與匿名使用者共用資源。自 2023 年 4 月起建立的值區預設會封鎖所有公開存取，除非您變更此設定。我們建議您將值區設定為封鎖公用存取權，並且只授予已驗證使用者的存取權。如需封鎖公開存取的詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

AWS 服務

資源擁有者可以授與另一個 Amazon S3 資源的 AWS 服務存取權。例如，您可以授與 AWS CloudTrail 服務`s3:PutObject`權限，將記錄檔寫入值區。如需詳細資訊，請參閱提供 AWS 服務存取權。

公司目錄識別

資源擁有者可以使用 S3 存取授權，將您公司目錄中的使用者或角色授與 S3 資源的存取權。如需新增公司目錄的詳細資訊 AWS IAM Identity Center，請參閱什麼是 IAM 身分中心？ 。

值區或資源擁有者

您用來建立值區和上傳物件的資源擁有這些資源。 AWS 帳戶 值區擁有者可以將跨帳戶權限授與另一個帳戶 AWS 帳戶 (或其他帳戶中的使用者)，以便上傳物件。

當值區擁有者允許其他帳戶將物件上傳至值區時，值區擁有者預設會擁有上傳至其值區的所有物件。不過，如果強制執行的值區擁有者和值區擁有者偏好的值區設定都已關閉，則上傳物件將擁有這些物件，且值區擁有者對其他帳戶所擁有的物件沒有權限，但下列情況例外： AWS 帳戶

- 儲存貯體擁有者支付帳單。儲存貯體擁有者可拒絕對任何物件之存取，或刪除儲存貯體中的任何物件，而無須考慮其擁有者為何。
- 值區擁有者可以封存任何物件或還原已封存物件，不論物件的擁有者為何。封存指的是指用於存放物件的儲存體方案。如需詳細資訊，請參閱 管理儲存生命週期。

存取管理工具

Amazon S3 提供了各種安全性功能和工具。以下是這些功能和工具的完整清單。您不需要所有這些存取管理工具，但您必須使用一或多個授與 Amazon S3 資源的存取權。正確應用這些工具可以幫助確保您的資源只有預期的用戶可以訪問。

最常用的存取管理工具是存取原則。存取政策可以是附加至資源的 AWS 資源型政策，例如值區的儲存貯體政策。存取政策也可以是附加至 (IAM) 身分識別 AWS Identity and Access Management (例如 IAM 使用者、群組或角色) 的身分識別型政策。撰寫存取政策以授 AWS 帳戶 與 IAM 使用者、群組和角色對資源執行作業的權限。例如，您可以將`PUT Object`權限授予另一個帳戶，以 AWS 帳戶 便其他帳戶可以將物件上傳到您的值區。

存取原則說明誰可以存取哪些項目。Amazon S3 收到請求時，必須評估所有存取政策以確定是授權還是拒絕該請求。如需 Amazon S3 如何評估這些政策的詳細資訊，請參閱 Amazon S3 如何授權要求。

以下是 Amazon S3 提供的存取管理工具。

儲存貯體政策

Amazon S3 儲存貯體政策是附加到特定儲存貯體的 JSON 格式 AWS Identity and Access Management (IAM) 資源型政策。使用值區政策授與值區及其中物件的其他身分 AWS 帳戶 或 IAM 身分權限。使用儲存貯體政策可以滿足許多 S3 存取管理使用案例。透過儲存貯體政策，您可以個人化儲存貯體存取權，確保只有您已核准的身分識別可以存取資源並在其中執行動作。如需詳細資訊，請參閱 Amazon S3 的存儲桶政策。

以下為儲存貯體政策的範例。您可以使用 JSON 檔案來表示儲存貯體政策。此範例政策授予存儲桶中所有物件的 IAM 角色讀取權限。它包含一個名為的陳述式`BucketLevelReadPermissions`，允許對名為的值區中的物件`s3:GetObject`執行動作 (讀取權限) `DOC-EXAMPLE-BUCKET1`。將 IAM 角色指定為`Principal`，此政策將存取權授予任何具有此角色的 IAM 使用者。若要使用此範例政策，請以您自己的資訊取代 *`user input placeholders`*。

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

```
    {
      "Sid":"BucketLevelReadPermissions",
      "Effect":"Allow",
      "Principal": {
     "AWS": "arn:aws:iam::123456789101:role/s3-role"
      },
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"]
    }]
  }
```

> **Note**
>
> 建立政策時，請避免在 `Principal` 元素中使用萬用字元 (*)，因為使用萬用字元可讓任何人存取您的 Amazon S3 資源。而是明確列出允許存取值區的使用者或群組，或列出使用原則中的條件子句必須符合的條件。此外，您不必為使用者或群組的動作包含萬用字元，而是在適用時授予他們特定權限。

身分型政策

以身分識別為基礎或 IAM 使用者政策是一種 AWS Identity and Access Management (IAM) 政策。以身分識別為基礎的政策是 JSON 格式的政策，附加至帳戶中的 IAM 使用者、群組或角色。 AWS 您可以使用以身分識別為基礎的政策，授予值區或物件的 IAM 身分存取權。您可以在帳戶中建立 IAM 使用者、群組和角色，並將存取政策附加到他們。然後，您可以授與 AWS 資源的存取權，包括 Amazon S3 資源。如需詳細資訊，請參閱 Amazon S3 的基於身份的政策。

以下是以身分識別為基礎的原則範例。範例政策允許關聯的 IAM 角色對儲存貯體及其中的物件執行六個不同的 Amazon S3 動作 (許可)。如果您將此政策附加到帳戶中的 IAM 角色，並將角色指派給某些 IAM 使用者，則具有此角色的使用者將能夠對策略中指定的資源 (值區) 執行這些動作。若要使用此範例政策，請以您自己的資訊取代 *user input placeholders*。

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Sid": "AssignARoleActions",
    "Effect": "Allow",
    "Action": [
   "s3:PutObject",
   "s3:GetObject",
```

```
    "s3:ListBucket",
    "s3:DeleteObject",
    "s3:GetBucketLocation"
     ],
     "Resource": [
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*",
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
     ]
     },
   {
     "Sid": "AssignARoleActions2",
     "Effect": "Allow",
     "Action": "s3:ListAllMyBuckets",
     "Resource": "*"
   }
   ]
}
```

S3 Access Grants

使用 S3 存取授權，針對公司身分目錄中的身分 AWS Identity and Access Management (例如Active Directory，IAM) 身分識別建立對 Amazon S3 資料的存取權授與。S3 存取授權可協助您大規模管理資料許可。此外，S3 存取授權會記錄最終使用者身分和用於存取 S3 資料的應用程式 AWS CloudTrail。這會提供詳細的稽核歷史記錄，直到最終使用者身分，以便存取 S3 儲存貯體中的資料。如需詳細資訊，請參閱 使用 S3 Access Grants 管理存取。

存取點

Amazon S3 存取點可簡化在 S3 上使用共用資料集的應用程式大規模管理資料存取。存取點是指連接至值區的名稱為網路端點。您可以使用存取點大規模執行 S3 物件操作，例如上傳和擷取物件。儲存貯體最多可連接 10,000 個存取點，而對於每個存取點，您可以強制執行不同的許可和網路控制，讓您詳細控制 S3 物件的存取權。S3 存取點可以與相同帳戶或其他信任帳戶中的儲存貯體相關聯。存取點政策是以資源為基礎的政策，會與基礎儲存貯體政策一起評估。如需詳細資訊，請參閱 使用 Amazon S3 存取點管理資料存取。

存取控制清單 (ACL)

ACL 是識別受權者和授予權限的授權清單。ACL 會將基本的讀取或寫入權限授與其他 AWS 帳戶權限。ACL 使用 Amazon S3 專屬的 XML 結構描述。ACL 是一種 AWS Identity and Access Management (IAM) 政策類型。物件 ACL 可用來管理物件的存取權，而值區 ACL 則用來管理值區的存取權。使用儲存貯體政策時，整個儲存貯體都有單一政策，但是會針對每個物件指定物件 ACL。建議

您將 ACL 保持關閉狀態，除非在特殊情況下，您必須個別控制每個物件的存取權限。如需使用 ACL 的詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

> **Warning**
>
> Amazon S3 中的大多數現代使用案例不需要使用 ACL。

下列為儲存貯體 ACL 的範例。ACL 中的授權會顯示具有完整控制權限的值區擁有者。

```
<?xml version="1.0" encoding="UTF-8"?>
 <AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
   <ID>Owner-Canonical-User-ID</ID>
   <DisplayName>owner-display-name</DisplayName>
  </Owner>
 <AccessControlList>
  <Grant>
   <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Canonical
 User">
    <ID>Owner-Canonical-User-ID</ID>
    <DisplayName>display-name</DisplayName>
   </Grantee>
   <Permission>FULL_CONTROL</Permission>
  </Grant>
 </AccessControlList>
</AccessControlPolicy>
```

物件所有權

若要管理物件的存取權限，您必須是物件的擁有者。您可以使用「物件擁有權」值區層級設定來控制上載至值區之物件的擁有權。此外，請使用「物件所有權」來開啟 ACL。依預設，[物件擁有權] 會設定為 [值區擁有者強制執行] 設定，且所有 ACL 都會關閉。關閉 ACL 時，值區擁有者會擁有值區中的所有物件，並專門管理資料的存取權。若要管理存取權，值區擁有者會使用政策或其他存取管理工具 (ACL 除外)。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

「物件擁有權」有三種設定，可用來控制上載至值區之物件的擁有權，以及開啟 ACL：

ACL 已關閉

- 強制執行值區擁有者 (預設值) — ACL 會關閉，值區擁有者會自動擁有並完全控制值區中的每個物件。ACL 不會影響 S3 儲存貯體中資料的許可。儲存貯體單獨使用政策來定義存取控制。

ACL 已開啟

- 儲存貯體擁有者偏好 – 儲存貯體擁有者擁有並完全控制其他帳戶使用 `bucket-owner-full-control` 標準 ACL 寫入儲存貯體的新物件。
- 物件寫入器 — 上傳物件的物件擁有物件、擁有物件的完全控制權，並且可以透過 ACL 授與其他使用者存取 AWS 帳戶 該物件。

其他最佳實務

請考慮使用下列值區設定和工具來協助保護傳輸中和靜態資料，這兩項設定對於維護資料的完整性和可存取性至關重要：

- 封鎖公用存取 — 不要關閉預設值區層級設定封鎖公用存取。此設定預設會封鎖公開存取您的資料。如需封鎖公開存取的詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。
- S3 版本控制 — 為了資料完整性，您可以實作 S3 版本控制儲存貯體設定，該設定會在您進行更新時對物件進行版本化，而不是覆寫它們。如有需要，您可以使用 S3 版本控制來保留、擷取和還原先前的版本。如需 S3 版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。
- S3 物件鎖定 — S3 物件鎖定是您可以實作的另一個設定，以達成資料完整性。此功能可以實作 write-once-read-many (WORM) 模型來儲存不變的物件。如需有關物件鎖定的詳細資訊，請參閱「使用 S3 物件鎖定」。
- 物件加密 — Amazon S3 提供多種物件加密選項，可保護傳輸中和靜態資料。伺服器端加密會先將物件加密，然後再將物件儲存在其資料中心的磁碟上，然後在下載物件時將其解密。如果您驗證您的請求，而且您具有存取權限，則存取加密或未加密物件的方式沒有任何差異。如需詳細資訊，請參閱使用伺服器端加密保護資料。S3 預設會加密新上傳的物件。如需詳細資訊，請參閱 對 Amazon S3 儲存貯體設定預設伺服器端加密行為。「用戶端加密」是在將資料傳送到 Amazon S3 之前加密資料的動作。如需詳細資訊，請參閱 使用用戶端加密保護資料。
- 簽名方法 — 簽名版本 4 是將身份驗證信息添加到由 HTTP 發送的 AWS 請求的過程。為了安全起見，大多數的請求都 AWS 必須使用訪問密鑰進行簽名，該訪問密鑰包括訪問密鑰 ID 和秘密訪問密鑰。這兩種金鑰通常稱為您的安全憑證。如需詳細資訊，請參閱身分驗證請求 (AWS Signature 第 4 版) 和 Signature 第 4 版簽章程序。

動作

如需 S3 許可和條件金鑰的完整清單，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

動作

Amazon S3 的 AWS Identity and Access Management (IAM) 動作是可在 S3 儲存貯體或物件上執行的可能動作。您可以將這些動作授與身分，以便它們可以對您的 S3 資源採取行動。S3 動作的範例是讀`s3:GetObject`取儲存貯體中的物件，`s3:PutObject`以及將物件寫入儲存貯體。

條件索引鍵

除了動作之外，IAM 條件金鑰僅限於只有在符合條件時才授予存取權。條件鍵是可選的。

> **Note**
>
> 在以資源為基礎的存取政策 (例如儲存貯體政策) 或以身分識別為基礎的政策中，您可以指定下列項目：
>
> - 原則陳述式`Action`元素中的動作或動作陣列。
> - 在策略聲明的`Effect`元素中，您可以指`Allow`定授與列出的動作，也可以指`Deny`定封鎖列出的動作。若要進一步維持最低權限的作法，存取原則`Effect`元素中的`Deny`敘述句應盡可能廣泛，而`Allow`敘述句應盡可能縮小。 `Deny`與`s3:*`動作配對的效果是另一個針對包含在原則條件陳述式中的識別實作選擇加入最佳作法的好方法。
> - 政策陳述式`Condition`元素中的條件索引鍵。

存取管理使用案例

Amazon S3 為資源擁有者提供各種工具來授予存取權。您使用的 S3 存取管理工具取決於您要共用的 S3 資源、要授予存取權的身分，以及您要允許或拒絕的動作。您可能想要使用一個或組合的 S3 存取管理工具來管理 S3 資源的存取。

在大多數情況下，您可以使用存取原則來管理權限。存取政策可以是以資源為基礎的政策，可以是附加到資源 (例如儲存貯體) 或其他 Amazon S3 資源 (S3 資源)。存取政策也可以是以身分識別為基礎的政策，並附加至帳戶中的 AWS Identity and Access Management (IAM) 使用者、群組或角色。您可能會發現值區政策更適合您的使用案例。如需詳細資訊，請參閱 Amazon S3 的存儲桶政策。或者，您可以使用 AWS Identity and Access Management (IAM) 在您的內部建立 IAM 使用者、群組和角色， AWS 帳戶 並透過身分型政策管理其對值區和物件的存取權限。如需詳細資訊，請參閱 Amazon S3 的基於身份的政策。

為了協助您瀏覽這些存取管理選項，以下是每個 S3 存取管理工具的常見 Amazon S3 客戶使用案例和建議。

AWS 帳戶 擁有者只想與同一帳戶內的使用者共用值區

所有存取管理工具都可以滿足此基本使用案例。針對此使用案例，我們建議使用下列存取管理工具：

- 值區政策 — 如果您想要授與存取一個值區或少量值區的存取權，或是儲存貯體的存取權限在不同值區之間類似，請使用值區政策。使用儲存貯體政策，您可以為每個值區管理一個政策。如需詳細資訊，請參閱 Amazon S3 的存儲桶政策。
- 以身分識別為基礎的政策 — 如果您有大量值區具有不同存取權限的值區，而且只能管理少數使用者角色，則可以針對使用者、群組或角色使用 IAM 政策。如果您要管理使用者對其他 AWS 資源的存取權限以及 Amazon S3 資源，IAM 政策也是不錯的選擇。如需詳細資訊，請參閱 範例 1：為其使用者授予儲存貯體許可的儲存貯體擁有者。
- S3 存取授權 — 您可以使用 S3 存取授權授與對 S3 儲存貯體、前置詞或物件的存取權。S3 存取授權可讓您大規模指定不同的物件層級許可；而儲存貯體政策的大小限制為 20 KB。如需詳細資訊，請參閱 開始使用 S3 Access Grants。
- 存取點 — 您可以使用存取點，這些存取點是連接至值區的名稱為網路端點。儲存貯體最多可連接 10,000 個存取點，而對於每個存取點，您可以強制執行不同的許可和網路控制，讓您詳細控制 S3 物件的存取權。如需詳細資訊，請參閱 使用 Amazon S3 存取點管理資料存取。

AWS 帳戶 擁有者想要與其他 AWS 帳戶 (跨帳戶) 的使用者共用值區或物件

若要將權限授予其他人 AWS 帳戶，您必須使用值區政策或下列其中一個建議的存取管理工具。您無法針對此使用案例使用以身分識別為基礎的存取原則。如需授予跨帳戶存取權的詳細資訊，請參閱如何提供對 Amazon S3 儲存貯體中物件的跨帳戶存取？

針對此使用案例，我們建議使用下列存取管理工具：

- 儲存貯體政策 — 使用儲存貯體政策，您可以為每個值區管理一個政策。如需詳細資訊，請參閱 Amazon S3 的存儲桶政策。
- S3 存取授權 — 您可以使用 S3 存取授權授與跨帳戶許可給 S3 儲存貯體、前置詞或物件。您可以使用 S3 存取授權來大規模指定不同的物件層級許可；而儲存貯體政策的大小限制為 20 KB。如需詳細資訊，請參閱 開始使用 S3 Access Grants。
- 存取點 — 您可以使用存取點，這些存取點是連接至值區的名稱為網路端點。儲存貯體最多可連接 10,000 個存取點，而對於每個存取點，您可以強制執行不同的許可和網路控制，讓您詳細控制 S3 物件的存取權。如需詳細資訊，請參閱 使用 Amazon S3 存取點管理資料存取。

AWS 帳戶 擁有者或儲存貯體擁有者必須在物件層級或首碼層級授與權限，而且這些權限因物件或前綴到前綴而異

舉例來說，在值區政策中，您可以授與值區內共用特定金鑰名稱前置詞或具有特定標籤之物件的存取權。您可以對以金鑰名稱前置詞開頭的物件授與讀取權限`logs/`。不過，如果您的存取權限因物件而異，則使用儲存貯體政策授予個別物件的權限可能不實際，特別是因為儲存貯體政策的大小限制為 20 KB。

針對此使用案例，我們建議使用下列存取管理工具：

- S3 存取授與 — 您可以使用 S3 存取授與來管理物件層級或首碼層級的許可。與儲存貯體政策不同，您可以使用 S3 存取授與大規模指定不同的物件層級許可。儲存貯體政策的大小限制為 20 KB。如需詳細資訊，請參閱 開始使用 S3 Access Grants。
- 存取點 — 您可以使用存取點來管理物件層級或首碼層級的權限。存取點是指連接至值區的名稱為網路端點。儲存貯體最多可連接 10,000 個存取點，而對於每個存取點，您可以強制執行不同的許可和網路控制，讓您詳細控制 S3 物件的存取權。如需詳細資訊，請參閱 使用 Amazon S3 存取點管理資料存取。
- ACL — 我們不建議使用存取控制清單 (ACL)，特別是因為每個物件的 ACL 限制為 100 個授權。不過，如果您選擇開啟 ACL，請在「值區設定」中，將「物件擁有權」設定為「值區擁有者偏好設定」，並啟用 ACL。使用此設定時，儲存貯體擁有者會自動擁有使用 `bucket-owner-full-control` 標準 ACL 寫入的新物件，而不是物件寫入者。然後，您可以使用物件 ACL (XML 格式的存取原則) 來授與其他使用者對物件的存取權。如需詳細資訊，請參閱 存取控制清單 (ACL) 概觀。

AWS 帳戶 擁有者或值區擁有者想要限制儲存貯體只能存取特定帳號 ID

針對此使用案例，我們建議使用下列存取管理工具：

- 儲存貯體政策 — 使用儲存貯體政策，您可以為每個值區管理一個政策。如需詳細資訊，請參閱 Amazon S3 的存儲桶政策。
- 存取點 — 存取點命名為連接至值區的網路端點。儲存貯體最多可連接 10,000 個存取點，而對於每個存取點，您可以強制執行不同的許可和網路控制，讓您詳細控制 S3 物件的存取權。如需詳細資訊，請參閱 使用 Amazon S3 存取點管理資料存取。

AWS 帳戶 擁有者或值區擁有者希望每個存取其資料的使用者或應用程式都需要不同的端點

針對此使用案例，我們建議使用下列存取管理工具：

- 存取點 — 存取點命名為連接至值區的網路端點。儲存貯體最多可連接 10,000 個存取點，而對於每個存取點，您可以強制執行不同的許可和網路控制，讓您詳細控制 S3 物件的存取權。每個存取點都會強制執行自訂的存取點政策，該政策可結合附加至基礎儲存貯體的儲存貯體政策運作。如需詳細資訊，請參閱 使用 Amazon S3 存取點管理資料存取。

AWS 帳戶 擁有者或儲存貯體擁有者必須管理來自 S3 Virtual Private Cloud (VPC) 端 (VPC) 端點的存取

Amazon S3 的 Virtual Private Cloud (VPC) 端 (VPC) 端點是 VPC 內的邏輯實體，僅允許連線至 S3。針對此使用案例，我們建議使用下列存取管理工具：

- VPC 設定中的值區 — 您可以使用儲存貯體政策來控制允許存取值區的人員，以及可存取的 VPC 端點。如需詳細資訊，請參閱 使用儲存貯體政策控制來自 VPC 端點的存取。
- 存取點 — 如果您選擇設定存取點，您可以使用存取點原則。您可以將任何存取點設定為僅接受來自 Virtual Private Cloud (VPC) 的請求，以限制只能透過私人網路存取 Amazon S3 資料。您也可以為每個存取點設定自訂封鎖公開存取設定。如需詳細資訊，請參閱 使用 Amazon S3 存取點管理資料存取。

AWS 帳戶 擁有者或儲存貯體擁有者必須公開使用靜態網站

使用 S3，您可以託管靜態網站，並允許任何人查看從 S3 儲存貯體託管的網站內容。

針對此使用案例，我們建議使用下列存取管理工具：

- Amazon CloudFront — 此解決方案可讓您向公眾託管 Amazon S3 靜態網站，同時繼續封鎖儲存貯體內容的所有公開存取權。如果您想要保持啟用所有四個 S3 區塊公開存取設定並託管 S3 靜態網站，可以使用 Amazon CloudFront 來源存取控制 (OAC)。Amazon CloudFront 提供設定安全靜態網站所需的功能。此外，不使用此解決方案的 Amazon S3 靜態網站只能支援 HTTP 端點。CloudFront 使用 Amazon S3 的耐用儲存，同時提供額外的安全標頭，例如 HTTPS。HTTPS 透過加密一般 HTTP 請求並防止常見的網路攻擊來增加安全性。

  如需詳細資訊，請參閱 Amazon 開 CloudFront 發人員指南中的安全靜態網站入門。
- 讓您的 Amazon S3 儲存貯體可公開存取 — 您可以將儲存貯體設定為用作公開存取的靜態網站。

> **Warning**
>
> 我們不建議使用此方法。相反，我們建議您使用 Amazon S3 靜態網站作為 Amazon 的一部分 CloudFront。如需詳細資訊，請參閱上一個選項，或參閱開始使用安全靜態網站。

要在沒有 Amazon 的情況下創建一個 Amazon S3 靜態網站 CloudFront，首先，您必須關閉所有阻止公共訪問設置。為靜態網站撰寫儲存貯體政策時，請確定您只允許 `s3:GetObject` 動作，而不允許 `ListObject` 或 `PutObject` 許可。這有助於確保使用者無法檢視值區中的所有物件或新增自己的內容。如需詳細資訊，請參閱 設定網站存取許可。

AWS 帳戶 擁有者或值區擁有者想要公開儲存貯體的內容

建立新的 Amazon S3 儲存貯體時，預設會啟用「封鎖公用存取」設定。如需封鎖公開存取的詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

我們不建議允許公開存取您的儲存貯體。但是，如果您必須針對特定使用案例執行此動作，建議您針對此使用案例使用下列存取管理工具：

- 停用封鎖公用存取設定 — 值區擁有者可允許值區未經驗證的要求。例如，當值區具有公用值區政策，或儲存貯體 ACL 授予公開存取權時，允許未驗證的 PUT Object 要求。所有未經身份驗證的請求都是由其他任意 AWS 用戶，甚至是未經身份驗證的匿名用戶提出。此使用者在 ACL 中是以特定的正式使用者 ID `65a011a29cdf8ec533ec3d1ccaae921c` 呈現。如果將物件上載至`WRITE`或`FULL_CONTROL`，則這會特別授與「所有使用者」群組或匿名使用者的存取權。如需公有儲存貯體政策和公開存取控制清單 (ACL) 的詳細資訊，請參閱「「公有」的意義」。

AWS 帳戶 擁有者或儲存貯體擁有者已超過存取原則大小限制

儲存貯體政策和以身分識別為基礎的政策都有 20 KB 的大小限制。如果您的存取權限需求很複雜，則可能會超過此大小限制。

針對此使用案例，我們建議使用下列存取管理工具：

- 存取點 — 如果這適用於您的使用案例，請使用存取點。透過存取點，每個儲存貯體都有多個已命名的網路端點，每個端點都有自己的存取點政策，可與基礎儲存貯體策略搭配使用。不過，存取點只能對物件作用，而不能對儲存貯體作用，而且不支援跨區域複寫。如需詳細資訊，請參閱 使用 Amazon S3 存取點管理資料存取。

- S3 存取授權 — 使用 S3 存取授權，該授權支援大量授予存取儲存貯體、前置詞或物件的授權。如需詳細資訊，請參閱 開始使用 S3 Access Grants。

AWS 帳戶 擁有者或管理員角色想要將值區、前置詞或物件存取權直接授與公司目錄中的使用者或群組

您可以將公司目錄新增至，而不是透過 AWS Identity and Access Management (IAM) 管理使用者、群組和角色 AWS IAM Identity Center。如需詳細資訊，請參閱什麼是 IAM 身分中心？ 。

將公司目錄新增至之後 AWS IAM Identity Center，建議您使用下列存取管理工具授與公司目錄身分存取 S3 資源：

- S3 存取授權 — 使用 S3 存取授與，支援將存取權授予公司目錄中的使用者或角色。如需詳細資訊，請參閱 開始使用 S3 Access Grants。

AWS 帳戶 擁有者或儲存貯體擁有者想要授予 AWS CloudFront 服務存取權，以便將 CloudFront 日誌寫入 S3 儲存貯體

針對此使用案例，我們建議使用下列存取管理工具：

- 儲存貯體 ACL — 儲存貯體 ACL 的唯一建議使用案例是將許可授予特定項目 AWS 服務，例如 Amazon CloudFront `awslogsdelivery` 帳戶。當您建立或更新發行版並開啟 CloudFront 記錄功能時，請 CloudFront 更新值區 ACL，以授與`awslogsdelivery`帳戶將記錄寫入值區的`FULL_CONTROL`權限。如需詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的設定標準記錄和存取日誌檔所需的權限。如果儲存記錄的儲存貯體使用 S3 物件擁有權強制執行的儲存貯體擁有者設定來關閉 ACL，則 CloudFront 無法將記錄寫入儲存貯體。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

身為值區擁有者，您希望保有其他使用者新增至值區的物件的完整控制權

您可以使用儲存貯體政策、存取點或 S3 存取權授予其他帳戶將物件上傳到儲存貯體的存取權。如果您已授予值區的跨帳戶存取權，則可以確保上傳到值區的任何物件仍然受到您的完全控制。

針對此使用案例，我們建議使用下列存取管理工具：

- 物件擁有權 — 將值區層級設定保留為預設值區擁有者強制執行的物件擁有權設定。

存取管理疑難排

下列資源可協助您疑難排解 S3 存取管理的任何問題：

針對拒絕存取 (403 禁止) 錯誤進行疑難排解

如果您遇到存取拒絕問題，請檢查帳戶層級和儲存貯體層級設定。此外，請檢查您用來授與存取權的存取管理功能，以確定原則、設定或組態正確無誤。如需 Amazon S3 中導致拒絕存取 (403 禁止) 錯誤之常見原因的詳細資訊，請參閱 針對 Amazon S3 中的拒絕存取 (403 禁止) 錯誤進行疑難排解。

IAM Access Analyzer for S3

如果您不想公開任何資源，或者想要限制對資源的公開存取，可以使用適用於 S3 的 IAM 存取分析器。在 Amazon S3 主控台上，使用適用於 S3 的 IAM 存取分析器來檢閱所有具有儲存貯體存取控制清單 (ACL)、儲存貯體政策或授予公用或共用存取權的存取點政策的儲存貯體。適用於 S3 的 IAM Access Analyzer 會向您發出設定為允許存取網際網路或其他 AWS 帳戶任何人 (包括組織 AWS 帳戶外部) 的儲存貯體提醒您。針對每個公開或共用儲存貯體，您會收到報告公開或共用存取來源和層級的發現項目。

在 S3 適用的 IAM 存取分析器中，您可以透過單一動作封鎖儲存貯體的所有公開存取。建議您封鎖值區的所有公開存取權，除非您需要公開存取權才能支援特定使用案例。在封鎖所有公開存取之前，請確定您的應用程式在沒有公開存取權的情況下繼續正常運作。如需詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

您也可以檢閱儲存貯體層級的權限設定，以設定詳細的存取層級。對於需要公開或共用存取的特定和經驗證使用案例，您可以將對儲存貯體的發現項目存檔，以確認並記錄您要讓儲存貯體保持公開或共用。您可以隨時再次瀏覽和修改這些儲存貯體組態。您也可以將發現項目下載為 CSV 報告，供稽核之用。

您無需額外付費，即可在 Amazon S3 主控台上使用 IAM Access Analyzer for S3。IAM Access Analyzer for S3 採用 AWS Identity and Access Management (IAM) IAM Access Analyzer 技術。若要在 Amazon S3 主控台上使用適用於 S3 的 IAM 存取分析器，您必須造訪 IAM 主控台，並在 IAM 存取分析器中為每個個別區域建立帳戶層級分析器。

如需 IAM Access Analyzer for S3 的詳細資訊，請參閱 使用 IAM Access Analyzer for S3 檢閱儲存貯體存取權。

日誌記錄和監控

監控是維護 Amazon S3 解決方案的可靠性、可用性和效能的重要組成部分，因此您可以更輕鬆地對存取失敗進行偵錯。記錄可讓您深入瞭解使用者收到的任何錯誤，以及發出要求的時間和內容。 AWS 提供數種監控 Amazon S3 資源的工具，例如：

- AWS CloudTrail
- Amazon S3 存取日誌

- AWS Trusted Advisor
- Amazon CloudWatch

如需詳細資訊，請參閱 在 Amazon S3 中記錄和監控。

適用於 Amazon S3 的 Identity and Access Management

AWS Identity and Access Management (IAM) 可協助管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有許可) 來使用 Amazon S3 資源。IAM 是您可以使用的 AWS 服務 ，無需額外付費。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱 什麼是 S3 Express One Zone？ 和 目錄值區。

主題

- 物件
- 使用身分驗證
- 使用政策管理存取權
- Amazon S3 如何與 IAM 配合使用
- Amazon S3 中的政策和許可
- Amazon S3 的存儲桶政策
- Amazon S3 的基於身份的政策
- 使用政策管理 Amazon S3 資源存取權的逐步解說
- Amazon S3 如何授權要求
- AWS Amazon S3 的受管政策
- 讓 Amazon S3 Storage Lens 使用服務連結角色
- 疑難排解 Amazon S3 身分識別和存取

物件

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，具體取決於您在 Amazon S3 中執行的工作。

服務使用者 — 如果您使用 Amazon S3 服務執行工作，則管理員會為您提供所需的登入資料和許可。當您使用更多 Amazon S3 功能完成工作時，您可能需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法在 Amazon S3 中存取某個功能，請參閱疑難排解 Amazon S3 身分識別和存取。

服務管理員 — 如果您負責公司的 Amazon S3 資源，您可能擁有 Amazon S3 的完整存取權。判斷服務使用者應存取哪些 Amazon S3 功能和資源是您的任務。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何將 IAM 與 Amazon S3 搭配使用，請參閱Amazon S3 如何與 IAM 配合使用。

IAM 管理員 — 如果您是 IAM 管理員，您可能想要了解如何撰寫政策以管理 Amazon S3 存取權的詳細資訊。若要檢視可在 IAM 中使用的 Amazon S3 身分型政策範例，請參閱。Amazon S3 的基於身份的政策

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根

使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時登入資料進行存取 AWS 服務 。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。如需了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 和 VPC 是支援 ACL 的服務範例。 AWS WAF如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 IAM 實體許可界限。
- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需 Organizations 和 SCP 的詳細資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

Amazon S3 如何與 IAM 配合使用

在您使用 IAM 管理 Amazon S3 的存取權限之前，請先了解哪些 IAM 功能可用於 Amazon S3。

您可以搭配 Amazon S3 使用的 IAM 功能

IAM 功能	支援 Amazon S3
身分型政策	是
資源型政策	是
政策動作	是
政策資源	是
政策條件索引鍵 (服務特定)	是
ACL	是
ABAC(政策中的標籤)	部分
臨時憑證	是
轉送存取工作階段 (FAS)	是
服務角色	是
服務連結角色	部分

若要深入瞭解 Amazon S3 和其他 AWS 服務如何與大多數 IAM 功能搭配運作，請參閱 IAM 使用者指南中的搭配 IAM 使用的AWS 服務。

Amazon S3 的基於身份識別的政策

支援身分型政策	是

身分型政策是可以連接到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

Amazon S3 的基於身分識別的政策範例

若要檢視 Amazon S3 身分識別型政策的範例，請參閱。Amazon S3 的基於身份的政策

Amazon S3 內的資源型政策

支援以資源基礎的政策	是

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

如需啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源位於不同時 AWS 帳戶，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。

Amazon S3 服務支援儲存貯體政策、存取點政策和存取授權：

- 儲存貯體政策是附加至 Amazon S3 儲存貯體的資源型政策。值區政策定義了哪些主體可以對值區執行動作。
- 存取點政策是以資源為基礎的政策，會與基礎儲存貯體政策一起評估。
- 存取授權是一種簡化的模型，可透過前綴、儲存貯體或物件定義 Amazon S3 中資料的存取許可。如需 S3 存取授與的相關資訊，請參閱使用 S3 Access Grants 管理存取。

值區政策的主體

`Principal`元素會指定允許或拒絕存取資源的使用者、帳戶、服務或其他實體。以下為指定 `Principal` 的範例。如需詳細資訊，請參閱《IAM 使用者指南》中的委託人。

授予權限 AWS 帳戶

若要授與權限 AWS 帳戶，請使用下列格式識別帳戶。

```
"AWS":"account-ARN"
```

範例如下。

```
"Principal":{"AWS":"arn:aws:iam::AccountIDWithoutHyphens:root"}
```

```
"Principal":{"AWS":
["arn:aws:iam::AccountID1WithoutHyphens:root","arn:aws:iam::AccountID2WithoutHyphens:root"]}
```

將許可授予 IAM 使用者

若要將許可授予您帳戶內的 IAM 使用者，您必須提供 "AWS":"*user-ARN*" 名稱/值對。

```
"Principal":{"AWS":"arn:aws:iam::account-number-without-hyphens:user/username"}
```

如需提供 step-by-step 指示的詳細範例，請參閱範例 1：為其使用者授予儲存貯體許可的儲存貯體擁有者和範例 3：授予對其未擁有之物件的許可的儲存貯體擁有者。

> **Note**
>
> 若您在更新儲存貯體政策後刪除 IAM 身分，儲存貯體政策會在主體中顯示唯一識別符，而非 ARN。這些唯一 ID 不會重複使用，因此您可以安全地從所有政策陳述式中移除具有唯一識別符的主體。如需唯一識別符的詳細資訊，請參閱《IAM 使用者指南》中的 IAM 識別符。

授予匿名許可

> **Warning**
>
> 授予 Amason S3 儲存貯體的匿名存取時請小心。當您授予匿名存取時，全球所有人皆可存取您的儲存貯體。我們鄭重建議您絕不要授予任何種類的 S3 儲存貯體匿名寫入存取。

若要將許可授予所有人 (又稱為匿名存取)，請將萬用字元 ("*") 設為 Principal 值即可。例如，若將儲存貯體設定為網站，您會希望儲存貯體中所有物件都可公開存取。

```
"Principal":"*"
```

```
"Principal":{"AWS":"*"}
```

在"Principal": "*"以資源為基礎的策略中使用可讓任何人（即使未登入）存取您的資源。Allow AWS

將 "Principal" : { "AWS" : "*" } 與資源型政策中的 Allow 效用搭配使用，即可允許位於同一分割區的任何帳戶中的任何根使用者、IAM 使用者、擔任角色工作階段或聯合身分使用者存取您的資源。

對於匿名使用者，這兩種方法相同。如需詳細資訊，請參閱《IAM 使用者指南》中的 所有主體。

您不能使用萬用字元來比對部分主體名稱或 ARN。

> **Important**
>
> 因為任何人都可以建立 AWS 帳戶，所以這兩種方法的安全性層級是相等的，即使它們的功能不同。

限制資源許可

您也可以使用資源政策，針對可另提供給 IAM 主體使用的資源限制其存取權限。使用 Deny 陳述式防止存取。

下列範例會在未使用安全的傳輸通訊協定時封鎖存取：

```
{"Effect": "Deny",
 "Principal": "*",
 "Action": "s3:*",
 "Resource": <bucket ARN>,
 "Condition": {
   "Boolean": { "aws:SecureTransport" : "false"}
 }
}
```

使用 "Principal": "*" 對所有人套用此限制，是此政策的最佳實務，而不是嘗試只使用此方法拒絕對特定帳戶或主體的存取。

需要透過網 CloudFront 址存取

您可以要求使用者只使用 CloudFront URL 而不是 Amazon S3 URL 來存取您的 Amazon S3 內容。若要這麼做，請建立 CloudFront 原始存取控制 (OAC)。然後，變更 S3 資料的許可。在值區政策中，您可以設定 CloudFront 為主體，如下所示：

```
"Principal":{"Service":"cloudfront.amazonaws.com"}
```

使用政策中的Condition元素，只有在請求代表包含 S3 來源的 CloudFront 發佈時才允許 CloudFront 存取儲存貯體。

```
       "Condition": {
          "StringEquals": {
             "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/CloudFront-distribution-ID"
          }
       }
```

如需要透過 CloudFront URL 存取 S3 的詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的限制對 Amazon 簡單儲存服務來源的存取。如需使用 Amazon 的安全性和隱私權優勢的詳細資訊 CloudFront，請參閱設定安全存取和限制內容存取。

Amazon S3 的資源型政策範例

- 若要檢視 Amazon S3 儲存貯體的政策範例，請參閱Amazon S3 的存儲桶政策。
- 若要檢視存取點的原則範例，請參閱配置使用存取點的 IAM 原則。

Amazon S3 的政策動作

支援政策動作	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼条件下可以對什麼資源執行哪些動作。

JSON 政策的 Action 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

以下顯示 S3 API 操作和必要政策動作之間的不同類型的映射關係。

- O ne-to-one 映射具有相同名稱。例如，若要使用 PutBucketPolicy API 作業，則需要執行s3:PutBucketPolicy原則動作。
- O ne-to-one 映射具有不同名稱。例如，若要使用 ListObjectsV2 API 作業，則需要執行s3:ListBucket原則動作。
- O ne-to-many 映射。例如，若要使用 HeadObject API 作業，s3:GetObject則是必要的。此外，當您使用 S3 物件鎖定並想要取得物件的法律訴訟保留狀態或保留設定時，您也必須先執行對應的s3:GetObjectLegalHold或s3:GetObjectRetention政策動作，才能使用 HeadObject API 作業。
- M any-to-one 映射。例如，若要使用ListObjectsV2或 HeadBucket API 作業，則需要執行s3:ListBucket原則動作。

若要查看在政策中使用的 Amazon S3 動作清單，請參閱服務授權參考中 Amazon S3 定義的動作。如需 Amazon S3 API 操作的完整清單，請參閱 Amazon 簡單儲存服務 API 參考中的 Amazon S3 API 動作。

Amazon S3 中的政策動作會在動作前使用下列前置詞：

```
s3
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "s3:action1",
      "s3:action2"
         ]
```

儲存貯體操作

儲存貯體操作是在儲存貯體資源類型上運作的 S3 API 操作。例如：CreateBucket、ListObjectsV2 和 PutBucketPolicy。儲存貯體操作的 S3 政策動作要求儲存貯體政策或 IAM 身分型政策中的Resource元素必須是 S3 儲存貯體類型 Amazon 資源名稱 (ARN) 識別碼，採用下列範例格式。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
```

下列儲存貯體政策授予擁有帳戶 *12345678901* 的使用者 *Agua* 執行 ListObjectsV2 API 操作和列出 S3 儲存貯體中的物件的s3:ListBucket權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to list objects in the bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
        }
    ]
}
```

存取點原則中的儲存貯體作業

只有在基礎值區允許相同權限時，存取點政策中授予的權限才有效。使用 S3 Access Point 時，您必須將存取控制從儲存貯體委派給存取點，或將存取點政策中的相同權限新增至基礎儲存貯體的政策。如需詳細資訊，請參閱 配置使用存取點的 IAM 原則。在存取點政策中，儲存貯體操作的 S3 政策動作會要求您針對以下格式的Resource元素使用 accesspoint ARN。

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT"
```

下列存取點政策授予擁有帳戶 *12345678901* 的使用者 *Agua* s3:ListBucket 權限，以便透過 S3 存取點####存取點執行 ListObjectsV2 API 作業，以列出存取點關聯儲存貯體中的物件。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to list objects in the bucket through access point",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
```

```
            "Action": [
            "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT"
        }
    ]
}
```

> **Note**
>
> S3 存取點並非所有儲存貯體操作都支援。如需詳細資訊，請參閱 存取點與 S3 操作的相容性。

物件操作

物件操作是根據物件資源類型起作用的 S3 API 操作。例如：GetObject、PutObject 和 DeleteObject。物件操作的 S3 政策動作要求政策中的Resource元素必須是 S3 物件 ARN，採用下列範例格式。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
```

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/prefix/*"
```

> **Note**
>
> 物件 ARN 必須在值區名稱後加上正斜線，如前面的範例所示。

下列儲存貯體政策授予擁有帳戶 *12345678901* 的使用者 *Agua* 執行 PutObjectAPI 操作以將物件上傳到 S3 儲存貯體的s3:PutObject權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to upload objects",
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

存取點原則中的物件作業

當您使用 S3 存取點控制物件操作的存取時，您可以使用存取點政策。當您使用存取點政策時，物件操作的 S3 政策動作會要求您針對Resource元素使用 accesspoint ARN，格式如下：arn:aws:s3:region:account-id:accesspoint/access-point-name/object/resource對於使用存取點的物件作業，您必須在Resource元素中的整個存取點 ARN 之後加入/object/值。請見下方範例。

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT/object/*"
```

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT/object/prefix/*"
```

下列存取點原則授與擁有帳戶 *12345678901* 的使用者 *Agua* s3:GetObject 權限，可透過存取點*####*存取點在存取點關聯儲存貯體中的所有物件上執行 GetObjectAPI 作業。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to get objects through access point",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT/object/*"
```

```
        }
    ]
}
```

> **Note**
>
> S3 存取點並非支援所有物件操作。如需詳細資訊，請參閱 存取點與 S3 操作的相容性。

存取點作業

存取點操作是在accesspoint資源類型上運作的 S3 API 操作。例如：CreateAccessPoint、DeleteAccessPoint 和 GetAccessPointPolicy。存取點操作的 S3 政策動作只能用於 IAM 身分型政策，而不能用於儲存貯體政策或存取點政策。存取點作業需要以下列範例格式的Resource元素為 accesspoint ARN。

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT"
```

IAM ######## S3 ############# GetAccessPointPolicyAPI ## #s3:GetAccessPointPolicy###

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Grant permission to retrieve the access point policy of access
 point DOC-EXAMPLE-ACCESS-POINT",
            "Effect": "Allow",
            "Action": [
            "s3:GetAccessPointPolicy"
            ],
            "Resource": "arn:aws:s3:*:123456789012:access point/DOC-EXAMPLE-ACCESS-
POINT"
        }
    ]
}
```

當您使用存取點時，若要控制值區作業的存取，請參閱存取點原則中的儲存貯體作業；若要控制物件作業的存取，請參閱存取點原則中的物件作業。如需如何設定存取點原則的相關資訊，請參閱配置使用存取點的 IAM 原則。

物件 Lambda 存取點作業

藉助 Amazon S3 Object Lambda，您可將自己的程式碼新增至 Amazon S3 GET、LIST 和 HEAD 請求，以便在資料傳回應用程式時對其做出修改和處理。您可以透過物件 Lambda 存取點提出請求，該存取點的運作方式與透過其他存取點發出請求的運作方式相同。如需詳細資訊，請參閱 使用 S3 Object Lambda 轉換物件。

如需如何設定物件 Lambda 存取點作業原則的詳細資訊，請參閱設定 Object Lambda 存取點的 IAM 政策。

多區域存取點作業

多區域存取點提供了一個全域端點，應用程式可用來滿足來自多個 AWS 區域 S3 儲存貯體的請求。您可以使用多區域存取點，使用與單一區域相同的架構來建置多區域應用程式，然後在世界任何地方執行這些應用程式。如需詳細資訊，請參閱 Amazon S3 中的多區域存取點。

如需如何設定多區域存取點作業原則的相關資訊，請參閱多區域存取點政策範例。

Batch 工作作業

(Batch 作業) 工作作業是在工作資源類型上運作的 S3 API 作業。例如，DescribeJob 和 CreateJob。任務操作的 S3 政策動作只能用於 IAM 身分型政策，而不能用於儲存貯體政策。此外，工作作業要求 IAM 身分型政策中的Resource元素必須是下列範例格式的 job ARN。

```
"Resource": "arn:aws:s3:*:123456789012:job/*"
```

```
## IAM ######## S3 Batch ######## Job ### DescribeJobAPI ###s3:DescribeJob#
##
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow describing the Batch operation job DOC-EXAMPLE-JOB",
            "Effect": "Allow",
            "Action": [
            "s3:DescribeJob"
            ],
            "Resource": "arn:aws:s3:*:123456789012:job/DOC-EXAMPLE-JOB"
        }
    ]
```

```
}
```

S3 儲存鏡頭組態操作

如需如何設定 S3 儲存鏡頭組態操作的相關資訊，請參閱Amazon S3 Storage Lens 許可。

帳戶操作

帳戶操作是在帳戶層級運作的 S3 API 操作。例如，GetPublicAccessBlock（對於帳戶）。帳戶不是由 Amazon S3 定義的資源類型。帳戶操作的 S3 政策動作只能用於 IAM 身分型政策，而不能用於儲存貯體政策。此外，帳戶操作還要求 IAM 身分型政策中的Resource元素為。"*"

下列 IAM 身分型政策授予執行帳戶層級 GetPublicAccessBlockAPI 操作及擷取帳戶層級公用存取區塊設定的s3:GetAccountPublicAccessBlock權限。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"Allow retrieving the account-level Public Access Block settings",
         "Effect":"Allow",
         "Action":[
            "s3:GetAccountPublicAccessBlock"
         ],
         "Resource":[
            "*"
         ]
       }
    ]
}
```

Amazon S3 的政策範例

- 若要檢視 Amazon S3 身分識別型政策的範例，請參閱。Amazon S3 的基於身份的政策
- 若要檢視 Amazon S3 以資源為基礎的政策範例，請參閱Amazon S3 的存儲桶政策和配置使用存取點的 IAM 原則。

Amazon S3 的政策資源

支援政策資源	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Resource` JSON 政策元素可指定要套用動作的物件。陳述式必須包含 `Resource` 或 `NotResource` 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

部分 Amazon S3 API 動作支援多種資源。例如，`s3:GetObject`存取 EXAMPLE-RESOURCE-1 和 EXAMPLE-RESOURCE-2，因此主參與者必須具有存取這兩個資源的權限。若要在單一陳述式中指定多項資源，請使用逗號分隔 ARN。

```
"Resource": [
      "EXAMPLE-RESOURCE-1",
      "EXAMPLE-RESOURCE-2"
```

Amazon S3 中的資源包括儲存貯體、物件、存取點或任務。在政策中，使用儲存貯體、物件、存取點或任務的 Amazon 資源名稱 (ARN) 來識別資源。

若要查看 Amazon S3 資源類型及其 ARN 的完整清單，請參閱服務授權參考中 Amazon S3 定義的資源。若要了解可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon S3 定義的動作。

資源 ARN 的萬用字元

資源 ARN 中可以使用萬用字元。您可以在任何 ARN 區段內 (各部分以冒號分隔)，使用萬用字元 (* 和 ?)。星號 (*) 代表零或更多字元的任何組合，而問號 (?) 則代表任何單一字元。您可以在各區段使用多個數個 * 或 ? 字元，但萬用字元不能跨區段。

- 下列 ARN 在 ARN 的 relative-ID 部分中使用萬用字元 *，以識別 `examplebucket` 儲存貯體的所有物件。

  ```
  arn:aws:s3:::examplebucket/*
  ```

- 下列 ARN 用*來指示所有 S3 儲存貯體和物件。

  ```
  arn:aws:s3:::*
  ```

- 下列 ARN 在 `relative-ID` 部分中同時使用萬用字元 * 和 ?。它會識別儲存貯體中的所有物件，例如 `example1bucket`、`example2bucket`、`example3bucket` 等等。

```
arn:aws:s3:::example?bucket/*
```

資源 ARN 的政策變數

Amazon S3 ARN 中可以使用政策變數。在政策評估期間，這些預先定義的變數會為其對應值所取代。假設您將儲存貯體組織為資料夾集合，每位使用者各一個資料夾。資料夾名稱與使用者名稱相同。若要授予使用者其資料夾許可，您可在資源 ARN 中指定政策變數：

```
arn:aws:s3:::bucket_name/developers/${aws:username}/
```

在執行階段，當評估原則時，資源 ARN `${aws:username}` 中的變數會以提出要求之人員的使用者名稱取代。

Amazon S3 的政策範例

- 若要檢視 Amazon S3 身分識別型政策的範例，請參閱。Amazon S3 的基於身份的政策
- 若要檢視 Amazon S3 以資源為基礎的政策範例，請參閱Amazon S3 的存儲桶政策和配置使用存取點的 IAM 原則。

Amazon S3 的政策條件金鑰

支援服務特定政策條件金鑰	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Condition` 元素 (或 `Condition` 區塊) 可讓您指定使陳述式生效的條件。`Condition` 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 `Condition` 元素，或是在單一 `Condition` 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

每個 Amazon S3 條件金鑰都會對應至可設定條件的 API 允許的相同名稱請求標頭。Amazon S3 特定的條件索引鍵決定相同名稱請求標頭的行為。例如，`s3:VersionId`用來授與權限條件式權限的條件索引鍵會定義您在 GET Object 要求中設定之`versionId`查詢參數的行為。`s3:GetObjectVersion`

若要查看 Amazon S3 條件金鑰清單，請參閱服務授權參考中的 Amazon S3 條件金鑰。若要了解可以使用條件金鑰的動作和資源，請參閱 Amazon S3 定義的動作。

範例：限制物件上傳至具有特定儲存區類別的物件

假設帳戶 A (以帳戶 ID 123456789012 代表) 擁有一個儲存貯體。帳戶 A 系統管理員想要限制帳戶 A 中的使用者 Dave，以便 Dave 只能將物件上傳到隨儲存區類別一起儲存的`STANDARD_IA`值區。若要限制物件上傳至特定儲存體方案，帳戶 A 管理員可以使用 `s3:x-amz-storage-class` 條件金鑰，如下列儲存貯體政策範例所示。

```
{
                "Version": "2012-10-17",
                "Statement": [
                  {
                    "Sid": "statement1",
                    "Effect": "Allow",
                    "Principal": {
                      "AWS": "arn:aws:iam::123456789012:user/Dave"
                    },
                    "Action": "s3:PutObject",
                    "Resource": "arn:aws:s3:::example-s3-bucket1/*",
                    "Condition": {
                      "StringEquals": {
                        "s3:x-amz-storage-class": [
                          "STANDARD_IA"
                        ]
                      }
                    }
                  }
                ]
```

```
        }
```

在範例中，Condition 區塊指定的 StringEquals 條件會套用到指定的金鑰/值對 "s3:x-amz-acl":["public-read"]。您可使用一組預先定義的金鑰來表達條件。此範例使用 s3:x-amz-acl 條件索引鍵。此條件需要使用者在每個 PUT 物件要求中包含值為 public-read 的 x-amz-acl 標頭。

Amazon S3 的政策範例

- 若要檢視 Amazon S3 身分識別型政策的範例，請參閱。Amazon S3 的基於身份的政策
- 若要檢視 Amazon S3 以資源為基礎的政策範例，請參閱Amazon S3 的存儲桶政策和配置使用存取點的 IAM 原則。

Amazon S3 中的 ACL

支援 ACL	是

在 Amazon S3 中， AWS 帳戶 具有存取資源許可的存取控制清單 (ACL) 控制項。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

> **Important**
>
> Amazon S3 中的大多數新式使用案例不再需要使用 ACL。

如需使用 ACL 控制 Amazon S3 中存取的相關資訊，請參閱使用 ACL 管理存取。

Amazon S3 的 ABAC

支援 ABAC (政策中的標籤)	部分

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

如需根據標籤控制存取，請使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱 IAM 使用者指南中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱 IAM 使用者指南中的使用屬性型存取控制 (ABAC)。

若要檢視以身分識別為基礎的政策範例，以根據標籤限制對 S3 Batch 操作任務的存取，請參閱。使用任務標籤控制 S3 批次作業的許可

ABAC 和對象標籤

在 ABAC 原則中，物件會使用s3:標籤而非aws:標籤。若要根據物件標籤控制物件的存取，您可以使用下列標籤在原則的條件元素中提供標籤資訊：

- s3:ExistingObjectTag/*tag-key*
- s3:s3:RequestObjectTagKeys
- s3:RequestObjectTag/*tag-key*

如需有關使用物件標籤來控制存取的資訊，包括權限原則範例，請參閱標記與存取控制政策。

使用 Amazon S3 的臨時登入資料

支援臨時憑證	是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱 IAM 使用者指南中的切換至角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而非使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

Amazon S3 的轉發存取工作階段

支援轉寄存取工作階段 (FAS)	是

當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

- 當使用 SSE-KMS 加密物件時，Amazon S3 會使用 FAS AWS KMS 來進行呼叫以解密物件。如需詳細資訊，請參閱 使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)。
- S3 存取授權也使用 FAS。針對特定身分建立 S3 資料的存取權授與後，受權者會向 S3 Access Grants 請求臨時登入資料。S3 存取授權會從要求者取得暫時登入資料， AWS STS 並將認證分配給要求者。如需詳細資訊，請參閱 透過 S3 Access Grants 請求存取 Amazon S3 資料。

Amazon S3 的服務角色

支援服務角色	是

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務服務。

> **Warning**
>
> 變更服務角色的許可可可可可能會中斷 Amazon S3 功能。只有在 Amazon S3 提供指導時才編輯服務角色。

Amazon S3 的服務連結角色

支援服務連結角色	部分

服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

Amazon S3 支援 Amazon S3 Storage Lens 的服務連結角色。如需建立或管理 Amazon S3 服務連結角色的詳細資訊，請參閱讓 Amazon S3 Storage Lens 使用服務連結角色。

Amazon S3 服務作為主要

策略中的服務名稱	S3 功能	其他資訊
s3.amazonaws.com	S3 複寫	設定即時複製
s3.amazonaws.com	S3 事件通知	Amazon S3 事件通知
s3.amazonaws.com	S3 庫存	Amazon S3 清查
access-grants.s3.amazonaws.com	S3 Access Grants	註冊位置
batchoperations.s3.amazonaws.com	S3 批次操作	授予 Amazon S3 批次操作的許可
logging.s3.amazonaws.com	S3 伺服器存取記錄	啟用 Amazon S3 伺服器存取記錄日誌
storage-lens.s3.amazonaws.com	S3 Storage Lens	使用資料匯出檢視 Amazon S3 Storage Lens 指標

Amazon S3 中的政策和許可

此頁面提供 Amazon S3 中儲存貯體和使用者政策的概觀，並說明政策的基本元素。您可透過以下每個所列元素的連結，取得該元素的詳細資訊以及使用範例。

如需 Amazon S3 動作、資源和條件的完整清單，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

就其最基本意義而言，政策包含下列元素：

- 資源 - 政策適用的 Amazon S3 儲存貯體、物件、存取點或任務。使用儲存貯體、物件、存取點或任務的 Amazon Resource Name (ARN) 識別資源。

儲存貯體層級操作的範例：

- "Resource": "arn:aws:s3:::*bucket_name*".

物件層級操作的範例：

- "Resource": "arn:aws:s3:::*bucket_name/**" 代表儲存貯體中的所有物件。

- "Resource": "arn:aws:s3:::*bucket_name/prefix/**" 代表儲存貯體中具有特定字首的物件。

如需詳細資訊，請參閱 Amazon S3 的政策資源。

- 動作 – 針對每個資源，Amazon S3 支援一組操作。您可使用動作關鍵字，來識別允許 (或拒絕) 資源操作。

  例如，s3:ListBucket 許可允許使用者使用 Amazon S3 GET 儲存貯體 (列出物件) 操作。如需有關使用 Amazon S3 動作的詳細資訊，請參閱 Amazon S3 的政策動作。如需 Amazon S3 動作的完整清單，請參閱動作。
- 效果 – 當使用者請求特定動作時會產生什麼效果，可能是允許或拒絕。

  如果您不明確授予存取 (允許) 資源，即隱含拒絕存取。您也可以明確拒絕存取資源。您可以這樣做以確保使用者無法存取資源，即使不同的原則授予存取權限也一樣。如需詳細資訊，請參閱 IAM JSON 政策元素：Effect。
- 委託人 - 允許存取陳述式中動作與資源的帳戶或使用者。在儲存貯體政策中，委託人是身為此許可收件人的使用者、帳戶、服務或其他實體。如需詳細資訊，請參閱 值區政策的主體。
- 條件 – 政策何時生效的條件。您可以使用 AWS全金鑰和 Amazon S3 特定金鑰，在 Amazon S3 存取政策中指定條件。如需詳細資訊，請參閱 使用條件鍵值區政策範例。

下列範例儲存貯體政策顯示了影響、主體、動作和資源元素。該政策允許 Agua (帳戶*## ID* 中的使用者) s3:GetObjects3:GetBucketLocation，以及儲存貯體上的 s3:ListBucket Amazon S3 許可。awsexamplebucket1

```
{
    "Version": "2012-10-17",
    "Id": "ExamplePolicy01",
    "Statement": [
        {
            "Sid": "ExampleStatement01",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Akua"
            },
            "Action": [
                "s3:GetObject",
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::awsexamplebucket1/*",
                "arn:aws:s3:::awsexamplebucket1"
            ]
        }
    ]
}
```

如需完整的政策語言資訊，請參閱 IAM 使用者指南中的 IAM 和 IAM JSON 政策參考中的政策和許可。

許可委派

如果 AWS 帳戶 擁有資源，它可以將這些權限授與另一個資源 AWS 帳戶。該帳戶則可以將這些許可或其中的一些許可，委派給帳戶中的使用者。這稱為權限委派。但是從另一個帳戶接收權限的帳戶無法將權限跨帳戶委派給另一個 AWS 帳戶帳戶。

Amazon S3 儲存貯體和物件擁有權

儲存貯體與物件都是 Amazon S3 資源。依預設，只有資源擁有者可以存取這些資源。資源擁有者指的 AWS 帳戶 是建立資源的。例如：

- 您用來建立值區和上傳物件的資源擁有這些資源。 AWS 帳戶
- 如果您使用 AWS Identity and Access Management (IAM) 使用者或角色登入資料上傳物件，則 AWS 帳戶 該使用者或角色所屬的擁有物件。
- 值區擁有者可以將跨帳戶權限授與另一個帳戶 AWS 帳戶 (或其他帳戶中的使用者)，以便上傳物件。在這種情況下，上載 AWS 帳戶 的對象擁有這些對象。除了下列狀況之外，儲存貯體擁有者並不擁有其他帳戶所擁有之物件的許可：
  - 儲存貯體擁有者支付帳單。儲存貯體擁有者可拒絕對任何物件之存取，或刪除儲存貯體中的任何物件，而無須考慮其擁有者為何。
  - 儲存貯體擁有者可封存任何物件或是還原封存的物件，而無須考慮擁有者為誰。封存指的是指用於存放物件的儲存體方案。如需詳細資訊，請參閱 管理儲存生命週期。

擁有者和請求身分驗證

對儲存貯體的所有請求可能是已驗證或未驗證。已驗證的請求，必須包含能驗證要求傳送者身分的簽章值，未驗證的請求則沒有。如需請求身分驗證的詳細資訊，請參閱「提出要求」。

儲存貯體擁有者可以允許未驗證的請求。例如，當值區具有公用值區政策，或儲存貯體 ACL 特別授與或FULL_CONTROL存取All Users群組WRITE或匿名使用者時，便允許未驗證的PUT Object要求。如需公有儲存貯體政策和公開存取控制清單 (ACL) 的詳細資訊，請參閱「「公有」的意義」。

所有未驗證的請求是由匿名使用者提出。此使用者在 ACL 中是以特定的正式使用者 ID 65a011a29cdf8ec533ec3d1ccaae921c 呈現。如果物件是透過未驗證的請求上傳至儲存貯體，則匿名使用者會擁有該物件。預設的 ACL 會授與 FULL_CONTROL 給匿名使用者作為物件的擁有者。因此，Amazon S3 會允許未驗證的請求擷取物件或修改其 ACL。

為了防止物件遭到匿名使用者修改，建議您不要實作會允許對您的儲存貯體進行匿名寫入的政策，或是使用會允許匿名使用者對您的儲存貯體的寫入存取權的 ACL。您可以使用 Amazon S3 封鎖公開存取來強制此建議的行為。

如需封鎖公開存取的詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。如需 ACL 的詳細資訊，請參閱「存取控制清單 (ACL) 概觀」。

> **Important**
>
> 我們建議您不要使用 AWS 帳戶 root 使用者認證來發出已驗證的要求。而是建立 IAM 角色，然後授予該角色完整的存取。我們稱擁有此角色的使用者為管理員使用者。您可以使用指派給系統管理員角色的認證 (而非 AWS 帳戶 root 使用者認證) 與其互動 AWS 並執行工作，例如建立值區、建立使用者以及授與權限。如需詳細資訊，請參閱AWSIAM 使用者指南中的安全登入資料和 IAM 使用者指南中的安全最佳實務。

Amazon S3 的存儲桶政策

儲存貯體政策是以資源為基礎的政策，您可以使用這些政策來將存取許可授予 Amazon S3 儲存貯體及其中物件。只有儲存貯體擁有者可建立政策與儲存貯體的關聯。連接到儲存貯體的許可會套用至儲存貯體擁有者帳戶擁有的所有儲存貯體物件。這些權限不適用於其他人擁有的物件 AWS 帳戶。

S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來控制上傳至儲存貯體之物件的擁有權，以及停用或啟用存取控制清單 (ACL)。根據預設，物件擁有權設定為強制執行的儲存貯體擁有者設

定，而且所有 ACL 都會停用。儲存貯體擁有者擁有儲存貯體中的每個物件，並使用政策專門管理對資料的存取。

儲存貯體政策使用 JSON 型 AWS Identity and Access Management (IAM) 政策語言。您可以使用儲存貯體政策來新增或拒絕儲存貯體中物件的許可。儲存貯體政策可以允許或拒絕以政策中元素為基礎的請求。這些元素包括請求的申請者、S3 動作、資源，以及其他方面或條件 (例如，用來提出要求的 IP 地址)。

例如，您可以建立儲存貯體政策，執行下列動作：

- 授予其他帳戶跨帳戶許可，以將物件上傳至您的 S3 儲存貯體
- 請確定您 (儲存貯體擁有者) 具有已上載物件的完整控制權

如需詳細資訊，請參閱 Amazon S3 儲存貯體政策範例。

> **Important**
>
> 您無法使用儲存貯體政策來防止 S3 生命週期規則刪除或轉換。例如，即使儲存貯體政策拒絕所有主體的所有動作，S3 生命週期組態仍可正常運作。

本節中的主題提供了範例，並示範如何在 S3 主控台中新增儲存貯體政策。如需以身分為基礎的原則的資訊，請參閱。Amazon S3 的基於身份的政策如需有關儲存貯體政策語言的資訊，請參閱 Amazon S3 中的政策和許可

主題

- 使用 Amazon S3 主控台新增儲存貯體政策
- 使用儲存貯體政策控制來自 VPC 端點的存取
- Amazon S3 儲存貯體政策範例
- 使用條件鍵值區政策範例

使用 Amazon S3 主控台新增儲存貯體政策

您可以使用 AWS 政策產生器和 Amazon S3 主控台新增儲存貯體政策，或編輯現有的儲存貯體政策。儲存貯體政策是以資源為基礎的 AWS Identity and Access Management (IAM) 政策。您可以將值區政策新增至值區，以授與其他 AWS 帳戶 或 IAM 使用者存取該值區及其中物件的存取權限。物件許可只

會套用至該儲存貯體擁有者所建立的物件。如需儲存貯體政策的詳細資訊，請參閱「適用於 Amazon S3 的 Identity and Access Management」。

請務必解決安全性警告、錯誤、一般警告，以及 AWS Identity and Access Management Access Analyzer 建議，然後再儲存政策。IAM Access Analyzer 會比對 IAM 政策文法和最佳實務來執行政策檢查，以驗證您的政策。這些檢查會產生問題清單並提供可行的建議，協助您撰寫具有功能性且符合安全最佳實務的政策。若要進一步了解如何使用 IAM Access Analyzer 驗證政策，請參閱《IAM 使用者指南》中的 IAM Access Analyzer 政策驗證。若要檢視 IAM Access Analyzer 傳回的警告、錯誤和建議清單，請參閱 IAM Access Analyzer 政策檢查參考。

如需對政策錯誤進行故障診斷的指引，請參閱 針對 Amazon S3 中的拒絕存取 (403 禁止) 錯誤進行疑難排解。

建立或編輯儲存貯體政策

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇要建立儲存貯體政策的儲存貯體名稱，或您想編輯之儲存貯體政策的儲存貯體名稱。
4. 選擇許可索引標籤標籤。
5. 在 Bucket policy (儲存貯體政策) 下方，選擇 Edit (編輯)。Edit bucket policy (編輯儲存貯體政策) 頁面隨即出現。
6. 在 Edit bucket policy (編輯儲存貯體政策) 頁面上，執行下列其中一項動作：

   - 若要查看《Amazon S3 使用者指南》中的儲存貯體政策範例，請選擇 Policy examples (政策範例)。
   - 若要自動產生政策，或在 Policy (政策) 區段中編輯 JSON，請選擇 Policy generator (原則產生器)。

   如果您選擇「策略產生器」，則「 AWS 策略產生器」會在新視窗中開啟。

   a. 在 AWS Policy Generator (AWS 政策產生器) 頁面上，針對 Select Type of Policy (選取政策類型)，選擇 S3 Bucket Policy (S3 儲存貯體政策)。
   b. 在提供的欄位中輸入資訊，以新增陳述式，然後選擇 Add Statement (新增陳述式)。針對您想要新增的任意數量陳述式重複此步驟。如需這些欄位的詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

> **Note**
>
> 為了您的方便，Edit bucket policy (編輯儲存貯體政策) 頁面會在 Policy (政策) 文字欄位上方顯示目前儲存貯體的 Bucket ARN (Amazon Resource Name) (儲存貯體 ARN (Amazon 資源名稱))。您可以複製此 ARN，以在 AWS Policy Generator (政策產生器) 頁面上的陳述式中使用。

c. 完成新增陳述式後，選擇 Generate Policy (產生政策)。

d. 複製產生的政策文字，選擇 Close (關閉)，然後退回 Amazon S3 主控台中的 Edit bucket policy (編輯儲存貯體政策) 頁面。

7. 在 [原則] 方塊中，編輯現有政策，或從 [原則產生器] 貼上值區 AWS 原則。請務必先處理安全性警告、錯誤、一般警告，以及建議，然後再儲存政策。

> **Note**
>
> 儲存貯體政策的大小限制為 20 KB。

8. (選用) 選擇右下角的 Preview external access (預覽外部存取)，以預覽新政策會如何影響資源的公開和跨帳戶存取權。在儲存政策之前，您可以檢查它是否引入新的 IAM Access Analyzer 問題清單，或是解決現有的問題清單。如果您沒有看到作用中的分析器，請選擇 Go to Access Analyzer (移至 Access Analyzer)，以在 IAM Access Analyzer 中建立帳戶分析器。如需詳細資訊，請參閱《IAM 使用者指南》中的預覽存取。

9. 選擇 Save changes (儲存變更)，這會讓您回到 Permissions (許可) 索引標籤。

使用儲存貯體政策控制來自 VPC 端點的存取

您可以使用 Amazon S3 儲存貯體政策，控制來自特定虛擬私有雲端 (VPC) 端點或特定 VPC 對儲存貯體的存取。本節包含範例儲存貯體政策，可用於從 VPC 端點控制 Amazon S3 儲存貯體存取。若要了解如何設定 VPC 端點，請參閱《VPC 使用指南》中的 VPC 端點。

VPC 可讓您將 AWS 資源啟動到您定義的虛擬網路中。VPC 端點可讓您在 VPC 和另一個端點之間建立私人連線。 AWS 服務此私人連線不需要透過網際網路、透過虛擬私人網路 (VPN) 連線、NAT 執行個體或透過存取 AWS Direct Connect。

適用於 Amazon S3 的 VPC 端點是 VPC 內只允許連線到 Amazon S3 的邏輯實體。VPC 端點會將請求路由至 Amazon S3，並將回應路由回到 VPC。VPC 端點僅供要求路由連接方式 Amazon S3 公有端

點和 DNS 名稱仍然適用於 VPC 端點。如需將 VPC 端點與 Amazon S3 搭配使用的重要資訊，請參閱 VPC 使用者指南中的 Amazon S3 閘道端點和閘道端點。

適用於 Amazon S3 的 VPC 端點提供兩種方式來控制對 Amazon S3 資料的存取：

- 您可以控制經由特定 VPC 端點允許的請求、使用者或群組。如需有關此類型存取控制的資訊，請參閱《VPC 使用手冊》中的使用端點策略控制對 VPC 端點的存取。
- 您可以使用 Amazon S3 儲存貯體政策，控制哪些 VPC 或 VPC 端點可以存取您的儲存貯體。如需這類儲存貯體政策存取控制的範例，請參閱下列限制存取主題。

主題

> **Important**
>
> 針對本節所述的 VPC 端點套用 Amazon S3 儲存貯體政策時，您可能會無意中封鎖對儲存貯體的存取。來自您 VPC 端點，旨在特別限制儲存貯體存取連線的儲存貯體許可，可能會封鎖所有對儲存貯體的連線。如需有關如何修正此問題的詳細資訊，請參閱如何在我的儲存貯體原則具有錯誤的 VPC 或 VPC 端點識別碼時修正？ 在AWS Support 知識中心。

限制特定 VPC 端點的存取

下列 Amazon S3 儲存貯體政策範例限制只能從 ID 為 `vpce-1a2b3c4d` 的 VPC 端點存取特定儲存貯體 `awsexamplebucket1`。如果未使用指定的端點，則政策會拒絕對值區的所有存取。`aws:SourceVpce`條件會指定端點。此`aws:SourceVpce`條件不需要 VPC 端點資源的 Amazon 資源名稱 (ARN)，只需要 VPC 端點識別碼。如需在政策中使用條件的詳細資訊，請參閱「使用條件鍵值區政策範例」。

> **Important**
>
> - 使用下列範例政策之前，請以適合您使用案例的適當值取代 VPC 端點 ID。否則，您將無法存取儲存貯體。
> - 此原則會停用對指定值區的主控台存取，因為主控台要求不是來自指定的 VPC 端點。

```
{
   "Version": "2012-10-17",
   "Id": "Policy1415115909152",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPCE-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::awsexamplebucket1",
                    "arn:aws:s3:::awsexamplebucket1/*"],
       "Condition": {
         "StringNotEquals": {
           "aws:SourceVpce": "vpce-1a2b3c4d"
         }
       }
     }
   ]
}
```

限制特定 VPC 的存取

您可以使用 aws:SourceVpc 條件，建立可限制存取特定 VPC 的儲存貯體政策。如果您在相同的 VPC 中設定多個 VPC 端點，而且想要管理所有端點對 Amazon S3 儲存貯體的存取，則這十分有用。下列政策範例拒絕 VPC vpc-111bbb22 外的任何人存取 awsexamplebucket1 和其物件。如果未使用指定的 VPC，則原則會拒絕對值區的所有存取。此陳述式不會授與值區的存取權。若要授予存取權，您必須新增個別Allow陳述式。vpc-111bbb22條件金鑰不需要 VPC 資源的 ARN，只需要 VPC 識別碼。

> **Important**
>
> - 使用下列範例政策之前，請以適合您使用案例的適當值取代 VPC ID。否則，您將無法存取儲存貯體。
> - 此原則會停用對指定值區的主控台存取，因為主控台要求不是來自指定的 VPC。

```
{
   "Version": "2012-10-17",
   "Id": "Policy1415115909153",
   "Statement": [
```

```
    {
      "Sid": "Access-to-specific-VPC-only",
      "Principal": "*",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::awsexamplebucket1",
                   "arn:aws:s3:::awsexamplebucket1/*"],
      "Condition": {
        "StringNotEquals": {
          "aws:SourceVpc": "vpc-111bbb22"
        }
      }
    }
  ]
}
```

Amazon S3 儲存貯體政策範例

使用 Amazon S3 儲存貯體政策，您可以安全地存取儲存貯體中的物件，以便只有具有適當許可的使用者才能存取它們。您甚至可以防止已驗證但沒有適當許可的使用者存取 Amazon S3 資源。

本節顯示儲存貯體政策之一般使用案例的範例。這些範例策略會使用 *example-s3-bucket* 作為資源值。若要測試這些政策，請將 *user input placeholders* 取代為您自己的資訊 (例如儲存貯體名稱)。

若要授予或拒絕一組物件的許可，您可以在 Amazon Resource Name (ARN) 和其他值上使用萬用字元 (*)。例如，您可以控制物件群組的存取權，這些群組以通用前置詞開頭或以特定副檔名結尾，例如 `.html`。

如需有關 AWS Identity and Access Management (IAM) 政策語言的詳細資訊，請參閱Amazon S3 中的政策和許可。

> **Note**
>
> 使用 Amazon S3 主控台來測試許可時，您必須授予主控台所需的其他許可：`s3:ListAllMyBuckets`、`s3:GetBucketLocation` 和 `s3:ListBucket`。如需授予使用者許可並使用主控台測試這些許可的演練範例，請參閱 使用使用者政策來控制對儲存貯體的存取。

建立值區政策的其他資源包括：

- 如需建立儲存貯體政策時可使用的 IAM 政策動作、資源和條件金鑰清單，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。
- 如需建立 S3 政策的指引，請參閱 使用 Amazon S3 主控台新增儲存貯體政策。
- 若要對政策錯誤進行故障排除，請參閱 針對 Amazon S3 中的拒絕存取 (403 禁止) 錯誤進行疑難排解。

主題

- 將唯讀權限授予公開匿名使用者
- 需要加密
- 使用標準 ACL 管理儲存貯體
- 使用物件標記來管理物件存取
- 使用全域條件金鑰管理物件存取
- 根據特定 IP 地位址管理存取
- 根據 HTTP 或 HTTPS 請求管理存取
- 管理使用者對特定資料夾的存取
- 管理存取日誌的存取
- 管理對 Amazon CloudFront OAI 的訪問
- 管理對 Amazon S3 Storage Lens 的存取
- 管理 S3 庫存、S3 分析和 S3 庫存報告的許可
- 需要 MFA
- 防止使用者刪除物件

將唯讀權限授予公開匿名使用者

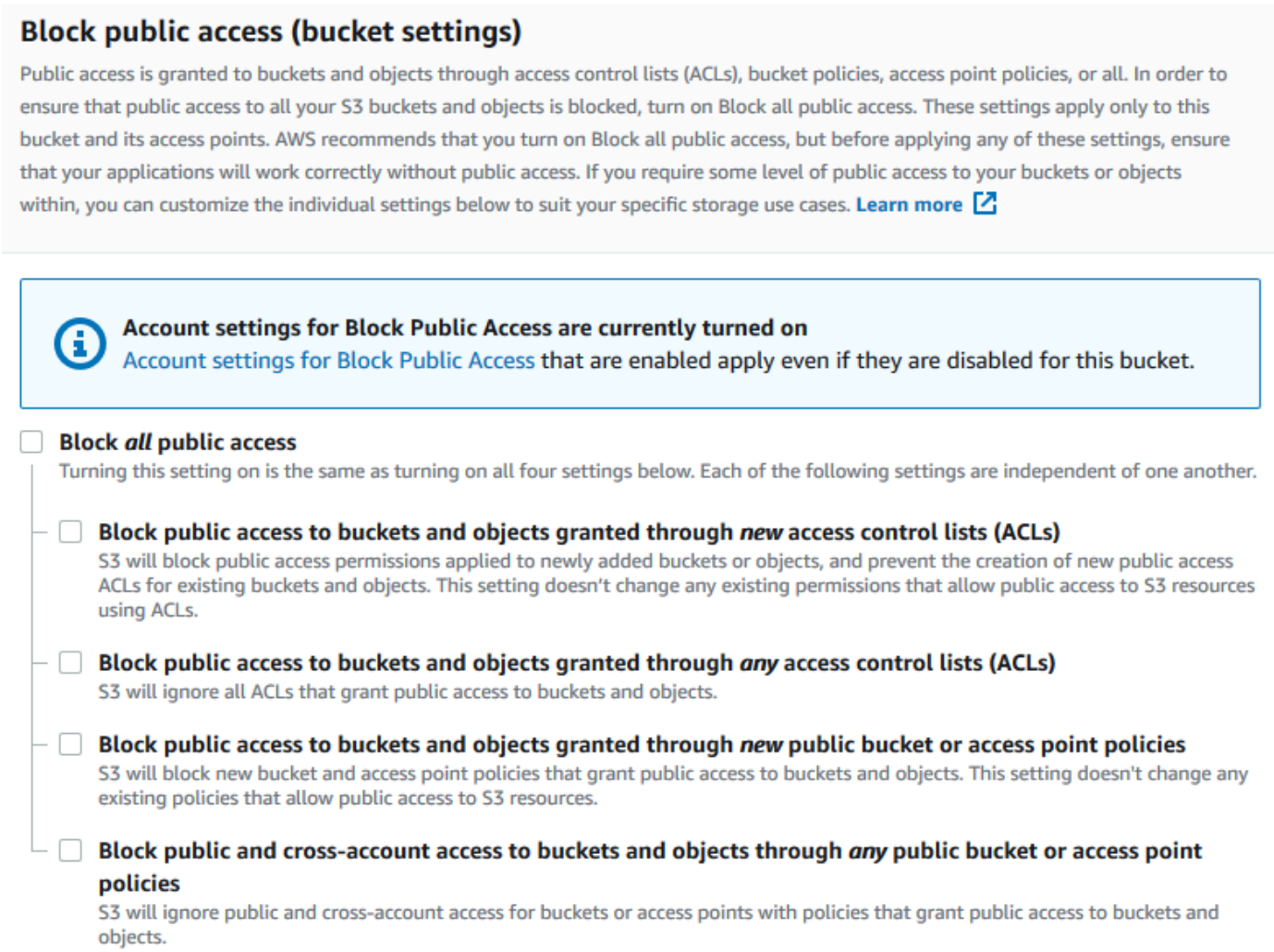
您可以使用政策設定將存取權授予公開匿名使用者，如果您將儲存貯體設定為靜態網站，這很有用。您必須針對儲存貯體停用封鎖公有存取權。如需有關如何執行此作業以及所需原則的詳細資訊，請參閱設定網站存取許可。若要了解如何針對相同目的設定更嚴格的政策，請參閱如何授與 Amazon S3 儲存貯體中某些物件的公開讀取存取權限？ 在 AWS 知識中心。

根據預設，Amazon S3 會封鎖對帳戶和儲存貯體的公開存取。如想要使用儲存貯體託管靜態網站，您可使用這些步驟編輯封鎖公有存取設定：

> **Warning**
>
> 在完成此步驟之前，請檢閱 封鎖對 Amazon S3 儲存體的公開存取權 以確保您了解並接受允許公開存取所涉及的風險。當您關閉封鎖公開存取設定以公開儲存貯體時，網際網路上的任何人都可以存取您的儲存貯體。我們建議您封鎖對儲存貯體的所有公用存取權。

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 選擇已設定為靜態網站的儲存貯體名稱。
3. 選擇 Permissions (許可)。
4. 在 Block public access (bucket settings) (封鎖公開存取 (儲存貯體設定)) (封鎖公開存取 (儲存貯體設定)) 下，選擇 Edit (編輯)。
5. 清除 Block all public access (封鎖所有公開存取)，然後選擇 Save changes (儲存變更)。

> **Warning**
>
> 在完成此步驟之前，請檢閱 封鎖對 Amazon S3 儲存體的公開存取權 以確保您了解並接受允許公開存取所涉及的風險。當您關閉封鎖公開存取設定以公開儲存貯體時，網際網路上的任何人都可以存取您的儲存貯體。我們建議您封鎖對儲存貯體的所有公用存取權。



Amazon S3 會關閉儲存貯體的封鎖公開存取設定。若要建立公開的靜態網站，在新增儲存貯體原則之前，可能還需要針對您的帳戶編輯封鎖公開存取設定。如果帳戶的封鎖公開存取設定目前已開啟，您在 封鎖公開存取 (儲存貯體設定) 下會看到附註。

需要加密

寫入儲存貯體的所有物件都需要 SSE-KMS

下列範例政策要求寫入儲存貯體的每個物件都必須使用 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS) 使用伺服器端加密進行加密。如果物件未使用 SSE-KMS 加密，則會拒絕該要求。

```
{
"Version": "2012-10-17",
"Id": "PutObjPolicy",
```

```
"Statement": [{
  "Sid": "DenyObjectsThatAreNotSSEKMS",
  "Principal": "*",
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
  "Condition": {
    "Null": {
      "s3:x-amz-server-side-encryption-aws-kms-key-id": "true"
    }
  }
}]
}
```

寫入儲存貯體的所有物件都需要 SSE-KMS 搭配特定 AWS KMS key

下列範例政策會拒絕任何物件寫入儲存貯體 (如果這些物件未使用特定的 KMS 金鑰 ID 搭配 SSE-KMS 加密的話)。即使物件是透過 SSE-KMS 使用每個要求標頭或儲存貯體預設加密，如果尚未使用指定的 KMS 金鑰加密物件，物件也無法寫入儲存貯體。請務必將本範例中使用的 KMS 金鑰 ARN 取代為您自己的 KMS 金鑰 ARN。

```
{
"Version": "2012-10-17",
"Id": "PutObjPolicy",
"Statement": [{
  "Sid": "DenyObjectsThatAreNotSSEKMSWithSpecificKey",
  "Principal": "*",
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
  "Condition": {
    "ArnNotEqualsIfExists": {
      "s3:x-amz-server-side-encryption-aws-kms-key-id": "arn:aws:kms:us-
east-2:111122223333:key/01234567-89ab-cdef-0123-456789abcdef"
    }
  }
}]
}
```

使用標準 ACL 管理儲存貯體

將許可授予多個帳戶，以上傳物件或設定物件 ACL 進行公開存取

下列範例原則會授s3:PutObject與和s3:PutObjectAcl權限給多個 AWS 帳戶。此外，範例原則要求這些作業的任何要求都必須包含public-read固定存取控制清單 (ACL)。如需詳細資訊，請參閱 Amazon S3 的政策動作 及 Amazon S3 的政策條件金鑰。

> **Warning**
>
> public-read 標準 ACL 可讓全世界的任何人檢視您儲存貯體中的物件。授予對 Amazon S3 bucket 儲存貯體的匿名存取，或停用封鎖公開存取設定時，請小心。當您授予匿名存取時，全球所有人皆可存取您的儲存貯體。我們建議您永遠不要授予匿名存取您的 Amazon S3 儲存貯體，除非您特別需要 (例如使用靜態網站託管時)。如果您想要針對靜態網站託管啟用封鎖公開存取設定，請參閱教學課程：在 Amazon S3 上設定靜態網站。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AddPublicReadCannedAcl",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root",
                    "arn:aws:iam::444455556666:root"
                ]
            },
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": [
                        "public-read"
                    ]
                }
            }
        }
```

```
    ]
}
```

授予跨帳戶許可，以在確保儲存貯體擁有者具有完全控制時上傳物件

下列範例顯示如何允許其他人將物件上載 AWS 帳戶 至您的值區，同時確保您擁有上載物件的完全控制權。只有在上載時包含bucket-owner-full-control固定 ACL 時，此策略才會授與特定 AWS 帳戶 (*111122223333*) 上載物件的能力。政策中的 StringEquals 條件指定 s3:x-amz-acl 條件金鑰來表示標準 ACL 需求。如需詳細資訊，請參閱 Amazon S3 的政策條件金鑰。

```
{
   "Version":"2012-10-17",
   "Statement":[
     {
       "Sid":"PolicyForAllowUploadWithACL",
       "Effect":"Allow",
       "Principal":{"AWS":"111122223333"},
       "Action":"s3:PutObject",
       "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
       "Condition": {
         "StringEquals": {"s3:x-amz-acl":"bucket-owner-full-control"}
       }
     }
   ]
}
```

使用物件標記來管理物件存取

允許使用者只讀取具有特定索引標籤金鑰和值的物件

下列許可政策會限制使用者只能讀取具有 environment: production 索引標籤金鑰和值的物件。此政策會使用 s3:ExistingObjectTag 條件金鑰來指定索引標籤金鑰和值。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Principal":{
            "AWS":"arn:aws:iam::111122223333:role/JohnDoe"
         },
         "Effect":"Allow",
         "Action":[
```

```
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
         "Condition":{
            "StringEquals":{
               "s3:ExistingObjectTag/environment":"production"
            }
         }
      }
   ]
}
```

限制使用者可以新增哪些物件索引標籤金鑰

下列範例政策授予使用者執行 s3:PutObjectTagging 動作的許可，允許使用者將索引標籤新增至現有的物件。此條件使用 s3:RequestObjectTagKeys 條件金鑰指定允許的索引標籤金鑰，例如 Owner 或 CreationDate。如需詳細資訊，請參閱《IAM 使用者指南》中的建立測試多個金鑰值的條件。

此政策可確保請求中指定的每個索引標籤金鑰都是授權的標籤金鑰。條件中的 ForAnyValue 限定詞可確保請求中至少會出現其中一個指定的金鑰。

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
            "arn:aws:iam::111122223333:role/JohnDoe"
         ]
       },
 "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"ForAnyValue:StringEquals": {"s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        }
      }
```

```
    }
  ]
}
```

允許使用者新增物件標籤時，需要特定索引標籤金鑰和值

下列範例政策授予使用者執行 s3:PutObjectTagging 動作的許可，允許使用者將標籤新增至現有的物件。此條件需要使用者包括值設為 *X* 的特定索引標籤 (例如 *Project*)。

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
       "arn:aws:iam::111122223333:user/JohnDoe"
         ]
       },
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"StringEquals": {"s3:RequestObjectTag/Project": "X"
        }
      }
    }
  ]
}
```

允許使用者只新增具有特定物件索引標籤金鑰和值的物件

下列範例政策會授與使用者執行 s3:PutObject 動作的許可，讓他們可將物件新增至儲存貯體。不過，Condition 陳述式會限制所上載物件上允許的索引標籤金鑰和值。在此範例中，使用者只能將具有值設為 *Finance* 的特定索引標籤金鑰 (*Department*) 的物件新增至儲存貯體。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Principal":{
            "AWS":[
                 "arn:aws:iam::111122223333:user/JohnDoe"
         ]
```

```
        },
        "Effect": "Allow",
        "Action": [
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        ],
        "Condition": {
            "StringEquals": {
                "s3:RequestObjectTag/Department": "Finance"
            }
        }
    }]
}
```

使用全域條件金鑰管理物件存取

全域條件索引鍵是具有aws前置詞的條件內容索引鍵。 AWS 服務 可以支援全域條件金鑰或包含服務前置詞的服務特定金鑰。您可以使用 JSON 政策的 Condition 元素，來比較請求中的金鑰和您在政策中指定的金鑰值。

限制只能存取 Amazon S3 伺服器存取日誌交付

在下列範例儲存貯體政策中，aws:SourceArn全域條件金鑰是用來比較資源的 Amazon 資源名稱 (ARN)，並使用政策中指定的 ARN 發出 service-to-service 請求。aws:SourceArn 全域條件金鑰用來防止 Amazon S3 服務在服務之間用作交易期間的混淆代理人。只允許 Amazon S3 服務將物件新增至 Amazon S3 儲存貯體。

此範例儲存貯體政策只向日誌記錄服務主體 (logging.s3.amazonaws.com) 授予 s3:PutObject 許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutObjectS3ServerAccessLogsPolicy",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET-logs/*",
```

```
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111111111111"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::EXAMPLE-SOURCE-BUCKET"
                }
            }
        },
        {
            "Sid": "RestrictToS3ServerAccessLogs",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET-logs/*",
            "Condition": {
                "ForAllValues:StringNotEquals": {
                    "aws:PrincipalServiceNamesList": "logging.s3.amazonaws.com"
                }
            }
        }
    ]
}
```

僅允許存取您的組織

如果您想要求存取資源的所有 IAM 主體都是來自組織 AWS 帳戶 中的資源 (包括 AWS Organizations 管理帳戶)，則可以使用aws:PrincipalOrgID全域條件金鑰。

若要授予或限制此類型的存取權，請在儲存貯體政策中定義 aws:PrincipalOrgID 條件，並將值設為您的組織 ID。組織 ID 是用來控制對儲存貯體的存取。當您使用 aws:PrincipalOrgID 條件時，來自儲存貯體政策的許可也會套用至新增至組織的所有新帳戶。

以下是資源型儲存貯體政策的範例，您可以使用此政策，授予組織中特定 IAM 主體存取儲存貯體的權限。透過將 aws:PrincipalOrgID 全域條件金鑰新增至儲存貯體政策，現在需要主體帳戶位於您的組織中，才能取得資源的存取權。即使您在授予存取權時意外指定了不正確的帳戶，aws:PrincipalOrgID 全域條件金鑰也會作為額外的防護。當此全域金鑰用於政策時，其可防止指定組織以外的所有主體存取 S3 儲存貯體。只有來自所列組織中帳戶的主體才能取得資源的存取權。

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

```
        "Sid": "AllowGetObject",
        "Principal": {
            "AWS": "*"
        },
        "Effect": "Allow",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
        "Condition": {
            "StringEquals": {
                "aws:PrincipalOrgID": ["o-aa111bb222"]
            }
        }
    }]
}
```

根據特定 IP 地位址管理存取

限制特定 IP 地址的存取

下列範例拒絕所有使用者對指定 S3 儲存貯體中的物件執行任何 Amazon S3 操作，除非請求源自指定的 IP 地址範圍。

> **Note**
>
> 限制對特定 IP 地址的存取時，請確定您也指定了哪些 VPC 端點、VPC 來源 IP 地址或外部 IP 地址可以存取 S3 儲存貯體。否則，如果您的政策拒絕所有使用者對您儲存貯體中的物件執行任何 S3 操作，而未設置任何適當的許可，您可能會失去對儲存貯體的存取權。

此政策的 `Condition` 陳述式會將 *192.0.2.0/24* 識別為允許之網際網路通訊協定第 4 版 (IPv4) IP 地址的範圍。

該`Condition`塊使用`NotIpAddress`條件和`aws:SourceIp`條件鍵，這是一個 AWS 寬條件鍵。`aws:SourceIp` 條件鍵僅可用於公有 IP 地址範圍。如需這些條件索引鍵的詳細資訊，請參閱「Amazon S3 的政策條件金鑰」。`aws:SourceIp` IPv4 值會使用標準 CIDR 表示法。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

> **Warning**
>
> 使用此政策之前，請以適當的使用案例值取代此範例中的 *192.0.2.0/24* IP 地址範圍。否則，您將失去存取儲存貯體的能力。

```
{
    "Version": "2012-10-17",
    "Id": "S3PolicyId1",
    "Statement": [
        {
            "Sid": "IPAllow",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ],
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "192.0.2.0/24"
                }
            }
        }
    ]
}
```

同時允許 IPv4 和 IPv6 地址

當您開始使用 IPv6 地址時，建議除了現有 IPv4 範圍之外，還使用 IPv6 地址範圍來更新組織的所有政策。這樣做有助於確保政策在您轉換為 IPv6 時繼續運作。

下列儲存貯體政策範例示範如何混合使用 IPv4 與 IPv6 地址範圍，以涵蓋組織中所有的有效 IP 地址。政策範例允許存取 IP 地址範例 *192.0.2.1* 與 *2001:DB8:1234:5678::1*，並且拒絕存取地址 *203.0.113.1* 與 *2001:DB8:1234:5678:ABCD::1*。

`aws:SourceIp` 條件鍵僅可用於公有 IP 地址範圍。`aws:SourceIp` 的 IPv6 值必須為標準 CIDR 格式。針對 IPv6，我們支援使用 `::` 代表 0 的範圍 (例如，`2001:DB8:1234:5678::/64`)。如需詳細資訊，請參閱《IAM 使用者指南》中的 IP 位址條件運算子。

> **Warning**
>
> 使用此政策之前，請以適當的使用案例值取代此範例中的 IP 地址範圍。否則，您可能會失去存取儲存貯體的能力。

```
{
    "Id": "PolicyId2",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowIPmix",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
                        "2001:DB8:1234:5678::/64"
                    ]
                },
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "203.0.113.0/24",
                        "2001:DB8:1234:5678:ABCD::/80"
                    ]
                }
            }
        }
    ]
}
```

根據 HTTP 或 HTTPS 請求管理存取

限制僅 HTTPS 請求才能存取

如果您想要防止潛在攻擊者操控網路流量，可以使用 HTTPS (TLS) 來僅允許加密連線，同時限制 HTTP 請求存取您的儲存貯體。若要判斷請求是 HTTP 還是 HTTPS，請在 S3 儲存貯體政策中使用 aws:SecureTransport 全域條件金鑰。`aws:SecureTransport` 條件金鑰會檢查是否已使用 HTTP 傳送請求。

如果請求傳回 `true`，則請求是透過 HTTPS 傳送。如果請求傳回 `false`，則請求是透過 HTTP 傳送。然後，您可以根據所需的請求配置來允許或拒絕對儲存貯體的存取。

在下列範例中，儲存貯體政策明確地拒絕 HTTP 請求。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "RestrictToTLSRequestsOnly",
        "Action": "s3:*",
        "Effect": "Deny",
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        ],
        "Condition": {
            "Bool": {
                "aws:SecureTransport": "false"
            }
        },
        "Principal": "*"
    }]
}
```

限制對特定 HTTP 參照的存取

假設您擁有具備網域名稱 (*www.example.com* 或 *example.com*) 的網站，其中包含您儲存貯體 (名為 *example-s3-bucket*) 中存放之照片和影片的連結。依預設，所有 Amazon S3 資源均為私有資源，因此只 AWS 帳戶 有建立資源的資源可以存取。

若要允許網站中這些物件的讀取存取，您可以新增一個儲存貯體政策，允許條件為 GET 請求必須源自於特定網頁的 s3:GetObject 許可。下列政策會使用 StringLike 條件搭配 aws:Referer 條件金鑰來限制請求。

```
{
  "Version":"2012-10-17",
  "Id":"HTTP referer policy example",
  "Statement":[
    {
      "Sid":"Allow only GET requests originating from www.example.com and
 example.com.",
      "Effect":"Allow",
      "Principal":"*",
      "Action":["s3:GetObject","s3:GetObjectVersion"],
      "Resource":"arn:aws:s3:::example-s3-bucket/*",
```

```
      "Condition":{
        "StringLike":{"aws:Referer":["http://www.example.com/*","http://example.com/
*"]}
      }
    }
  ]
}
```

請確定您使用的瀏覽器在請求中包含 HTTP `referer` 標頭。

> **Warning**
>
> 我們建議您使用 `aws:Referer` 條件金鑰時要小心。包含公開已知 HTTP 推薦者標頭值相當危險。未授權方可以使用修改的或自訂瀏覽器來提供他們選擇的任何 `aws:Referer` 值。因此，請勿用於防`aws:Referer`止未經授權的人士提出直接 AWS 請求。
> `aws:Referer` 條件金鑰僅用於允許客戶保護其數位內容 (例如存放在 Amazon S3 中的內容)，使其不被未經授權的第三方網站參考。如需詳細資訊，請參閱《IAM 使用者指南》中的 aws:Referer。

管理使用者對特定資料夾的存取

授予使用者存取特定資料夾的權限

假設您正在嘗試授予使用者存取特定資料夾的權限。如果 IAM 使用者和 S3 儲存貯體屬於同一個儲存貯體 AWS 帳戶，則可以使用 IAM 政策授與使用者特定儲存貯體資料夾的存取權。透過此方式，您不需要更新儲存貯體政策來授予存取權。您可以將 IAM 政策新增至多個使用者可切換的 IAM 角色。

如果 IAM 身分和 S3 儲存貯體屬於不同 AWS 帳戶，則您必須同時在 IAM 政策和儲存貯體政策中授予跨帳戶存取權。如需如何授予跨帳户存取權的詳細資訊，請參閱授予跨帳戶儲存貯體許可的儲存貯體擁有者。

下列範例儲存貯體政策僅授予 *JohnDoe* 其資料夾 (home/*JohnDoe*/) 的完整主控台存取權。透過建立 home 資料夾並將適當的許可授予使用者，您可以讓多個使用者共用單一儲存貯體。此政策包含三個 `Allow` 陳述式：

- *AllowRootAndHomeListingOfCompanyBucket*：允許使用者許 (*JohnDoe*) 列出 *DOC-EXAMPLE-BUCKET* 儲存貯體根層級和 home 資料夾中的物件。此陳述式還允許使用者透過使用主控台搜尋字首 home/。

- *AllowListingOfUserFolder*：允許使用者 (*JohnDoe*) 列出 home/*JohnDoe*/ 資料夾和任何子資料夾中的所有物件。
- *AllowAllS3ActionsInUserFolder*：透過授予Read、Write 和 Delete 許可來允許使用者執行所有 Amazon S3 動作。許可限於儲存貯體擁有者的主資料夾。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRootAndHomeListingOfCompanyBucket",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Effect": "Allow",
            "Action": ["s3:ListBucket"],
            "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
            "Condition": {
                "StringEquals": {
                    "s3:prefix": ["", "home/", "home/JohnDoe"],
                    "s3:delimiter": ["/"]
                }
            }
        },
        {
            "Sid": "AllowListingOfUserFolder",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Action": ["s3:ListBucket"],
            "Effect": "Allow",
            "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
            "Condition": {
                "StringLike": {
                    "s3:prefix": ["home/JohnDoe/*"]
                }
            }
        },
        {
```

```
            "Sid": "AllowAllS3ActionsInUserFolder",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Action": ["s3:*"],
            "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/home/JohnDoe/*"]
        }
    ]
}
```

管理存取日誌的存取

授予 Application Load Balancer 的存取權以啟用存取

對 Application Load Balancer 啟用存取記錄時，您必須指定 S3 儲存貯體的名稱，負載平衡器將在其中存放日誌。儲存貯體必須具有附加的政策，以授予 Elastic Load Balancing 寫入儲存貯體的許可。

在下列範例中，儲存貯體政策會授予 Elastic Load Balancing (ELB) 將存取日誌寫入儲存貯體的許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": "arn:aws:iam::elb-account-id:root"
      },
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::example-s3-bucket/prefix/AWSLogs/111122223333/*"
    }
  ]
}
```

> **Note**
>
> 務必針對您的 AWS 區域將 *elb-account-id* 取代為 Elastic Load Balancing 的 AWS 帳戶 ID。如需 Elastic Load Balancing 區域的清單，請參閱《Elastic Load Balancing 使用者指南》中的將政策附加到 Amazon S3 儲存貯體。

如果您 AWS 區域 未出現在支援的 Elastic Load Balancing 區域清單中，請使用下列原則，該原則會授與指定記錄傳遞服務的權限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
       "Principal": {
         "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
          },
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::example-s3-bucket/prefix/AWSLogs/111122223333/*"
    }
  ]
}
```

然後，務必啟用 Elastic Load Balancing 存取日誌來設定這些存取日誌。您可以建立測試檔案來驗證儲存貯體許可。

管理對 Amazon CloudFront OAI 的訪問

授予 Amazon CloudFront OAI 的許可

下列範例儲存貯體政策授予 CloudFront 來源存取身分識別 (OAI) 權限，以取得 (讀取) S3 儲存貯體中的所有物件。您可以使用 CloudFront OAI 允許使用者透過 Amazon S3 存取儲存貯體中的物件，CloudFront 但不能直接存取儲存貯體中的物件。如需詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的使用來源存取身分限制 Amazon S3 內容的存取。

下列政策使用 OAI 的 ID 作為政策的 `Principal`。如需使用 S3 儲存貯體政策授與 CloudFront OAI 存取權的詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的從原始存取身分 (OAI) 遷移到原始存取控制 (OAC)。

若要使用此範例：

- 將 *EH1HDMB1FH2TC* 取代為 OAI 的 ID。若要尋找 OAI 的 ID，請參閱 CloudFront 主控台上的「來源存取身分識別」頁面，或在 CloudFront API ListCloudFrontOriginAccessIdentities中使用。
- 用您的儲存貯體名稱取代 *example-s3-bucket*。

```
{
```

```
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity EH1HDMB1FH2TC"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::example-s3-bucket/*"
        }
    ]
}
```

管理對 Amazon S3 Storage Lens 的存取

授予 Amazon S3 Storage Lens 的許可

S3 Storage Lens 會彙總您的指標，並在 Amazon S3 主控台 Buckets (儲存貯體) 頁面上的 Account snapshot (帳戶快照) 區段中顯示資訊。S3 Storage Lens 也會提供互動式儀表板，您可以用來以視覺化方式呈現洞見與趨勢、標記異常值，以及接收最佳化儲存成本和套用資料保護最佳實務的建議。您的儀表板具有向下切入選項可產生及視覺化組織、帳戶、 AWS 區域、儲存類別、儲存貯體、字首或 Storage Lens 群組層級的洞察。您也可以將 CSV 或 Parquet 格式的每日指標匯出傳送到 S3 儲存貯體。

S3 Storage Lens 可將您彙總的儲存體用量指標匯出到 Amazon S3 儲存貯體，以供進一步分析。S3 Storage Lens 存放其指標匯出的儲存貯體稱為「目的地儲存貯體」。設定 S3 Storage Lens 指標匯出時，您必須具有目的地儲存貯體的儲存貯體政策。如需詳細資訊，請參閱 使用 Amazon S3 Storage Lens 評估儲存活動和使用量。

下列範例儲存貯體政策授予 Amazon S3 許可，將物件 (PUT 請求) 寫入至目的地儲存貯體。設定 S3 Storage Lens 指標匯出時，您可以對目的地儲存貯體使用類似這樣的儲存貯體政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3StorageLensExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "storage-lens.s3.amazonaws.com"
```

```
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::destination-bucket/destination-prefix/
StorageLens/111122223333/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": "111122223333",
                    "aws:SourceArn": "arn:aws:s3:region-code:111122223333:storage-
lens/storage-lens-dashboard-configuration-id"
                }
            }
        }
    ]
}
```

當您設定 S3 Storage Lens 組織層級指標匯出時，對上一個儲存貯體政策的 `Resource` 陳述式使用下列修改。

```
"Resource": "arn:aws:s3:::destination-bucket/destination-prefix/StorageLens/your-
organization-id/*",
```

管理 S3 庫存、S3 分析和 S3 庫存報告的許可

授予 S3 清查與 S3 分析的許可

S3 清查會建立儲存貯體中的物件清單，而 S3 分析儲存體類別分析匯出會建立分析中所使用資料的輸出檔案。庫存列出其物件的儲存貯體稱為來源儲存貯體。庫存檔案和分析匯出檔案撰寫至其中的儲存貯體稱為「目的地儲存貯體」。設定庫存或分析匯出時，您必須針對目的地儲存貯體建立儲存貯體政策。如需詳細資訊，請參閱 Amazon S3 清查 及 Amazon S3 分析 – 儲存類別分析。

下列儲存貯體政策範例授予 Amazon S3 許可，允許從來源儲存貯體的帳戶將物件 (PUT 請求) 寫入至目的地儲存貯體。設定 S3 清查與 S3 分析匯出時，您可以在目標儲存貯體上使用這類的儲存貯體政策。

```
{
      "Version": "2012-10-17",
```

```
      "Statement": [
        {
            "Sid": "InventoryAndAnalyticsExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-DESTINATION-BUCKET/*"
            ],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET"
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

控制 S3 庫存報告組態建立

Amazon S3 清查 會建立 S3 儲存貯體中物件的清單，以及各物件的中繼資料。此s3:PutInventoryConfiguration權限可讓使用者建立詳細目錄組態，其中包含預設可用的所有物件中繼資料欄位，並指定用於儲存詳細目錄的目的地值區。對目的地儲存貯體中的物件具有讀取權限的使用者可以存取庫存報告中所有可用的物件中繼資料欄位。如需 S3 庫存中可用的中繼資料欄位的詳細資訊，請參閱 Amazon S3 清查清單。

若要限制使用者設定 S3 庫存報告，請移除該使用者的s3:PutInventoryConfiguration權限。

S3 庫存報告組態中的某些物件中繼資料欄位是選用的，這表示預設可使用，但是當您授與使用者s3:PutInventoryConfiguration權限時，這些欄位可能會受到限制。您可以使用s3:InventoryAccessibleOptionalFields條件索引鍵，控制使用者是否可以在報表中包含這些選擇性的中繼資料欄位。如需 S3 庫存中可用的選用中繼資料欄位清單，請參閱 Amazon 簡單儲存服務 API 參考OptionalFields中的。

若要授與使用者使用特定選擇性中繼資料欄位建立詳細目錄組態的權限，請使用s3:InventoryAccessibleOptionalFields條件索引鍵來調整值區政策中的條件。

下列範例原則會授與 user (*Ana*) 有條件地建立詳細目錄組態的權限。原則中的`ForAllValues:StringEquals`條件會使用`s3:InventoryAccessibleOptionalFields`條件索引鍵來指定兩個允許的選擇性中繼資料欄位，亦即`Size`和`StorageClass`。因此，在建立詳細目錄組態時*Ana*，她可以包含的唯一選擇性中繼資料欄位是`Size`和`StorageClass`。

```
{
 "Id": "InventoryConfigPolicy",
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "AllowInventoryCreationConditionally",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action":
    "s3:PutInventoryConfiguration",
   "Resource":
    "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
   "Condition": {
    "ForAllValues:StringEquals": {
     "s3:InventoryAccessibleOptionalFields": [
        "Size",
        "StorageClass"
        ]
      }
    }
   }
 ]
}
```

若要限制使用者設定包含特定選用中繼資料欄位的 S3 庫存報告，請在來源儲存貯體的儲存貯體政策中新增`Deny`明確的陳述式。下列範例值區政策會拒絕使用者*Ana*在包含選用`ObjectAccessControlList`或中`ObjectOwner`繼資料欄位的來源值區*DOC-EXAMPLE-SOURCE-BUCKET*中建立詳細目錄組態。使用者仍然*Ana*可以使用其他選擇性中繼資料欄位建立庫存組態。

```
{
 "Id": "InventoryConfigSomeFields",
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "AllowInventoryCreation",
   "Effect": "Allow",
   "Principal": {
```

```
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action": "s3:PutInventoryConfiguration",
   "Resource":
    "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",

  },
  {
   "Sid": "DenyCertainInventoryFieldCreation",
   "Effect": "Deny",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action": "s3:PutInventoryConfiguration",
   "Resource":
     "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
   "Condition": {
    "ForAnyValue:StringEquals": {
     "s3:InventoryAccessibleOptionalFields": [
        "ObjectOwner",
        "ObjectAccessControlList"
        ]
      }
    }
   }
 ]
}
```

> **Note**
>
> 儲存貯體政策中使用s3:InventoryAccessibleOptionalFields條件金鑰不會影響以現有庫存組態為基礎的庫存報告傳遞。

> **Important**
>
> 我們建議您使ForAllValues用有Allow效果或ForAnyValue有Deny效果，如前面的範例所示。
>
> 請勿有Deny效或ForAllValuesForAnyValue有效地使用，因為這些組合可能會受到過度限制，並阻止庫存組態刪除。Allow

若要進一步了解ForAllValues和ForAnyValue條件集運算子，請參閱《IAM 使用者指南》中的多重值內容金鑰。

需要 MFA

Amazon S3 支援 MFA 保護的 API 存取，在存取 Amazon S3 資源時，此功能可以強制執行 Multi-Factor Authentication (MFA)。多重要素驗證提供您可套用至 AWS 環境的額外安全性層級。MFA 是一種安全功能，需要使用者提供有效的 MFA 代碼來證明 MFA 裝置的實體擁有。如需詳細資訊，請參閱 AWS 多重要素驗證。只要是請求存取 Amazon S3 資源，您都可以要求 MFA。

若要強制執行 MFA 要求，請在儲存貯體政策中使用 aws:MultiFactorAuthAge 條件金鑰。IAM 使用者可以使用 AWS Security Token Service (AWS STS) 核發的臨時登入資料存取 Amazon S3 資源。您可以在 AWS STS 請求時提供 MFA 代碼。

當 Amazon S3 收到實施多重要素驗證的請求時，aws:MultiFactorAuthAge 條件金鑰會提供一個數值，指出暫時憑證是在多久之前建立 (以秒為單位)。如果要求中所提供的暫時性憑證不是使用 MFA 裝置所建立，則此金鑰值為 null (不存在)。在儲存貯體政策中，您可以新增可檢查此值的條件，如下列範例所示。

如果請求並未使用 MFA 進行驗證，此範例政策會拒絕對 *example-s3-bucket* 儲存貯體中的 */taxdocuments* 資料夾進行任何 Amazon S3 操作。若要進一步了解 MFA，請參閱《IAM 使用者指南》中的在 AWS中使用多重要素驗證 (MFA)。

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::example-s3-bucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true }}
      }
    ]
 }
```

如果 aws:MultiFactorAuthAge 條件金鑰值為 null，則 Condition 區塊中的 Null 條件會評估為 true，表示已在沒有 MFA 裝置的情況下建立請求中的暫時性安全憑證。

下列儲存貯體政策是先前儲存貯體政策的延伸。下列值區政策包含兩個政策陳述式。一個陳述式會允許任何人在儲存貯體 (*example-s3-bucket*) 上的 s3:GetObject 許可。另一個陳述式則會透過要求 MFA，進一步限制對儲存貯體中 *example-s3-bucket/taxdocuments* 資料夾的存取。

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::example-s3-bucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true } }
      },
      {
        "Sid": "",
        "Effect": "Allow",
        "Principal": "*",
        "Action": ["s3:GetObject"],
        "Resource": "arn:aws:s3:::example-s3-bucket/*"
      }
    ]
 }
```

您可以選擇性地使用數值條件，來限制 aws:MultiFactorAuthAge 金鑰有效的持續時間。您使用 aws:MultiFactorAuthAge 金鑰指定的持續時間，與驗證請求時所使用之臨時安全憑證的生命週期無關。

例如，除了需要 MFA 身分驗證之外，下列儲存貯體政策也會檢查在多久之前建立暫時性工作階段。如果 aws:MultiFactorAuthAge 金鑰值指出已在一小時 (3,600 秒) 之前建立暫時性工作階段，則此政策會拒絕任何操作。

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
```

```
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::example-s3-bucket/taxdocuments/*",
        "Condition": {"Null": {"aws:MultiFactorAuthAge": true }}
      },
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::example-s3-bucket/taxdocuments/*",
        "Condition": {"NumericGreaterThan": {"aws:MultiFactorAuthAge": 3600 }}
       },
       {
         "Sid": "",
         "Effect": "Allow",
         "Principal": "*",
         "Action": ["s3:GetObject"],
         "Resource": "arn:aws:s3:::example-s3-bucket/*"
       }
    ]
 }
```

防止使用者刪除物件

這些使用者預設不具備任何許可。但是當您建立原則時，您可能會授與使用者您不想授與的權限。若要避免此類權限漏洞，您可以透過新增明確拒絕來撰寫更嚴格的存取原則。

若要明確封鎖使用者或帳戶刪除物件，您必須將下列動作新增至值區政策：`s3:DeleteObjects`、`s3:DeleteObjectVersion`、和`s3:PutLifecycleConfiguration`權限。這三個動作都是必要的，因為您可以透過明確呼叫 DELETE Object API 或設定物件的生命週期來刪除物件 (請參閱管理儲存生命週期)，以便 Amazon S3 可以在物件的存留期到期時移除物件。

在下列原則範例中，您明確拒絕使用者 Dave 的 DELETE 物件權限。明確拒絕一律會取代授與的任何其他權限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/Dave"
```

```
      },
      "Action": [
        "s3:GetObjectVersion",
        "s3:GetBucketAcl"
      ],
      "Resource": [
        "arn:aws:s3:::example-s3-bucket1",
   "arn:aws:s3:::example-s3-bucket1/*"
      ]
    },
    {
      "Sid": "statement2",
      "Effect": "Deny",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/Dave"
      },
      "Action": [
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutLifecycleConfiguration"
      ],
      "Resource": [
        "arn:aws:s3:::example-s3-bucket1",
     "arn:aws:s3:::example-s3-bucket1/*"
      ]
    }
  ]
}
```

使用條件鍵值區政策範例

您可以使用存取政策語言來指定授予許可時的條件。您可以使用選用 Condition 元素或 Condition 區塊來指定政策何時生效的條件。

如需使用 Amazon S3 條件索引鍵進行物件和儲存貯體操作的政策，請參閱下列範例。如需條件索引鍵的詳細資訊，請參閱 Amazon S3 的政策條件金鑰。如需可在政策中指定的 Amazon S3 動作、條件金鑰和資源的完整清單，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

範例 — 物件操作的 Amazon S3 條件索引鍵

本節提供的範例示範如何將 Amazon S3 特定的條件索引鍵用於物件操作。如需可在政策中指定的 Amazon S3 動作、條件金鑰和資源的完整清單，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

有幾個政策範例示範如何搭配 PUT 物件操作來使用條件金鑰。PUT 物件操作允許存取控制清單 (ACL) 特定的標頭，可用來授予以 ACL 為基礎的許可。使用這些金鑰，儲存貯體擁有者可設定條件，在使用者上傳物件時要求特定的存取許可。您也可以授與以 ACL 為基礎的權限與作業。PutObjectAcl 如需詳細資訊，請參閱 Amazon S3 Amazon 簡單儲存服務 API 參考PutObjectAcl中的。如需 ACL 的詳細資訊，請參閱「存取控制清單 (ACL) 概觀」。

主題

- 範例 1：授與 s3：需要使用伺服器端加密存放物件的PutObject 權限
- 示例 2：授予 s3：複製對象的PutObject 權限，並限制複製源
- 範例 3：授與物件特定版本的存取權
- 範例 4：根據物件標籤授與權限
- 範例 5：依值區擁有者的 AWS 帳戶 ID 限制存取
- 範例 6：需要最低 TLS 版本

範例 1：授與 s3：需要使用伺服器端加密存放物件的PutObject 權限

假設帳戶 A 擁有儲存貯體。帳戶管理員想要將上傳物件的許可授予帳戶 A 中的使用者 Jane，但條件是 Jane 一律要求伺服器端加密，以便 Amazon S3 儲存加密的物件。帳戶 A 管理員可使用 `s3:x-amz-server-side-encryption` 條件索引鍵完成此操作，如下所示。`Condition` 區塊中的金鑰/值對會指定 `s3:x-amz-server-side-encryption` 金鑰。

```
"Condition": {
     "StringNotEquals": {
         "s3:x-amz-server-side-encryption": "AES256"
     }}
```

使用測試權限時 AWS CLI，您必須使用參數新增必要的`--server-side-encryption`參數。

```
aws s3api put-object --bucket example1bucket --key HappyFace.jpg --body c:
\HappyFace.jpg --server-side-encryption "AES256" --profile AccountBadmin
```

示例 2：授予 s3：複製對象的PutObject 權限，並限制複製源

在 PUT 物件請求中，當您指定來源物件時，即為複製操作 (請參閱 PUT 物件 - 複製)。相對的，儲存貯體擁有者可以授予使用者複製物件的許可，但來源有所限制，例如：

- 僅允許複製來自 `sourcebucket` 儲存貯體的物件。

- 允許從來源儲存貯體複製物件，而且只複製金鑰名稱前綴以 public/f 開頭的物件 (例如，sourcebucket/public/*)。
- 允許只複製來源儲存貯體中的特定物件 (例如 sourcebucket/example.jpg)。

以下儲存貯體政策會授予使用者 (Dave) s3:PutObject 許可。這允許其複製僅具有以下條件的物件：要求包含 s3:x-amz-copy-source 標頭，而且標頭值指定 /awsexamplebucket1/public/* 金鑰名稱前綴。

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
            "Sid": "cross-account permission to user in your own account",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*"
        },
        {
            "Sid": "Deny your user permission to upload object if copy source is not /
bucket/folder",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
                "StringNotLike": {
                    "s3:x-amz-copy-source": "awsexamplebucket1/public/*"
                }
            }
        }
    ]
}
```

使用 AWS CLI

您可以使用 AWS CLI copy-object指令測試權限。您可以透過新增 --copy-source 參數來指定來源，而金鑰名稱前綴必須符合政策中允許的前綴。您需要使用 --profile 參數，來提供使用者 Dave 的憑證。若要取得有關設定的更多資訊 AWS CLI，請參閱使用 AWS CLI 來透過 Amazon S3 進行開發。

```
aws s3api copy-object --bucket awsexamplebucket1 --key HappyFace.jpg
--copy-source examplebucket/public/PublicHappyFace1.jpg --profile AccountADave
```

提供僅複製特定物件的許可

前項政策使用 StringNotLike 條件。若要授予僅複製特定物件的許可，您必須將條件從 StringNotLike 變更成 StringNotEquals，然後指定確切的物件金鑰，如下所示。

```
"Condition": {
       "StringNotEquals": {
           "s3:x-amz-copy-source": "awsexamplebucket1/public/PublicHappyFace1.jpg"
       }
}
```

範例 3：授與物件特定版本的存取權

假設帳戶 A 擁有已啟用版本控制的儲存貯體。儲存貯體有數個版本的 HappyFace.jpg 物件。現在，帳戶管理員希望授予使用者 Dave 只取得特定物件版本的許可。帳戶管理員可有條件地授予 Dave s3:GetObjectVersion 許可來完成此操作，如下所示。Condition 區塊中的金鑰/值對會指定 s3:VersionId 條件索引鍵。在本例中，Dave 需要知道確切的物件版本 ID 才能擷取物件。

如需詳細資訊，請參閱 Amazon 簡易儲存服務 API 參考GetObject中的。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::examplebucketversionenabled/HappyFace.jpg"
        },
        {
```

```
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::examplebucketversionenabled/HappyFace.jpg",
            "Condition": {
                "StringNotEquals": {
                    "s3:VersionId": "AaaHbAQitwiL_h47_44lRO2DDfLlBO5e"
                }
            }
        }
    ]
}
```

使用 AWS CLI

您可以使用 AWS CLI `get-object`指令與識別特定物件版本的`--version-id`參數來測試權限。此命令會擷取物件並將它儲存到 `OutputFile.jpg` 檔案。

```
aws s3api get-object --bucket examplebucketversionenabled --key HappyFace.jpg
 OutputFile.jpg --version-id AaaHbAQitwiL_h47_44lRO2DDfLlBO5e --profile AccountADave
```

範例 4：根據物件標籤授與權限

如需有關如何搭配 Amazon S3 操作使用物件標記條件索引鍵的範例，請參閱標記與存取控制政策。

範例 5：依值區擁有者的 AWS 帳戶 ID 限制存取

您可以使用 `aws:ResourceAccount` 或 `s3:ResourceAccount` 索引鍵來撰寫 IAM 或虛擬私有雲端 (VPC) 端點政策，以限制使用者、角色或應用程式對特定 AWS 帳戶 ID 擁有的 Amazon S3 儲存貯體的存取權。您可以使用此條件索引鍵來限制 VPC 內的用戶端存取您未擁有的儲存貯體。

不過，請注意，某些 AWS 服務需要存取 AWS 受管理值區。因此，在 IAM 政策使用 `aws:ResourceAccount` 或者 `s3:ResourceAccount` 金鑰也可能影響對這些資源的存取。

如需詳細資訊和範例，請參閱下列資源：

- 《AWS PrivateLink 指南》中限制對指定 AWS 帳戶帳戶中儲存貯體的存取
- Amazon ECR 指南中的限制對 Amazon ECR 使用的儲存貯體的存取

- 在AWS Systems Manager 指南中為 AWS 受管 Amazon S3 儲存貯體提供必要的系統管理員存取權
- AWS 儲存部落格中限制對特定 AWS 帳戶帳戶所擁有 Amazon S3 儲存貯體 的存取

範例 6：需要最低 TLS 版本

您可以使用 s3: TlsVersion 條件金鑰撰寫 IAM、虛擬私有雲端端點 (VPCE) 或儲存貯體政策，以根據用戶端使用的 TLS 版本限制使用者或應用程式存取 Amazon S3 儲存貯體。您可以使用此條件索引鍵來撰寫需要最低 TLS 版本的政策。

Example

此範例值區政策會 PutObject 拒絕 TLS 版本低於 1.2 的用戶端 (例如 1.1 或 1.0) 的要求。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::example-s3-bucket1",
                "arn:aws:s3:::example-s3-bucket1/*"
            ],
            "Condition": {
                "NumericLessThan": {
                    "s3:TlsVersion": 1.2
                }
            }
        }
    ]
}
```

Example

此範例值區政策允許 TLS 版本高於 1.1 的用戶端發出 PutObject 要求，例如 1.2、1.3 或更高版本。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::example-s3-bucket1",
                "arn:aws:s3:::example-s3-bucket1/*"
            ],
            "Condition": {
                "NumericGreaterThan": {
                    "s3:TlsVersion": 1.1
                }
            }
        }
    ]
}
```

範例 — 儲存貯體操作的 Amazon S3 條件索引鍵

本節提供的政策範例示範如何將 Amazon S3 特定的條件索引鍵用於儲存貯體操作。

主題

- 示例 1：授予 s3：具有 IP 地址條件的GetObject 權限
- 範例 2：取得儲存貯體中包含特定前綴的物件清單
- 範例 3：設定金鑰數上限

示例 1：授予 s3：具有 IP 地址條件的GetObject 權限

如果要求來自特定 IP 位址範圍 (192.0.2.*)，您可以授與已驗證的使用者使用該`s3:GetObject`動作的權限，除非 IP 位址是 192.0.2.188。在條件區塊中，`IpAddress` 與 `NotIpAddress` 是條件，而且每個條件都獲得一組用於評估的金鑰/值對。本範例中的兩個鍵值對都使用了`aws:SourceIp` AWS寬鍵。

> **Note**
>
> 條件中指定的 `IPAddress` 與 `NotIpAddress` 金鑰值使用 CIDR 表示法，如 RFC 4632 中所述。如需詳細資訊，請參閱 http://www.rfc-editor.org/rfc/rfc4632.txt。

```
{
    "Version": "2012-10-17",
    "Id": "S3PolicyId1",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": "*",
            "Action":"s3:GetObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition" : {
                "IpAddress" : {
                    "aws:SourceIp": "192.0.2.0/24"
                },
                "NotIpAddress" : {
                    "aws:SourceIp": "192.0.2.188/32"
                }
            }
        }
    ]
}
```

您也可以在 Amazon S3 政策中使用其他 AWS全狀況金鑰。例如，您可以在 VPC 端點的儲存貯體政策中指定 `aws:SourceVpce` 和 `aws:SourceVpc` 條件索引鍵。如需特定範例，請參閱 使用儲存貯體政策控制來自 VPC 端點的存取。

> **Note**
>
> 對於某些 AWS 全域條件索引鍵，僅支援某些資源類型。因此，請檢查 Amazon S3 是否支援您要使用的全域條件金鑰和資源類型，或者您是否需要改用 Amazon S3 特定的條件金鑰。如需 Amazon S3 支援資源類型和條件金鑰的完整清單，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

範例 2：取得儲存貯體中包含特定前綴的物件清單

您可以使用`s3:prefix`條件鍵將 GET 值區 (ListObjects) API 的回應限制為具有特定前置詞的金鑰名稱。如果您是儲存貯體擁有者，則可限制使用者列出儲存貯體中特定前綴的內容。如果儲存貯體中的物件是按金鑰名稱前綴來組織，此條件索引鍵會很有用。Amazon S3 主控台會使用金鑰名稱前綴來顯示資料夾概念。只有主控台支援資料夾的概念，Amazon S3 API 僅支援儲存貯體和物件。如需有關使用字首和分隔符號來篩選存取許可的詳細資訊，請參閱使用使用者政策來控制對儲存貯體的存取。

例如，若您擁有的兩個物件金鑰名稱分別為 public/object1.jpg 和 public/object2.jpg，則主控台會在 public 資料夾下顯示物件。在 Amazon S3 API 中，這些是具有字首的物件，而不是資料夾內的物件。不過，在 Amazon S3 API 中，如果您使用這類字首來組織物件金鑰，則可以搭配 s3:prefix 條件來授予 s3:ListBucket 許可，以允許使用者取得具有這些特定字首的金鑰名稱清單。

在此範例中，儲存貯體擁有者及使用者所屬的父帳戶是同一個。所以儲存貯體擁有者可以使用儲存貯體政策或使用者政策。如需可搭配 GET 值區 (ListObjects) API 使用之其他條件金鑰的詳細資訊，請參閱ListObjects。

使用者政策

下列使用者政策授予 s3:ListBucket 許可 (請參閱 GET 儲存貯體 (列出物件))，條件是使用者需要在請求中指定 prefix 的值為 projects。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:ListBucket",
         "Resource":"arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "projects"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringNotEquals" : {
                 "s3:prefix": "projects"
             }
          }
       }
    ]
}
```

此條件限制使用者列出使用 projects 字首的物件金鑰。無論使用者可能擁有什麼樣的許可，新增的明確拒絕都會拒絕使用者列出具有任何其他字首金鑰的要求。例如，只要更新上述使用者政策或透過儲存貯體政策，使用者就有可能取得列出物件金鑰的許可，而不受任何限制。明確拒絕會取代一切，所以使用者列出 projects 前綴以外之金鑰的要求將遭拒。

儲存貯體政策

如果將 Principal 元素新增至上述使用者政策來找出使用者，則您現在就會有儲存貯體政策，如下所示。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/bucket-owner"
         },
         "Action":  "s3:ListBucket",
         "Resource": "arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "projects"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/bucket-owner"
         },
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringNotEquals" : {
                 "s3:prefix": "projects"
             }
          }
       }
    ]
}
```

使用 AWS CLI

您可以使用下列`list-object` AWS CLI 命令來測試原則。在命令中，您要使用 `--profile` 參數來提供使用者憑證。若要取得有關設定和使用的更多資訊 AWS CLI，請參閱使用 AWS CLI 來透過 Amazon S3 進行開發。

```
aws s3api list-objects --bucket awsexamplebucket1 --prefix examplefolder --profile
 AccountADave
```

如果儲存貯體已啟用版本控制，您必須在上述政策中授予 `s3:ListBucketVersions` 許可 (而非 `s3:ListBucket` 許可)，才能列出儲存貯體中的物件。此許可也支援 `s3:prefix` 條件索引鍵。

範例 3：設定金鑰數上限

您可以使用`s3:max-keys`條件鍵來設置請求者可以在 GET 存儲桶（ListObjects）或ListObjectVersions請求中返回的最大密鑰數量。API 預設傳回最多 1,000 個金鑰。如需您可搭配 `s3:max-keys` 使用的數值條件運算子清單和隨附範例，請參閱《IAM 使用者指南》中的數值條件運算子。

Amazon S3 的基於身份的政策

依預設，使用者和角色沒有建立或修改 Amazon S3 資源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行工作。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

如需 Amazon S3 定義的動作和資源類型的詳細資訊，包括每種資源類型的 ARN 格式，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

主題

- 政策最佳實務
- Amazon S3 的基於身分識別的政策範例
- 使用使用者政策來控制對儲存貯體的存取

政策最佳實務

以身分識別為基礎的政策決定某人是否可以在您的帳戶中建立、存取或刪除 Amazon S3 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

Amazon S3 的基於身分識別的政策範例

本節顯示用於控制 Amazon S3 存取的數個以身分識別為基礎的政策範例 AWS Identity and Access Management (IAM)。如需值區政策 (以資源為基礎的政策) 範例，請參閱Amazon S3 的存儲桶政策。如需有關 IAM 政策語言的資訊，請參閱 Amazon S3 中的政策和許可。

如果您是以程式設計方式來使用，則適用以下範例政策。但是，若要透過 Amazon S3 主控台來使用，您必須授予主控台所需的其他許可。如需有關搭配 Amazon S3 主控台使用這些政策的資訊，請參閱使用使用者政策來控制對儲存貯體的存取。

主題

- 允許 IAM 使用者存取其中一個儲存貯體
- 允許每個 IAM 使用者存取儲存貯體中的資料夾
- 允許群組在 Amazon S3 中有共用資料夾
- 允許您的所有使用者讀取該儲存貯體某一部分中的物件
- 允許合作夥伴將檔案放置在該儲存貯體的特定部分中
- 限制對特定 AWS 帳戶中 Amazon S3 儲存貯體的存取
- 限制對組織單位內 Amazon S3 儲存貯體的存取
- 限制組織內 Amazon S3 儲存貯體的存取權
- 授與擷取PublicAccessBlock組態的權限 AWS 帳戶
- 將儲存貯體建立限制為一個區域

允許 IAM 使用者存取其中一個儲存貯體

在此範例中，您想要授與 IAM 使用者 AWS 帳戶 存取其中一個值區 (*## s3-bucket1*)，並允許使用者新增、更新和刪除物件。

除了授予使用者 s3:PutObject、s3:GetObject 與 s3:DeleteObject 許可之外，政策也會授予 s3:ListAllMyBuckets、s3:GetBucketLocation 與 s3:ListBucket 許可。這些是主控台需要的額外許可。還需要 s3:PutObjectAcl 與 s3:GetObjectAcl 動作才能在主控台中複製、剪下與貼上物件。如需授予使用者許可並使用主控台測試的演練範例，請參閱「使用使用者政策來控制對儲存貯體的存取」。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action": "s3:ListAllMyBuckets",
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":["s3:ListBucket","s3:GetBucketLocation"],
         "Resource":"arn:aws:s3:::example-s3-bucket1"
      },
      {
         "Effect":"Allow",
         "Action":[
```

```
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:GetObject",
            "s3:GetObjectAcl",
            "s3:DeleteObject"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/*"
      }
   ]
}
```

允許每個 IAM 使用者存取儲存貯體中的資料夾

在此範例中，您希望兩個 IAM 使用者 Mary 和 Carlos 能夠存取您的值區，*## s3-bucket1*，以便他們可以新增、更新和刪除物件。不過，您希望限制每位使用者只能存取儲存貯體中的單一字首 (資料夾)。您可以建立名稱與使用者名稱相符的資料夾。

```
example-s3-bucket1
   Mary/
   Carlos/
```

若要授與每位使用者只能存取自己資料夾的存取權，您可以為每位使用者撰寫一項政策，並分別予以連接。例如，您可以將下列政策連接至使用者 Mary，讓她擁有 *example-s3-bucket1*/*Mary* 資料夾的特定 Amazon S3 許可。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/Mary/*"
      }
   ]
}
```

然後，您可以將類似的政策附加至使用者 Carlos，並在 Resource 值中指定 *Carlos*。

和將政策連接至個別使用者不同，您可撰寫使用政策變數的單一政策，然後將政策連接至群組。首先，您必須建立群組，並將 Mary 及 Mary 新增至此群組。以下政策範例在 *example-s3-bucket1*/${aws:username} 資料夾中允許一組 Amazon S3 許可。評估原則時，原則變數會${aws:username}由要求者的使用者名稱取代。例如，如果 Mary 傳送放置物件的要求，只有當 Mary 將物件上傳至 *example-s3-bucket1*/Mary 資料夾時，才允許此操作。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/${aws:username}/*"
      }
   ]
}
```

> **Note**
>
> 使用政策變數時，您必須在政策中明確指定版本 2012-10-17。IAM 政策語言的預設版本 2008-10-17 不支援政策變數。

如果您希望在 Amazon S3 主控台上測試前一項政策，主控台需要額外的許可，如以下政策中所示。如需主控台如何使用這些許可的資訊，請參閱「使用使用者政策來控制對儲存貯體的存取」。

```
{
 "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGroupToSeeBucketListInTheConsole",
      "Action": [
       "s3:ListAllMyBuckets",
```

```
        "s3:GetBucketLocation"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Sid": "AllowRootLevelListingOfTheBucket",
      "Action": "s3:ListBucket",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::example-s3-bucket1",
      "Condition":{
            "StringEquals":{
                   "s3:prefix":[""], "s3:delimiter":["/"]
                         }
               }
    },
    {
      "Sid": "AllowListBucketOfASpecificUserPrefix",
      "Action": "s3:ListBucket",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::example-s3-bucket1",
      "Condition":{  "StringLike":{"s3:prefix":["${aws:username}/*"] }
       }
    },
      {
     "Sid": "AllowUserSpecificActionsOnlyInTheSpecificUserPrefix",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/${aws:username}/*"
      }
  ]
}
```

> **Note**
>
> 在 2012-10-17 版本的政策中，政策變數是以 $ 開頭。如果您的物件金鑰 (物件名稱) 包含 $，這項語法變更可能會造成衝突。

> 若要避免此衝突，請使用 ${$} 指定 $ 字元。例如，若要在政策中包含物件金鑰 my$file，請將它指定為 my${$}file。

雖然 IAM 使用者名稱是方便人類閱讀的識別符，但它們不一定要是全域唯一的。例如，如果使用者 Carlos 離職但有另一位 Carlos 入職，則新 Carlos 可以存取舊 Carlos 的資訊。

您可以根據 IAM 使用者 ID 建立資料夾，而不是使用使用者名稱。每個 IAM 使用者 ID 都是唯一的。在本例中，您必須修改前一項政策，以使用 ${aws:userid} 政策變數。如需使用者識別符的詳細資訊，請參閱《IAM 使用者指南》中的 IAM 識別符。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/home/${aws:userid}/*"
      }
   ]
}
```

允許非 IAM 使用者 (行動應用程式使用者) 存取儲存貯體中的資料夾

假設您希望開發行動應用程式，一種將使用者資料存放在 S3 儲存貯體的遊戲。您希望在您的儲存貯體中，為每一位應用程式使用者建立一個資料夾。您還想限制每個用戶對自己的文件夾的訪問權限。 但是您無法在有人下載您的應用程式並開始玩遊戲之前建立資料夾，因為您沒有他們的使用者 ID。

在本例中，您可要求使用者使用 Login with Amazon、Facebook 或 Google 等公用身分提供者，登入您的應用程式。在使用者透過這些提供者之一登入您的應用程式後，他們就會有一個使用者 ID，您可用來在執行階段建立使用者專用資料夾。

然後，您可以使用聯合網頁身分識別， AWS Security Token Service 將身分識別提供者的資訊與您的應用程式整合，並取得每個使用者的臨時安全性憑證。然後，您就可以建立 IAM 政策，允許應用程式

存取您的儲存貯體並執行操作，例如建立使用者專用資料夾和上傳資料。如需 Web 聯合身分的詳細資訊，請參閱 IAM 使用者指南中的關於 Web 聯合身分。

允許群組在 Amazon S3 中有共用資料夾

如果將下列政策連接至群組，則會授予群組中的所有人存取 Amazon S3 的下列資料夾：*example-s3-bucket1*/share/marketing。群組成員只能存取原則所示的特定 Amazon S3 許可，而且僅限於特定資料夾中的物件。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/share/marketing/*"
      }
   ]
}
```

允許您的所有使用者讀取該儲存貯體某一部分中的物件

在此範例中，您建立名為 *AllUsers* 的群組，其中包含 AWS 帳戶擁有的所有 IAM 使用者。然後您要連接政策，允許群組存取 GetObject 與 GetObjectVersion，但只適用於 *example-s3-bucket1*/readonly 資料夾中的物件。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket1/readonly/*"
```

```
      }
   ]
}
```

允許合作夥伴將檔案放置在該儲存貯體的特定部分中

在本例中，您要建立名為 *AnyCompany* 的群組，代表合作夥伴公司。您為合作夥伴公司中需要存取權的特定人員或應用程式建立 IAM 使用者，然後將此使用者放入群組中。

然後您要連接政策，讓群組 PutObject 可存取儲存貯體中的下列資料夾：

example-s3-bucket1/uploads/anycompany

您不希望 *AnyCompany* 群組對儲存貯體執行任何其他操作，所以您新增陳述式明確拒絕任何 Amazon S3 動作的許可，但對 AWS 帳戶中的任何 Amazon S3 資源執行 PutObject 除外。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::example-s3-bucket1/uploads/anycompany/*"
      },
      {
         "Effect":"Deny",
         "Action":"s3:*",
         "NotResource":"arn:aws:s3:::example-s3-bucket1/uploads/anycompany/*"
      }
   ]
}
```

限制對特定 AWS 帳戶中 Amazon S3 儲存貯體的存取

如果您想要確保 Amazon S3 主體只存取受信任內部的資源 AWS 帳戶，您可以限制存取。例如，此以身分為基礎的 IAM 政策使用 Deny 效用封鎖對 Amazon S3 動作的存取權，除非正在存取的 Amazon S3 資源是在帳戶 *222222222222* 中。若要防止中的 IAM 主體存 AWS 帳戶 取帳戶以外的 Amazon S3 物件，請附加下列 IAM 政策：

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

```
      "Sid": "DenyS3AccessOutsideMyBoundary",
      "Effect": "Deny",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "222222222222"
          ]
        }
      }
    }
  ]
}
```

> **Note**
>
> 此政策不會取代您現有的 IAM 存取控制，因為它不會授予任何存取權。相反地，此政策會充當其他 IAM 許可的附加防護機制，無論透過其他 IAM 政策授與的許可為何。

務必使用您自己的 AWS 帳戶取代帳戶 ID *222222222222*。若要將政策套用至多個帳戶的同時仍維持此限制，請以 `aws:PrincipalAccount` 條件金鑰取代帳戶 ID。此條件要求主體和資源必須位於同一個帳户中。

限制對組織單位內 Amazon S3 儲存貯體的存取

如果您在中設定了組織單位 (OU) AWS Organizations，則可能想要將 Amazon S3 儲存貯體存取限制在組織的特定部分。在此範例中，我們將使用 `aws:ResourceOrgPaths` 金鑰限制 Amazon S3 儲存貯體只能存取組織中的 OU。在此範例中，OU ID 是 *ou-acroot-exampleou*。請務必使用您自己的 OU ID 來取代您自己政策中的這個值。

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Sid": "AllowS3AccessOutsideMyBoundary",
       "Effect": "Allow",
       "Action": [
         "s3:*"
```

```
        ],
        "Resource": "*",
        "Condition": {
          "ForAllValues:StringNotLike": {
            "aws:ResourceOrgPaths": [
              "o-acorg/r-acroot/ou-acroot-exampleou/"
            ]
          }
        }
      }
    ]
}
```

> **Note**
>
> 此原則不會授予任何存取權。相反地，此政策會充當其他 IAM 許可的逆止器，防止您的主體存取 OU 定義界限之外的 Amazon S3 物件。

該政策會拒絕存取 Amazon S3 動作，除非正在存取的 Amazon S3 物件位於組織中的 *ou-acroot-exampleou* OU 中。此 IAM 政策條件需要 `aws:ResourceOrgPaths` (多重值件金鑰)，以包含任何列出的 OU 路徑。此政策使用 `ForAllValues:StringNotLike` 運算子將 `aws:ResourceOrgPaths` 的值與列出的 OU 比較，不區分大小寫。

限制組織內 Amazon S3 儲存貯體的存取權

若要限制組織內 Amazon S3 物件的存取權，請將 IAM 政策附加至組織的根目錄，讓政策套用到您組織中的所有帳戶。如欲要求您的 IAM 主體遵循此規則，請使用服務控制政策 (SCP)。如果您選擇使用 SCP，請務必徹底測試 SCP，然後再將此政策附加到組織的根目錄。

在以下範例政策中，除非正在存取的 Amazon S3 物件與正在存取該物件的 IAM 主體位於同一個組織中，否則會拒絕存取 Amazon S3 動作：

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Sid": "DenyS3AccessOutsideMyBoundary",
       "Effect": "Deny",
       "Action": [
         "s3:*"
```

```
        ],
        "Resource": "arn:aws:s3:::*/*",
        "Condition": {
          "StringNotEquals": {
            "aws:ResourceOrgID": "${aws:PrincipalOrgID}"
          }
        }
      }
    ]
}
```

> **Note**
>
> 此原則不會授予任何存取權。相反地，此政策會充當其他 IAM 許可的逆止器，防止您的主體存取組織外部的任何 Amazon S3 物件。此政策也會套用到政策生效後建立的 Amazon S3 資源。

此範例中的 IAM 政策條件需要 `aws:ResourceOrgID` 和 `aws:PrincipalOrgID` 彼此相等。根據此需求，發出請求的主體與正在存取的資源必須位於同一個組織中。

授與擷取PublicAccessBlock組態的權限 AWS 帳戶

下列範例以身分識別為基礎的原則會將`s3:GetAccountPublicAccessBlock`權限授與使用者。您需要將這些許可的 `Resource` 值設成 `"*"`。如需有關資源 ARN 的資訊，請參閱Amazon S3 的政策資源。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action":[
            "s3:GetAccountPublicAccessBlock"
         ],
         "Resource":[
            "*"
         ]
      }
    ]
}
```

將儲存貯體建立限制為一個區域

假設 AWS 帳戶 系統管理員想要授與其使用者 (Dave) 權限，以便只在南美洲 (聖保羅) 區域建立值區。帳戶管理員可連接下列使用者政策，授予附條件的 `s3:CreateBucket` 許可，如下所示。`Condition` 區塊中的金鑰/值對會指定 `s3:LocationConstraint` 金鑰與 `sa-east-1` 區域當做值。

> **Note**
>
> 在此範例中，儲存貯體擁有者要將許可授予其使用者之一，所以可使用儲存貯體政策或使用者政策。本例示範使用者政策。

如需 Amazon S3 區域的清單，請參閱《AWS 一般參考》中的區域與端點。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:CreateBucket",
         "Resource": "arn:aws:s3:::*",
         "Condition": {
             "StringLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       }
    ]
}
```

新增明確拒絕

上述政策限制使用者只能在 `sa-east-1` 區域建立儲存貯體，任何其他區域皆不可。不過，可能有其他政策會授予此使用者許可，使其可在其他區域建立儲存貯體。例如，假設使用者屬於某個群組，而此群組所連接的政策允許群組許可範圍中的所有使用者在另一區域建立儲存貯體。為了確保用戶沒有獲得在任何其他區域中創建存儲桶的權限，您可以在上述策略中添加明確的拒絕聲明。

`Deny` 陳述式使用 `StringNotLike` 條件。亦即，如果位置限制條件不是 `sa-east-1`，便拒絕建立儲存貯體的要求。明確拒絕不允許用戶在任何其他區域中創建存儲桶，無論用戶獲得什麼其他權限。下列原則包含明確的拒絕陳述式。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:CreateBucket",
         "Resource": "arn:aws:s3:::*",
         "Condition": {
             "StringLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Action": "s3:CreateBucket",
         "Resource": "arn:aws:s3:::*",
         "Condition": {
             "StringNotLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       }
    ]
}
```

使用 AWS CLI

您可以使用下列create-bucket AWS CLI 命令來測試原則。本例使用 bucketconfig.txt 檔案指定位置限制條件。請注意Windows檔案路徑。您需要適時更新儲存貯體名稱及路徑。您必須使用 --profile 參數提供使用者憑證。若要取得有關設定和使用的更多資訊 AWS CLI，請參閱使用 AWS CLI 來透過 Amazon S3 進行開發。

```
aws s3api create-bucket --bucket examplebucket --profile AccountADave --create-bucket-configuration file://c:/Users/someUser/bucketconfig.txt
```

bucketconfig.txt 檔案會指定組態，如下所示。

```
{"LocationConstraint": "sa-east-1"}
```

使用使用者政策來控制對儲存貯體的存取

此演練說明使用者許可與 Amazon S3 搭配運作的方式。在此範例中，您建立含有資料夾的儲存貯體。然後，您可以在您的儲存貯體中建立 AWS Identity and Access Management IAM 使用者， AWS 帳戶並授予這些使用者對 Amazon S3 儲存貯體及其中資料夾的增量許可。

主題

儲存貯體與資料夾的基本概念

Amazon S3 資料模型是單層式結構：您建立儲存貯體，儲存貯體存放物件。子儲存貯體或子資料夾沒有階層，但您可以模擬資料夾階層。Amazon S3 主控台之類工具可以在儲存貯體中顯示這些邏輯資料夾和子資料夾。

此主控台會顯示名為 companybucket 的儲存貯體有三個資料夾：Private、Development 和 Finance 以及物件 s3-dg.pdf。此主控台使用物件名稱 (金鑰) 建立內含資料夾與子資料夾的邏輯階層。請考量下列範例：

- 當您建立 Development 資料夾時，主控台會使用 Development/ 索引鍵建立物件。請注意結尾斜線的 (/) 分隔符號。
- 在 Projects1.xls 資料夾上傳名為 Development 的物件，此主控台會上傳該物件並為該物件提供 Development/Projects1.xls 索引鍵。

在索引鍵中，Development 為字首，/ 為分隔符號。Amazon S3 API 允許在其操作中使用字首與分隔符號。例如，您可以取得儲存貯體中所有包含特定字首與分隔符號之物件的清單。在主控台中，當您開啟 Development 資料夾時，主控台便會列出該資料夾中的物件。在以下範例中，Development 資料夾包含一個物件。

主控台在 Development 儲存貯體中列出 companybucket 資料夾時會向 Amazon S3 傳送請求，並在請求中指定字首 Development 和分隔符號 /。此主控台的回應就像是電腦檔案系統中的資料夾清單。先前的範例顯示儲存貯體 companybucket 包含一個具有金鑰 Development/Projects1.xls 的物件。

主控台使用物件金鑰來推論邏輯階層。Amazon S3 沒有物理層次結構。Amazon S3 只有包含平面檔案結構中物件的儲存貯體。當您使用 Amazon S3 API 建立物件時，可以使用暗指邏輯階層的物件金鑰。如此演練所示範，您可以在建立物件的邏輯階層時，管理對個別資料夾的存取權。

在您開始前，確保您熟悉根層級儲存貯體內容的概念。假設您的 companybucket 儲存貯體包含下列物件：

- Private/privDoc1.txt
- Private/privDoc2.zip
- Development/project1.xls
- Development/project2.xls
- Finance/Tax2011/document1.pdf
- Finance/Tax2011/document2.pdf
- s3-dg.pdf

這些物件金鑰會建立以 Private、Development 與 Finance 作為根層級資料夾，並以 s3-dg.pdf 作為根層級物件的邏輯階層。當您在 Amazon S3 主控台上選擇儲存貯體名稱時，根層級項目會出現。此主控台會將最上層的字首 (Private/、Development/ 與 Finance/) 顯示為根層級資料夾。因為物件金鑰 s3-dg.pdf 沒有字首，所以會顯示為根層級項目。

演練摘要

您會在此演練中，建立其中含有三個資料夾 (Private、Development 和 Finance) 的儲存貯體。

您有兩名使用者：Alice 與 Bob。您希望 Alice 僅能夠存取 Development 資料夾，而您希望 Bob 僅能夠存取 Finance 資料夾。您希望讓 Private 資料夾內容保持未對外開放。在逐步解說中，您可以建立 IAM 使用者 (此範例使用使用者名稱 Alice 和 Bob) 並授與必要的權限來管理存取權。

IAM 也支援建立使用者群組和授予群組層級的許可，以套用至群組中的所有使用者。這可讓許可管理得更好。在此練習中，Alice 與 Bob 都需要一些一般許可。因此，您也會建立名為 Consultants 的群組，然後再於此群組中新增 Alice 與 Bob。您先將群組政策連接至該群組來授予許可。接著再將政策連接至特定的使用者，新增使用者專用的許可。

> **Note**
>
> 此演練使用 companybucket 作為儲存貯體名稱、使用 Alice 與 Bob 作為 IAM 使用者，並使用 Consultants 作為群組名稱。因為 Amazon S3 要求全域唯一的儲存貯體名稱，所以您必須將儲存貯體名稱換成您建立的名稱。

準備演練

在此範例中，您可以使用 AWS 帳戶 登入資料建立 IAM 使用者。這些使用者在一開始都不具備任何許可。您對這些使用者遞增授予許可來執行特定的 Amazon S3 動作。為測試這些許可，您將使用每位使用者的憑證登入主控台。當您以 AWS 帳戶 擁有者身分遞增授予許可並以 IAM 使用者身分測試許可時，您每次都需要使用不同的登入資料登入和登出。您可以使用瀏覽器來執行此測試，但如果您能夠使用兩個不同的瀏覽器，便能加快此程序的速度。使用一個瀏覽器 AWS Management Console 與您的 AWS 帳戶 登入資料連線到，另一個瀏覽器連線至 IAM 使用者登入資料。

若要 AWS Management Console 使用您的 AWS 帳戶 認證登入，請前往 https://console.aws.amazon.com/。IAM 使用者無法使用相同的連結登入。IAM 使用者必須使用已啟用 IAM 功能的登入頁面。帳戶擁有者可以將此連結提供給其使用者。

如需 IAM 的詳細資訊，請參閱《IAM 使用者指南》中的 AWS Management Console 登入頁面。

為 IAM 使用者提供登入連結

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 Navigation (導覽) 窗格中，選擇 IAM Dashboard (IAM 儀表板)。
3. 記下 IAM users sign in link: (IAM 使用者登入連結：) 下的 URL。將此連結提供給 IAM 使用者，讓他們使用 IAM 使用者名稱與密碼來登入主控台。

步驟 1：建立儲存貯體

在此步驟中，您可以使用登入資料 AWS 帳戶 登入 Amazon S3 主控台、建立儲存貯體、將資料夾新增至儲存貯體，然後在每個資料夾中上傳一或兩個範例文件。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 建立儲存貯體。

   如需 step-by-step 指示，請參閱建立儲存貯體。
3. 將一份文件上傳到儲存貯體。

   此練習假設您在此儲存貯體的根層級中已有 s3-dg.pdf 文件。若要上傳不同的文件，請將其檔案名稱替換為 s3-dg.pdf。
4. 將名為 Private、Finance 和 Development 的三個資料夾新增至儲存貯體。

   如需建立資料夾的指 step-by-step 示，請參閱 Amazon 簡易儲存服務使用者指南中的 在 Amazon S3 主控台中使用資料夾整理物件 >。
5. 在每個資料夾中上傳一到兩份文件。

   此練習假設您已在每個資料夾中上傳了一些文件，而且儲存貯體已有具有下列金鑰的物件：

   - Private/privDoc1.txt
   - Private/privDoc2.zip
   - Development/project1.xls
   - Development/project2.xls
   - Finance/Tax2011/document1.pdf
   - Finance/Tax2011/document2.pdf
   - s3-dg.pdf

   如需 step-by-step 指示，請參閱上傳物件。

步驟 2：建立 IAM 使用者與群組

現在，使用 IAM 主控台將兩個 IAM 使用者 (愛麗絲和 Bob) 新增至您的 AWS 帳戶. 如需 step-by-step 指示，請參閱 IAM 使用者指南 AWS 帳戶中的建立 IAM 使用者。

同時建立名為的系統管理群組Consultants。然後將兩個使用者新增至群組。如需 step-by-step 指示，請參閱建立 IAM 使用者群組。

> **Warning**
>
> 當您新增使用者與群組時，請勿指派任何會將許可授予這些使用者的政策。這些使用者在一開始不具有任何許可。您將在下列區段中授予額外的許可。您必須先確定已為這些 IAM 使用者指派密碼。您將使用這些使用者憑證來測試 Amazon S3 動作，並確認許可是否如預期般運作。

如需建立新 IAM 使用者的指 step-by-step 示，請參閱 IAM 使用者指南 AWS 帳戶中的「建立 IAM 使用者」。為此演練建立使用者時，請選取 AWS Management Console 存取，並清除程式化存取。

如需建立管理群組的指 step-by-step 示，請參閱《IAM 使用者指南》中的「建立您的第一個 IAM 管理員使用者和群組」。

步驟 3：確認 IAM 使用者沒有許可

如果您使用兩個瀏覽器，您現在可以使用第二個瀏覽器，以其中一個 IAM 使用者憑證來登入主控台。

1. 使用 IAM 使用者登入連結 (請參閱「為 IAM 使用者提供登入連結」)，使用任一組 IAM 使用者憑證登入 AWS Management Console 。
2. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

   驗證控制台消息告訴您訪問被拒絕。

您現在可以將額外的許可授予使用者。首先，您要連接群組政策，授予兩位使用者一些必要的許可。

步驟 4：授予群組層級的許可

您希望使用者都能執行下列動作：

- 列出父帳戶擁有的所有儲存貯體。要做到這一點，Bob 與 Alice 必須具備 `s3:ListAllMyBuckets` 動作的許可。
- 列出 `companybucket` 儲存貯體中的根層級項目、資料夾與物件。要做到這一點，Bob 與 Alice 必須具備可以對 `s3:ListBucket` 儲存貯體執行 `companybucket` 動作的許可。

首先，您建立會授予這些許可的政策，然後將該政策連接至 `Consultants` 群組。

步驟 4.1：授予列出所有儲存貯體的許可

在此步驟中，您將建立受管政策 (可將使用者基本許可授予使用者，以便其能列出父帳戶擁有的所有儲存貯體)。接著，您將政策連接至 Consultants 群組。當您將受管政策連接至使用者或群組時，即表示您授予該使用者或群組許可，讓他們可取得父 AWS 帳戶擁有之儲存貯體的清單。

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。

   > **Note**
   >
   > 由於您要授予使用者許可，請使用您的 AWS 帳戶 憑證來登入，不要以 IAM 使用者身分登入。

2. 建立受管政策。

   a. 在左邊的導覽窗格中，選擇 Policies (政策)，然後選擇 Create policy (建立政策)。

   b. 請選擇 JSON 索引標籤。

   c. 複製下列存取政策，並貼入政策的文字欄位中。

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "AllowGroupToSeeBucketListInTheConsole",
         "Action": ["s3:ListAllMyBuckets"],
         "Effect": "Allow",
         "Resource": ["arn:aws:s3:::*"]
       }
     ]
   }
   ```

   政策是 JSON 文件。在此文件中，Statement 是物件陣列，每一個物件均使用一組名稱值對來描述許可。前面的政策在描述一項特定的許可。Action 會指定存取類型。在政策中，s3:ListAllMyBuckets 是預先定義的 Amazon S3 動作。此動作涵蓋 Amazon S3 GET 服務操作，該操作會傳回已驗證寄件者擁有的所有儲存貯體清單。Effect 元素值決定要允許或拒絕特定許可。

   d. 選擇 Review Policy (檢閱政策)。在下一個頁面中，於 Name (名稱) 欄位輸入 AllowGroupToSeeBucketListInTheConsole，接著選擇 Create policy (建立政策)。

> **Note**
>
> Summary (摘要) 項目會顯示訊息，指出該政策未授予任何許可。就本演練為例，您可放心地忽略此訊息。

3. 將您建立的 `AllowGroupToSeeBucketListInTheConsole` 受管政策連接至 `Consultants` 群組。

   如需附加受管政策的指 step-by-step 示，請參閱 IAM 使用者指南中的新增和移除 IAM 身分許可。

   您在 IAM 主控台中將政策文件連接至 IAM 使用者和群組。您希望這兩個使用者都能夠列出儲存貯體，因此將該政策連接至該群組。

4. 測試許可。

   a. 使用 IAM 使用者登入連結 (請參閱 為 IAM 使用者提供登入連結)，利用任一組 IAM 使用者憑證來登入主控台。

   b. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

      主控台現在應會列出所有儲存貯體，但不會列出任何儲存貯體中的物件。

步驟 4.2：允許使用者列出儲存貯體的根層級內容

接下來，我們允許所有使用者在 `Consultants` 群組內列出根層級的 `companybucket` 儲存貯體項目。當使用者在 Amazon S3 主控台上選擇公司儲存貯體時，使用者可以看到儲存貯體中的根層級項目。

> **Note**
>
> 此範例使用 `companybucket` 進行說明。您必須使用您建立的儲存貯體名稱。

若要瞭解當您選擇儲存貯體名稱、Amazon S3 傳回的回應，以及主控台如何解譯回應時，主控台傳送給 Amazon S3 的請求，請仔細檢查流程。

當您選擇儲存貯體名稱時，主控台會將 GET 儲存貯體 (列出物件) 請求傳送給 Amazon S3。此要求包含下列參數：

- 其值為空字串的 `prefix` 參數。

- 其值為 delimiter 的 / 參數。

下列是範例要求。

```
GET ?prefix=&delimiter=/ HTTP/1.1
Host: companybucket.s3.amazonaws.com
Date: Wed, 01 Aug  2012 12:00:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Amazon S3 傳回包含下列 <ListBucketResult/> 元素的回應。

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>companybucket</Name>
  <Prefix></Prefix>
  <Delimiter>/</Delimiter>
   ...
  <Contents>
    <Key>s3-dg.pdf</Key>
    ...
  </Contents>
  <CommonPrefixes>
    <Prefix>Development/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>Finance/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>Private/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

金鑰 s3-dg.pdf 物件不含斜線 (/) 分隔符號，而 <Contents> 會在 Amazon S3 元素中傳回金鑰。不過，範例儲存貯體中的所有其他金鑰都包含 / 分隔符號。Amazon S3 將這些金鑰分組，並針對每個不同的字首值 <CommonPrefixes>、Development/ 和 Finance/，傳回一個 Private/ 元素，這是從這些金鑰開頭到第一次出現指定的 / 分隔符號為止的子字串。

此主控台會解譯此結果，並將根層級項目顯示為三個資料夾與一個物件金鑰。

如果 Bob 或 Alice 開啟 Development 資料夾，主控台會將 GET 儲存貯體 (列出物件) 請求傳送給 Amazon S3，其中 prefix 和 delimiter 參數會設為下列值：

- prefix 參數，值為 Development/。
- delimiter 參數，值為「/」。

Amazon S3 在回應中會傳回開頭為指定字首的物件金鑰。

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>companybucket</Name>
  <Prefix>Development</Prefix>
  <Delimiter>/</Delimiter>
   ...
  <Contents>
    <Key>Project1.xls</Key>
    ...
  </Contents>
  <Contents>
    <Key>Project2.xls</Key>
    ...
  </Contents>
</ListBucketResult>
```

此主控台會顯示物件金鑰。

現在重新授予使用者許可，以列出根層級的儲存貯體項目。若要列出儲存貯體內容，使用者需要呼叫 s3:ListBucket 動作的許可，如下列政策陳述式所示。為了確保使用者只會看到根層級內容，您新增條件來要求使用者必須在請求中指定空的 prefix，亦即不允許使用者按兩下任何根層級資料夾。最後，您會再新增一項條件，要求使用者要求中必須包含值為「delimiter」的 / 參數，才能要求資料夾的存取權。

```
{
  "Sid": "AllowRootLevelListingOfCompanyBucket",
  "Action": ["s3:ListBucket"],
  "Effect": "Allow",
  "Resource": ["arn:aws:s3:::companybucket"],
  "Condition":{
         "StringEquals":{
             "s3:prefix":[""], "s3:delimiter":["/"]
                       }
              }
}
```

當您在 Amazon S3 主控台上選擇儲存貯體時，主控台會先傳送 GET 儲存貯體位置請求，以尋找儲 AWS 區域 存貯體的部署位置。接著，主控台會使用儲存貯體的區域專用端點，傳送 GET 儲存貯體 (列出物件) 請求。因此，使用者若要使用主控台，您必須授予 s3:GetBucketLocation 動作的許可，如下列政策陳述式所示。

```
{
   "Sid": "RequiredByS3Console",
   "Action": ["s3:GetBucketLocation"],
   "Effect": "Allow",
   "Resource": ["arn:aws:s3:::*"]
}
```

允許使用者列出根層級儲存貯體內容

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。

   使用您的登入 AWS 帳戶 資料 (而非 IAM 使用者的登入資料) 登入主控台。

2. 以下列政策取代連接至 Consultants 群組的現有 AllowGroupToSeeBucketListInTheConsole 受管政策。此政策同樣也會允許 s3:ListBucket 動作。請記住*companybucket*在策略中Resource以值區的名稱取代。

   如需 step-by-step 指示，請參閱 IAM 使用者指南中的編輯 IAM 政策。按照 step-by-step 指示進行操作時，請務必遵循將變更套用至附加原則之所有主參與者實體的步驟。

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
        {
          "Sid":
    "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",
          "Action": [ "s3:ListAllMyBuckets", "s3:GetBucketLocation" ],
          "Effect": "Allow",
          "Resource": [ "arn:aws:s3:::*"  ]
        },
        {
          "Sid": "AllowRootLevelListingOfCompanyBucket",
          "Action": ["s3:ListBucket"],
          "Effect": "Allow",
          "Resource": ["arn:aws:s3:::companybucket"],
          "Condition":{
   ```

```
            "StringEquals":{
                  "s3:prefix":[""], "s3:delimiter":["/"]
                        }
                  }
     }
  ]
}
```

3. 測試更新後的許可。

   a. 使用 IAM 使用者登入連結 (請參閱為 IAM 使用者提供登入連結) 登入 AWS Management Console。

      開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

   b. 選擇您建立的儲存貯體，且主控台會顯示根層級儲存貯體項目。因為您尚未獲得指派這些許可，所以當您在儲存貯體中選擇任何資料夾時，會看不到資料夾內容。

當使用者使用 Amazon S3 主控台時，此測試會成功。在主控台上選擇儲存貯體時，主控台實作會傳送請求，該請求會包含值為空字串的 `prefix` 參數，以及其值為「`delimiter`」的 `/` 參數。

步驟 4.3：群組政策摘要

您新增的群組政策最終目的是將下列最低許可授予 IAM 使用者 Alice 與 Bob：

- 列出父帳戶擁有的所有儲存貯體。
- 查看 `companybucket` 儲存貯體中的根層級項目。

即使如此，使用者能執行的操作仍然有限。接下來我們要授予使用者專用的許可，如下所示：

- 允許 Alice 在 `Development` 資料夾中取得物件與將物件放置到其中。
- 允許 Bob 在 `Finance` 資料夾中取得物件與將物件放置到其中。

對於使用者專用的許可，您應將政策指派給特定使用者而非群組。在下列區段中，您將授予 Alice 在 `Development` 資料夾中工作的許可。您可以重複這些步驟，將類似的許可授予 Bob，以便在 `Finance` 資料夾中工作。

步驟 5：將特定許可授予 IAM 使用者 Alice

您現在要將額外的許可授予 Alice，讓她可以看到 `Development` 資料夾的內容，並從該資料夾取得物件與將物件放置到其中。

步驟 5.1：對 IAM 使用者 Alice 授予許可來列出 Development 資料夾內容

若要讓 Alice 列出`Development`資料夾內容，您必須將原則套用至授與`companybucket`值區`s3:ListBucket`動作權限的使用者 Alice，前提是要求包含前置詞`Development/`。您只想將此政策套用到使用者 Alice，因此會使用內嵌政策。如需內嵌政策的詳細資訊，請參閱《IAM 使用者指南》中的受管政策和內嵌政策。

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。

   使用您的登入 AWS 帳戶 資料 (而非 IAM 使用者的登入資料) 登入主控台。
2. 建立內嵌政策，將列出 `Development` 資料夾內容的許可授予使用者 Alice。

   a. 在左側導覽窗格中，選擇 Users (使用者)。

   b. 選擇用戶名愛麗絲。

   c. 在使用者詳細資訊頁面上，選擇 Permissions (許可) 索引標籤，然後選擇 Add inline policy (新增內嵌政策)。

   d. 請選擇 JSON 索引標籤。

   e. 複製下列原則，並將其貼到原則文字欄位中。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
      "Sid": "AllowListBucketIfSpecificPrefixIsIncludedInRequest",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::companybucket"],
      "Condition":{  "StringLike":{"s3:prefix":["Development/*"] }
       }
    }
  ]
}
```

   f. 選擇 Review Policy (檢閱政策)。在下一個頁面中，於 Name (名稱) 欄位輸入一個名稱，接著選擇 Create policy (建立政策)。
3. 測試 Alice 的許可變更：

   a. 使用 IAM 使用者登入連結 (請參閱為 IAM 使用者提供登入連結) 登入 AWS Management Console。

b. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

c. 在 Amazon S3 主控台上，確認 Alice 可以看到儲存貯體中 Development/ 資料夾中的物件清單。

當使用者選擇 /Development 資料夾來查看其中的物件清單時，Amazon S3 主控台會將含有字首 ListObjects 的 /Development 請求傳送給 Amazon S3。因為使用者獲許可查看字首為 Development 和分隔符號為 / 的物件清單，所以 Amazon S3 會傳回金鑰前綴為 Development/ 的物件清單，而主控台會顯示該清單。

步驟 5.2：對 IAM 使用者 Alice 授予許可在 Development 資料夾中取得和放置物件

Alice 若要能從 Development 資料夾取得物件與將物件放置到其中，就需要呼叫 s3:GetObject 與 s3:PutObject 動作的許可。下列政策陳述式可以授予這些許可，但前提是要求中必須包含值為 prefix 的 Development/ 參數。

```
{
    "Sid":"AllowUserToReadWriteObjectData",
    "Action":["s3:GetObject", "s3:PutObject"],
    "Effect":"Allow",
    "Resource":["arn:aws:s3:::companybucket/Development/*"]
 }
```

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

   使用您的登入 AWS 帳戶 資料 (而非 IAM 使用者的登入資料) 登入主控台。

2. 編輯您在前一個步驟中建立的內嵌政策。

   a. 在左側導覽窗格中，選擇 Users (使用者)。

   b. 選擇使用者名稱 Alice。

   c. 在使用者詳細資訊頁面上，選擇 Permissions (許可) 標籤，然後展開 Inline Policies (內嵌政策) 區段。

   d. 在前一個步驟中建立的政策名稱旁，選擇 Edit Policy (編輯政策)。

   e. 複製下列政策並貼到政策文字欄位，以取代現有政策。

```
{
     "Version": "2012-10-17",
```

```
     "Statement":[
      {
         "Sid":"AllowListBucketIfSpecificPrefixIsIncludedInRequest",
         "Action":["s3:ListBucket"],
         "Effect":"Allow",
         "Resource":["arn:aws:s3:::companybucket"],
         "Condition":{
            "StringLike":{"s3:prefix":["Development/*"]
            }
         }
      },
      {
        "Sid":"AllowUserToReadWriteObjectDataInDevelopmentFolder",
        "Action":["s3:GetObject", "s3:PutObject"],
        "Effect":"Allow",
        "Resource":["arn:aws:s3:::companybucket/Development/*"]
      }
   ]
}
```

3. 測試更新後的政策：

   a. 使用 IAM 使用者登入連結 (請參閱為 IAM 使用者提供登入連結) 登入 AWS Management Console。

   b. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

   c. 在 Amazon S3 主控台上，確認 Alice 現在已可以在 Development 資料夾中新增和下載物件。

步驟 5.3：對 IAM 使用者 Alice 明確拒絕儲存貯體中任何其他資料夾的許可

使用者 Alice 現在可以列出 companybucket 儲存貯體的根層級內容，也可以從 Development 資料夾取得物件與將物件放置到其中。若您確實想要限縮存取許可，可以明確拒絕 Alice 存取儲存貯體中的任何其他資料夾。如有任何其他政策 (儲存貯體政策或 ACL) 授予 Alice 存取儲存貯體中的任何其他資料夾，則這項明確拒絕將會覆寫這些許可。

您可以將下列陳述式新增至使用者 Alice 政策，以要求 Alice 傳送給 Amazon S3 的所有請求都包含 prefix 參數，值可以是 Development/* 或空字串。

```
{
   "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",
```

```
   "Action": ["s3:ListBucket"],
   "Effect": "Deny",
   "Resource": ["arn:aws:s3:::companybucket"],
   "Condition":{  "StringNotLike": {"s3:prefix":["Development/*",""] },
                  "Null"         : {"s3:prefix":false }
    }
}
```

Condition 區塊中有兩個條件式表達式。這些條件式表達式的結果使用了邏輯 AND 加以合併。當這兩項條件皆為 true 時，合併後條件的結果也會是 true。因為在此政策中的 Effect 為 Deny，當 Condition 評估結果為 true 時，使用者將無法執行指定的 Action。

- Null 條件式表達式會確定來自 Alice 的要求包含了 prefix 參數。

  prefix 參數必須能夠存取資料夾。如果您傳送的請求不含 prefix 參數，Amazon S3 會傳回所有物件金鑰。

  若要求中包含了 null 值的 prefix 參數，表達式會評估為 true，因此整個 Condition 也會評估為 true。您必須允許 prefix 參數使用空字串作為值。從前面的討論，回想那允許 null 字串讓 Alice 可以像先前討論主控台時執行的操作一樣，擷取根層級儲存貯體項目。如需詳細資訊，請參閱 步驟 4.2：允許使用者列出儲存貯體的根層級內容。
- StringNotLike 條件式表達式可確保若指定 prefix 參數的值，而非指定 Development/*，要求會失敗。

遵循以上區段中的步驟，並再次更新您為使用者 Alice 建立的內嵌政策。

複製下列政策並貼到政策文字欄位，以取代現有政策。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllowListBucketIfSpecificPrefixIsIncludedInRequest",
         "Action":["s3:ListBucket"],
         "Effect":"Allow",
         "Resource":["arn:aws:s3:::companybucket"],
         "Condition":{
            "StringLike":{"s3:prefix":["Development/*"]
            }
         }
      },
```

```
      {
        "Sid":"AllowUserToReadWriteObjectDataInDevelopmentFolder",
        "Action":["s3:GetObject", "s3:PutObject"],
        "Effect":"Allow",
        "Resource":["arn:aws:s3:::companybucket/Development/*"]
      },
      {
         "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",
         "Action": ["s3:ListBucket"],
         "Effect": "Deny",
         "Resource": ["arn:aws:s3:::companybucket"],
         "Condition":{  "StringNotLike": {"s3:prefix":["Development/*",""] },
                        "Null"         : {"s3:prefix":false }
          }
      }
   ]
}
```

步驟 6：將特定許可授予 IAM 使用者 Bob

現在，您要授予 Bob 對 Finance 資料夾的許可。請遵循您稍早用來為 Alice 授予許可的步驟，但將 Development 資料夾置換成 Finance 資料夾。如需 step-by-step 指示，請參閱步驟 5：將特定許可授予 IAM 使用者 Alice。

步驟 7：保護 Private 資料夾

在此範例中，您只有兩名使用者。您授予了群組層級所有必要的基本許可，並只有在您確實需要授予個別使用者層級的許可時，才授予使用者層級許可。此方法可將管理許可的工作降至最低。隨著使用者人數增加，管理許可可能會愈來愈麻煩。例如，我們不希望讓此範例中的任何使用者存取 Private 資料夾內容。如何確保您不會意外授予用戶對該Private文件夾的權限？ 您新增政策明確拒絕存取該資料夾。明確拒絕會覆寫其他任何許可。

若要確定 Private 資料夾未對外開放，可以在群組政策中新增下列兩個拒絕陳述式：

- 新增下列陳述式，明確拒絕任何對 Private 資料夾資源執行的動作 (companybucket/Private/*)。

```
{
  "Sid": "ExplictDenyAccessToPrivateFolderToEveryoneInTheGroup",
  "Action": ["s3:*"],
  "Effect": "Deny",
  "Resource":["arn:aws:s3:::companybucket/Private/*"]
```

```
}
```

- 要求中如有指定 Private/ 字首，您也可以拒絕列出物件動作的許可。在主控台上，如果 Bob 或 Alice 開啟 Private 資料夾，此政策會導致 Amazon S3 傳回錯誤回應。

```
{
  "Sid": "DenyListBucketOnPrivateFolder",
  "Action": ["s3:ListBucket"],
  "Effect": "Deny",
  "Resource": ["arn:aws:s3:::*"],
  "Condition":{
      "StringLike":{"s3:prefix":["Private/"]}
   }
}
```

以更新後包含前述拒絕陳述式的政策取代 Consultants 群組政策。套用更新後的政策後，群組中將無任何使用者可以存取儲存貯體中的 Private 資料夾。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

   使用您的登入 AWS 帳戶 資料 (而非 IAM 使用者的登入資料) 登入主控台。

2. 以下列政策取代連接至 Consultants 群組的現有 AllowGroupToSeeBucketListInTheConsole 受管政策。請務必將政策中的 *companybucket* 取代為您的儲存貯體名稱。

   如需相關指示，請參閱 IAM 使用者指南中的編輯客戶受管政策。執行逐步說明時，請務必遵循指示，將您的變更套用到獲指派該政策的所有主體實體。

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid":
    "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",
         "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
         "Effect": "Allow",
         "Resource": ["arn:aws:s3:::*"]
       },
       {
   ```

```
      "Sid": "AllowRootLevelListingOfCompanyBucket",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::companybucket"],
      "Condition":{
          "StringEquals":{"s3:prefix":[""]}
       }
    },
    {
      "Sid": "RequireFolderStyleList",
      "Action": ["s3:ListBucket"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::*"],
      "Condition":{
          "StringNotEquals":{"s3:delimiter":"/"}
       }
     },
    {
      "Sid": "ExplictDenyAccessToPrivateFolderToEveryoneInTheGroup",
      "Action": ["s3:*"],
      "Effect": "Deny",
      "Resource":["arn:aws:s3:::companybucket/Private/*"]
    },
    {
      "Sid": "DenyListBucketOnPrivateFolder",
      "Action": ["s3:ListBucket"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::*"],
      "Condition":{
          "StringLike":{"s3:prefix":["Private/"]}
       }
    }
  ]
}
```

步驟 8：清理

若要清理，請開啟 IAM 主控台並移除使用者愛麗絲和 Bob。如需 step-by-step 指示，請參閱 IAM 使用者指南中的刪除 IAM 使用者。

為確保您未因儲存而被收費，您也應刪除您為此練習建立的物件與儲存貯體。

相關資源

- 《IAM 使用者指南》中的管理 IAM 政策

使用政策管理 Amazon S3 資源存取權的逐步解說

本主題提供以下有關授予 Amazon S3 資源存取的簡介演練範例。這些範例使用 AWS Management Console 建立資源 (值區、物件、使用者) 並授與權限。範例接著會示範如何使用命令列工具來驗證許可，因此您不需要撰寫任何程式碼。我們提供指令同時使用 AWS Command Line Interface (AWS CLI) 和 AWS Tools for Windows PowerShell.

- 範例 1：為其使用者授予儲存貯體許可的儲存貯體擁有者

  您在帳戶中建立的 IAM 使用者預設沒有任何許可。在此練習中，您會授予使用者許可來執行儲存貯體與物件操作。

- 範例 2：授予跨帳戶儲存貯體許可的儲存貯體擁有者

  在此練習中，儲存貯體擁有者 (帳戶 A) 會將跨帳戶許可授予另一個 AWS 帳戶(帳戶 B)。帳戶 B 接著會將這些許可委派給其帳戶中的使用者。

- 在物件擁有者與儲存貯體擁有者不同的情況下管理物件許可

  此案例的情境範例是儲存貯體擁有者想要將物件許可授予其他人，但儲存貯體中並非所有物件都由儲存貯體擁有者所擁有。儲存貯體擁有者需要哪些許可，如何才能委派這些許可？

  建立值區的 AWS 帳戶 稱為「值區擁有者」。擁有者可以授與其他上載物件的 AWS 帳戶 權限，而建立物件的權限則擁有 AWS 帳戶 這些物件。儲存貯體擁有者並不擁有其他 AWS 帳戶所建立物件的許可。如果值區擁有者撰寫值區政策授與物件存取權，則該政策不會套用至其他帳戶所擁有的物件。

  在此情況下，物件擁有者必須先使用物件 ACL 將許可授予儲存貯體擁有者。然後，值區擁有者可以將這些物件權限委派給其他人、自己帳戶中的使用者，或委派給其他使用者 AWS 帳戶，如下列範例所示。

  - 範例 3：授予對其未擁有之物件的許可的儲存貯體擁有者

    在此練習中，儲存貯體擁有者必須先從物件擁有者取得許可。儲存貯體擁有者可接著將這些許可委派給其專屬帳戶中的使用者。

  - 範例 4-值區擁有者將跨帳戶權限授與其不擁有的物件

    收到物件擁有者的權限後，值區擁有者無法將權限委派給其他人， AWS 帳戶 因為不支援跨帳戶委派 (請參閱許可委派)。相反地，值區擁有者可以建立具有執行特定作業 (例如 get object) 許可的 IAM 角色，並允許其他 AWS 帳戶 人擔任該角色。擔任該角色的任何人，皆可存取物件。此範例示範儲存貯體擁有者如何使用 IAM 角色來啟用此跨帳戶委派。

嘗試演練範例之前

這些範例使用 AWS Management Console 建立資源並授與權限。若要測試權限，範例會使用命令列工具 AWS CLI、和 AWS Tools for Windows PowerShell，因此您不需要撰寫任何程式碼。若要測試許可，您必須設定其中一個工具。如需詳細資訊，請參閱 設定逐步解說的工具。

此外，在建立資源時，這些範例不會使用 AWS 帳戶. 反之，您會在這些帳戶中建立管理員使用者來執行這些任務。

關於使用管理員使用者來建立資源並授予許可

AWS Identity and Access Management (IAM) 建議不要使用您的 AWS 帳戶 根使用者憑證來提出請求。反之，請建立 IAM 使用者或角色，並授予完整存取許可，然後使用該使用者或角色憑證來發出請求。我們將此稱為管理員使用者或角色。如需詳細資訊，請參閱《AWS 一般參考》中的 AWS 帳戶根使用者 憑證與 IAM 身分，以及《IAM 使用者指南》中的 IAM 最佳實務。

本節中的所有演練範例會使用管理員使用者憑證。如果您尚未為您建立系統管理員使用者 AWS 帳戶，則主題會顯示如何進行。

若要使用 AWS Management Console 使用者認證登入，您必須使用 IAM 使用者登入 URL。IAM 主控台會為您的 AWS 帳戶. 這些主題示範如何取得 URL。

設定逐步解說的工具

介紹性範例 (請參閱使用政策管理 Amazon S3 資源存取權的逐步解說) 使用建立 AWS Management Console 資源和授與權限。為了測試權限，這些示例使用命令行工具， AWS Command Line Interface (AWS CLI) 和 AWS Tools for Windows PowerShell，因此您不需要編寫任何代碼。若要測試許可，您必須設定其中一個工具。

若要設定 AWS CLI

1. 下載和設定 AWS CLI。如需相關指示，請參閱《AWS Command Line Interface 使用者指南》中的下列主題：

   安裝或更新到最新版本的 AWS Command Line Interface

   開始使用 AWS Command Line Interface
2. 設定預設描述檔。

   您可以將使用者認證儲存在 AWS CLI 組態檔中。使用您的憑據在配置文件中創建默 AWS 帳戶 認配置文件。如需尋找和編輯組 AWS CLI 態檔的指示，請參閱組態和認證檔案設定。

```
[default]
aws_access_key_id = access key ID
aws_secret_access_key = secret access key
region = us-west-2
```

3. 在命令提示字元中輸入下列命令，以驗證設定。這兩個命令不會明確提供憑證，因此會使用預設描述檔的憑證。

   - 嘗試命help令。

     ```
     aws help
     ```

   - 若要取得已設定帳戶上的值區清單，請使用aws s3 ls指令。

     ```
     aws s3 ls
     ```

逐步解說時，您將會建立使用者，並建立設定檔，將使用者認證儲存在組態檔中，如下列範例所示。這些設定檔具有AccountAadmin和的名稱AccountBadmin。

```
[profile AccountAadmin]
aws_access_key_id = User AccountAadmin access key ID
aws_secret_access_key = User AccountAadmin secret access key
region = us-west-2

[profile AccountBadmin]
aws_access_key_id = Account B access key ID
aws_secret_access_key = Account B secret access key
region = us-east-1
```

若要使用這些使用者憑證來執行命令，請指定描述檔名稱以新增 `--profile` 參數。下列指 AWS CLI 令會擷取中的物件清單，*examplebucket*並指定AccountBadmin設定檔。

```
aws s3 ls s3://examplebucket --profile AccountBadmin
```

或者，您可以從命令提示字元變更 AWS_DEFAULT_PROFILE 環境變數，將其中一組使用者憑證設定為預設描述檔。完成此操作後，無論何時執行沒有--profile參數的 AWS CLI 命令，都 AWS CLI 會使用您在環境變數中設定的設定檔作為預設設定檔。

```
$ export AWS_DEFAULT_PROFILE=AccountAadmin
```

若要設定 AWS Tools for Windows PowerShell

1. 下載和設定 AWS Tools for Windows PowerShell。如需指示，請移至《使用指南》 AWS Tools for Windows PowerShell中的AWS Tools for Windows PowerShell 〈安裝〉。

   > **Note**
   >
   > 若要載入 AWS Tools for Windows PowerShell 模組，您必須啟用PowerShell指令碼執行。如需詳細資訊，請參閱使用指南中的「啟AWS Tools for Windows PowerShell 用指令碼執行」。

2. 對於這些逐步解說，您可以使用指令來指定每個工作階段的 AWS `Set-AWSCredentials`認證。該命令會將憑證儲存至持續性存放區 (`-StoreAs` 參數)。

   ```
   Set-AWSCredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -
   storeas string
   ```

3. 驗證設定。

   - 若要擷取可用於 Amazon S3 操作的可用命令清單，請執行`Get-Command`命令。

     ```
     Get-Command -module awspowershell -noun s3* -StoredCredentials string
     ```

   - 若要擷取值區中的物件清單，請執行`Get-S3Object`指令。

     ```
     Get-S3Object -BucketName bucketname -StoredCredentials string
     ```

如需命令的清單，請參閱 PowerShellCmdlet 參考的AWS 工具。

現在，您已準備好嘗試逐步解說。按照每個部分開頭提供的鏈接。

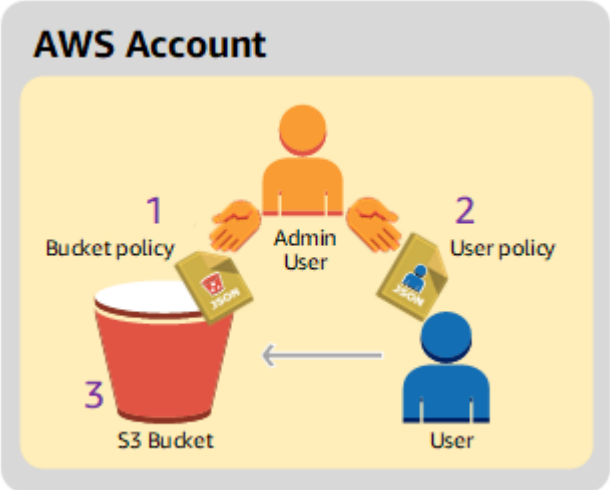
範例 1：為其使用者授予儲存貯體許可的儲存貯體擁有者

> **Important**
>
> 將許可授予 IAM 角色比將許可授予個別使用者更好的做法。如需有關如何授予 IAM 角色許可的詳細資訊，請參閱。了解跨帳戶許可和使用 IAM 角色

主題

在本逐步解說中， AWS 帳戶 擁有值區，而帳戶包含 IAM 使用者預設，使用者沒有權限。父帳戶必須授予使用者許可，使用者才能執行任何任務。儲存貯體擁有者及父帳戶是相同的。因此，若要授與使用者儲存貯體的權限， AWS 帳戶 可以使用值區政策、使用者原則或兩者。帳戶擁有者將使用儲存貯體政策授予部分許可，並使用使用者政策授予其他許可。

以下是演練步驟的摘要：



1. 帳戶管理員建立儲存貯體政策，並將一組許可授予使用者。
2. 帳戶管理員向使用者連接使用者政策，並授予其他許可。
3. 使用者隨後嘗試透過儲存貯體政策與使用者政策取得的許可。

在此範例中，您將需要一個 AWS 帳戶. 您必須建立管理員使用者，而非使用帳戶的根使用者憑證 (請參閱「關於使用管理員使用者來建立資源並授予許可」)。我們指的是 AWS 帳戶 和管理員使用者，如下表所示。

帳戶 ID	帳戶稱為	帳戶中的管理員使用者
1111-1111-1111	帳戶 A	AccountAadmin

> **Note**
>
> 此範例中的系統管理員使用者是AccountAadmin指帳戶 A，而非帳戶 A AccountAdmin。

所有建立使用者與授予許可的任務都是在 AWS Management Console完成。若要驗證權限，逐步解說會使用指令行工具 AWS Command Line Interface (AWS CLI) 和 AWS Tools for Windows PowerShell，因此您不需要撰寫任何程式碼。

準備演練

1. 確保您有一個 AWS 帳戶 並且它具有具有管理員權限的用戶。

   a. 如果需要 AWS 帳戶，請註冊。我們將此帳戶稱為帳戶 A。

      i. 轉到 https://aws.amazon.com/s3，然後選擇創建一個 AWS 帳戶。

      ii. 遵循螢幕說明。

      AWS 當您的帳戶處於活動狀態並可供您使用時，將通過電子郵件通知您。

   b. 在帳戶 A 中，建立系統管理員使用者**AccountAadmin**。使用帳戶 A 憑證來登入 IAM 主控台，然後執行下列操作：

      i. 建立使用者**AccountAadmin**並記下使用者安全認證。

      如需指示，請參閱 IAM 使用者指南 AWS 帳戶中的建立 IAM 使用者。

      ii. 附加提供完整存取權AccountAadmin的使用者原則，以授予管理員權限。

      如需說明，請參閱《IAM 使用者指南》中的管理 IAM 政策。

iii. 請注意的 IAM 使用者登入 URL AccountAadmin。登入 AWS Management Console時，您必須使用此 URL。如需有關在何處尋找登入 URL 的詳細資訊，請參閱 IAM 使用者指南中的 AWS Management Console 以 IAM 使用者身分登入。請注意每個帳戶的 URL。

2. 設定 AWS CLI 或 AWS Tools for Windows PowerShell。請確定您儲存系統管理員使用者認證，如下所示：

   - 如果使用 AWS CLI，請在組態檔`AccountAadmin`中建立設定檔。
   - 如果使用 AWS Tools for Windows PowerShell，請確定您將工作階段的認證儲存為`AccountAadmin`。

   如需說明，請參閱設定逐步解說的工具。

步驟 1：在帳戶 A 中建立資源並授與權限

使用帳戶 A `AccountAadmin` 中的使用者登入資料和特殊 IAM 使用者登入 URL，登入 AWS Management Console 並執行下列動作：

1. 建立值區和 IAM 使用者的資源

   a. 在 Amazon S3 主控台中建立儲存貯體。請注意您 AWS 區域 在其中建立值區的位置。如需說明，請參閱建立儲存貯體。

   b. 在 IAM 主控台中，執行下列動作：

      i. 建立名為 Dave 的使用者。

      如需 step-by-step 指示，請參閱 IAM 使用者指南中的建立 IAM 使用者 (主控台)。

      ii. 請注意`UserDave`認證。

      iii. 請注意使用者戴夫的 Amazon 資源名稱 (ARN)。在 IAM 主控台中選取使用者，「摘要」索引標籤會提供使用者 ARN。

2. 授予權限。

   由於值區擁有者和使用者所屬的父帳戶相同，因此 AWS 帳戶 可以使用值區政策、使用者策略或兩者同時授予使用者權限。在本範例中兩種方式都會使用。若物件也屬於相同帳戶，儲存貯體擁有者即可以儲存貯體政策或 IAM 政策來授予物件許可。

   a. 在 Amazon S3 主控台中，將下列儲存貯體政策連至 *awsexamplebucket1*。

此政策具有兩個陳述式。

- 第一個陳述式會授予 Dave 儲存貯體操作許可 s3:GetBucketLocation 與 s3:ListBucket。
- 第二個陳述式則會授予 s3:GetObject 許可。因為帳戶 A 也擁有該物件，所以帳戶管理員能夠授予 s3:GetObject 許可。

在 Principal 陳述式中，Dave 將以其 ARN 識別。如需政策元素的詳細資訊，請參閱「Amazon S3 中的政策和許可」。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "statement1",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
         },
         "Action": [
            "s3:GetBucketLocation",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::awsexamplebucket1"
         ]
      },
      {
         "Sid": "statement2",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
         },
         "Action": [
             "s3:GetObject"
         ],
         "Resource": [
            "arn:aws:s3:::awsexamplebucket1/*"
         ]
      }
   ]
```

```
}
```

b. 使用下列政策，建立使用者 Dave 的內嵌政策。政策也會將 `s3:PutObject` 許可授予使用者 Dave。您必須提供儲存貯體名稱以更新政策。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "PermissionForObjectOperations",
         "Effect": "Allow",
         "Action": [
            "s3:PutObject"
         ],
         "Resource": [
            "arn:aws:s3:::awsexamplebucket1/*"
         ]
      }
   ]
}
```

如需指示，請參閱《IAM 使用者指南》中的「管理 IAM 政策」。請記得您需要使用帳戶 A 憑證來登入主控台。

步驟 2：測試許可

使用 Dave 的憑證，驗證該許可可用。您可以從下列兩個程序中使用其中一項來進行：

使用測試權限 AWS CLI

1. 透過新增下列`UserDaveAccountA`設定檔來更新 AWS CLI 組態檔案。如需詳細資訊，請參閱 設定逐步解說的工具。

```
[profile UserDaveAccountA]
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
region = us-east-1
```

2. 驗證 Dave 可執行使用者政策中授予的操作。使用下列 AWS CLI `put-object`命令上傳範例物件。

命令中的 ---body 參數會識別要上傳的來源檔案。例如，如果檔案位於機器上 C: 磁碟Windows機的根目錄中，您可以指定c:\HappyFace.jpg。--key 參數提供物件的金鑰名稱。

```
aws s3api put-object --bucket awsexamplebucket1 --key HappyFace.jpg --
body HappyFace.jpg --profile UserDaveAccountA
```

運行以下 AWS CLI 命令以獲取對象。

```
aws s3api get-object --bucket awsexamplebucket1 --key HappyFace.jpg OutputFile.jpg
 --profile UserDaveAccountA
```

使用測試權限 AWS Tools for Windows PowerShell

1. 將戴夫的憑據存儲為AccountADave. 然後，您將這些認證用於PUT和GET對象。

```
set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas
 AccountADave
```

2. 使用使用者 Dave 的預存認證，使用 AWS Tools for Windows PowerShell Write-S3Object命令上傳範例物件。

```
Write-S3Object -bucketname awsexamplebucket1 -key HappyFace.jpg -file HappyFace.jpg
 -StoredCredentials AccountADave
```

下載上述已上傳的物件。

```
Read-S3Object -bucketname awsexamplebucket1 -key HappyFace.jpg -file Output.jpg -
StoredCredentials AccountADave
```

範例 2：授予跨帳戶儲存貯體許可的儲存貯體擁有者

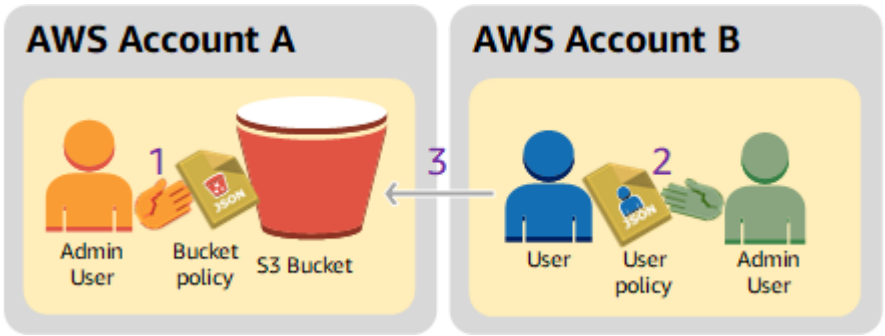
> **Important**
>
> 將權限授予 IAM 角色比授予個別使用者更好。若要了解如何操作，請參閱 了解跨帳戶許可和使用 IAM 角色。

主題

AWS 帳戶例如，帳戶 A — 可以授與另一個 AWS 帳戶帳戶 B 存取其資源 (例如值區和物件) 的權限。帳戶 B 接著可以將這些許可委派給其帳戶中的使用者。在此案例範例中，儲存貯體擁有者會將跨帳戶許可授予另一個帳戶，以執行特定儲存貯體操作。

> **Note**
>
> 帳戶 A 也可以直接將使用儲存貯體政策的許可授予帳戶 B 中的使用者。不過，即使帳戶 B 沒有帳戶 A 的權限，使用者仍需要使用者所屬之父帳號帳號 B 的權限，只要使用者同時擁有資源擁有者與父帳號的權限，使用者就能夠存取資源。

下列是演練步驟的摘要：



1. 帳戶 A 管理員使用者會將授予跨帳戶許可的儲存貯體政策連接至帳戶 B，以執行特定儲存貯體操作。

   請注意，帳戶 B 中的管理員使用者將自動繼承許可。

2. 帳戶 B 管理員使用者會將使用者政策連接至委派接收自帳戶 A 之許可的使用者。
3. 帳戶 B 中的使用者接著會存取帳戶 A 所擁有之儲存貯體中的物件來驗證許可。

在此範例中，您需要兩個帳戶。下表顯示如何參照這些帳戶與其中的管理員使用者。根據 IAM 準則 (請參閱關於使用管理員使用者來建立資源並授予許可)，我們不會在本逐步解說中使用 root 使用者登入資料。相反地，您可以在每個帳戶中建立管理員使用者，並使用這些憑證來建立資源以及將許可授予它們。

AWS 帳戶 身份證	帳戶稱為	帳戶中的管理員使用者
1111-1111-1111	帳戶 A	AccountAadmin
2222-2222-2222	帳戶 B	AccountBadmin

所有建立使用者與授予許可的任務都是在 AWS Management Console完成。若要驗證權限，逐步解說會使用命令列工具 AWS Command Line Interface (CLI) 和 AWS Tools for Windows PowerShell，因此您不需要撰寫任何程式碼。

準備演練

1. 請確定您有兩個帳戶， AWS 帳戶 而且每個帳戶都有一位系統管理員使用者，如前一節的表格所示。

   a. 如果需要 AWS 帳戶，請註冊。

   b. 使用帳戶 A 憑證來登入 IAM 主控台，建立管理員使用者：

      i. 創建用戶**AccountAadmin**並記下安全憑據。如需說明，請參閱《IAM 使用者指南》中的在 AWS 帳戶中建立 IAM 使用者。

      ii. 附加提供完整存取權AccountAadmin的使用者原則，以授予管理員權限。如需說明，請參閱《IAM 使用者指南》中的使用政策。

   c. 當您在 IAM 主控台中時，請記下儀表板上的 IAM 使用者登入 URL。登入 AWS Management Console時，帳戶中的所有使用者都必須使用此 URL。

如需詳細資訊，請參閱《IAM 使用者指南》中的使用者如何登入您的帳戶。

d. 使用帳戶 B 認證重複上述步驟，並建立系統管理員使用者**AccountBadmin**。

2. 設定 AWS Command Line Interface (AWS CLI) 或 AWS Tools for Windows PowerShell. 請確定您儲存系統管理員使用者認證，如下所示：

   - 如果使用 AWS CLI，請建立兩個設定檔AccountBadmin，AccountAadmin並在組態檔案中建立。
   - 如果使用 AWS Tools for Windows PowerShell，請確定您將工作階段的認證儲存為AccountAadmin和AccountBadmin。

   如需說明，請參閱設定逐步解說的工具。

3. 儲存管理員使用者憑證 (也稱為描述檔)。您可以使用描述檔名稱，而不是為所輸入的每個命令指定憑證。如需詳細資訊，請參閱 設定逐步解說的工具。

   a. 在每個管理員使用者的 AWS CLI 認證檔案中新增設定檔AccountBadmin，AccountAadmin並在兩個帳戶中新增設定檔。

   ```
   [AccountAadmin]
   aws_access_key_id = access-key-ID
   aws_secret_access_key = secret-access-key
   region = us-east-1

   [AccountBadmin]
   aws_access_key_id = access-key-ID
   aws_secret_access_key = secret-access-key
   region = us-east-1
   ```

   b. 如果您使用的是 AWS Tools for Windows PowerShell，請執行下列命令。

   ```
   set-awscredentials –AccessKey AcctA-access-key-ID –SecretKey AcctA-secret-
   access-key –storeas AccountAadmin
   set-awscredentials –AccessKey AcctB-access-key-ID –SecretKey AcctB-secret-
   access-key –storeas AccountBadmin
   ```

步驟 1：執行帳戶 A 任務

步驟 1.1：登入 AWS Management Console

使用帳戶 A 的 IAM 使用者登入 URL，首先以使用AccountAadmin者 AWS Management Console 身分登入。此使用者將建立儲存貯體並連接其政策。

步驟 1.2：建立儲存貯體

1. 在 Amazon S3 主控台中建立儲存貯體。本練習假設值區是在美國東部 (維吉尼亞北部) 建立 AWS 區域 並命名的*DOC-EXAMPLE-BUCKET*。

   如需說明，請參閱建立儲存貯體。
2. 將範例物件上傳至儲存貯體。

   如需取得說明，請前往 步驟 2：將物件上傳至您的儲存貯體。

步驟 1.3：連接儲存貯體政策以將跨帳戶許可授予帳戶 B

值區政策會將s3:GetLifecycleConfiguration和s3:ListBucket權限授予帳戶 B。假設您仍使用使用AccountAadmin者認證登入主控台。

1. 將下列儲存貯體政策連接至 *DOC-EXAMPLE-BUCKET*。政策會將 s3:GetLifecycleConfiguration 與 s3:ListBucket 動作的許可授予帳戶 B。

   如需說明，請參閱使用 Amazon S3 主控台新增儲存貯體政策。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:GetLifecycleConfiguration",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
```

```
            ]
        }
    ]
}
```

2. 確認帳戶 B (以及其系統管理員使用者) 可以執行作業。

   - 使用驗證 AWS CLI

     ```
     aws s3 ls s3://DOC-EXAMPLE-BUCKET --profile AccountBadmin
     aws s3api get-bucket-lifecycle-configuration --bucket DOC-EXAMPLE-BUCKET --
     profile AccountBadmin
     ```

   - 使用驗證 AWS Tools for Windows PowerShell

     ```
     get-s3object -BucketName DOC-EXAMPLE-BUCKET -StoredCredentials AccountBadmin
     get-s3bucketlifecycleconfiguration -BucketName DOC-EXAMPLE-BUCKET -
     StoredCredentials AccountBadmin
     ```

步驟 2：執行帳戶 B 任務

現在，帳戶 B 管理員會建立使用者 Dave，並委派接收自帳戶 A 的許可。

步驟 2.1：登入 AWS Management Console

使用帳戶 B 的 IAM 使用者登入 URL，首先以使用AccountBadmin者 AWS Management Console 身分登入。

步驟 2.2：在帳戶 B 中建立使用者 Dave

在 IAM 主控台中，建立使用者**Dave**。

如需說明，請參閱《IAM 使用者指南》中的建立 IAM 使用者 (主控台)。

步驟 2.3：將許可委派給使用者 Dave

使用下列政策，建立使用者 Dave 的內嵌政策。您需要提供儲存貯體名稱來更新政策。

假設您已使用使用AccountBadmin者認證登入主控台。

```
{
   "Version": "2012-10-17",
```

```
   "Statement": [
      {
         "Sid": "Example",
         "Effect": "Allow",
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
         ]
      }
   ]
}
```

如需說明，請參閱《IAM 使用者指南》中的管理 IAM 政策。

步驟 2.4：測試許可

現在，帳戶 B 中的 Dave 可以列出帳戶 A 所擁有 *DOC-EXAMPLE-BUCKET* 的內容。您可以使用下列任一個程序來驗證許可。

使用測試權限 AWS CLI

1. 將配置文UserDave件添加到 AWS CLI 配置文件中。如需組態檔的詳細資訊，請參閱「設定逐步解說的工具」。

   ```
   [profile UserDave]
   aws_access_key_id = access-key
   aws_secret_access_key = secret-access-key
   region = us-east-1
   ```

2. 在命令提示字元中，輸入下列 AWS CLI 命令以確認 Dave 現在可以從帳戶 A *DOC-EXAMPLE-BUCKET* 所擁有的物件清單中取得物件清單。請注意指定設定UserDave檔的命令。

   ```
   aws s3 ls s3://DOC-EXAMPLE-BUCKET --profile UserDave
   ```

   戴夫沒有任何其他權限。因此，如果他嘗試任何其他操作（例如，下面的get-bucket-lifecycle配置），Amazon S3 將返回被拒絕的權限。

   ```
   aws s3api get-bucket-lifecycle-configuration --bucket DOC-EXAMPLE-BUCKET --profile
    UserDave
   ```

使用測試權限 AWS Tools for Windows PowerShell

1. 將戴夫的憑據存儲為AccountBDave.

```
set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas
 AccountBDave
```

2. 嘗試列出儲存貯體命令。

```
get-s3object -BucketName DOC-EXAMPLE-BUCKET -StoredCredentials AccountBDave
```

戴夫沒有任何其他權限。因此，如果他嘗試任何其他操作（例如，以下操作），get-s3bucketlifecycleconfigurationAmazon S3 將返回被拒絕的權限。

```
get-s3bucketlifecycleconfiguration -BucketName DOC-EXAMPLE-BUCKET -
StoredCredentials AccountBDave
```

步驟 3：(選用) 嘗試明確拒絕

您可以使用存取控制清單 (ACL)、儲存貯體原則或使用者策略來授與權限。但是，如果儲存貯體原則或使用者原則有明確拒絕設定，則明確拒絕的優先順序會高於任何其他權限。若要進行測試，請更新值區政策，並明確拒絕帳戶 B 的s3:ListBucket權限。此政策也會授予s3:ListBucket權限。不過，明確拒絕優先順序，而帳戶 B 或帳戶 B 中的使用者將無法列出中的物件*DOC-EXAMPLE-BUCKET*。

1. 使用帳戶 A AccountAadmin 中使用者的認證，以下列方式取代值區政策。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:GetLifecycleConfiguration",
            "s3:ListBucket"
         ],
         "Resource": [
```

```
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
         ]
      },
      {
         "Sid": "Deny permission",
         "Effect": "Deny",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
         ]
      }
   ]
}
```

2. 現在，如果您嘗試使用AccountBadmin憑據獲取存儲桶列表，則訪問被拒絕。

   - 使用 AWS CLI，執行下列命令：

     ```
     aws s3 ls s3://DOC-EXAMPLE-BUCKET --profile AccountBadmin
     ```

   - 使用 AWS Tools for Windows PowerShell，執行下列命令：

     ```
     get-s3object -BucketName DOC-EXAMPLE-BUCKET -StoredCredentials AccountBDave
     ```

步驟 4：清理

1. 完成測試後，您可以執行以下操作來清理：

   - 使用帳戶 A 認證登入 AWS Management Console (AWS Management Console)，然後執行下列動作：

     - 在 Amazon S3 主控台中，移除連接至 *DOC-EXAMPLE-BUCKET* 的儲存貯體政策。在儲存貯體 Properties (屬性) 中，刪除 Permissions (許可) 區段中的政策。
     - 如果儲存貯體是為此練習而建立，請在 Amazon S3 主控台中刪除物件，然後刪除儲存貯體。
     - 在 IAM 主控台中，移除使AccountAadmin用者。

2. 使用帳戶 B 登入資料登入 IAM 主控台。刪除使用者AccountBadmin。如需 step-by-step 指示，請參閱 IAM 使用者指南中的刪除 IAM 使用者。

範例 3：授予對其未擁有之物件的許可的儲存貯體擁有者

> **Important**
>
> 將權限授予 IAM 角色比授予個別使用者更好。若要了解如何操作，請參閱 了解跨帳戶許可和使用 IAM 角色。

主題

- 步驟 0：準備演練
- 步驟 1：執行帳戶 A 任務
- 步驟 2：執行帳戶 B 任務
- 步驟 3：測試許可
- 步驟 4：清理

此範例的案例是值區擁有者想要授與存取物件的權限，但值區擁有者並未擁有值區中的所有物件。在此範例中，儲存貯體擁有者嘗試使用它自己的帳戶將許可授予使用者。

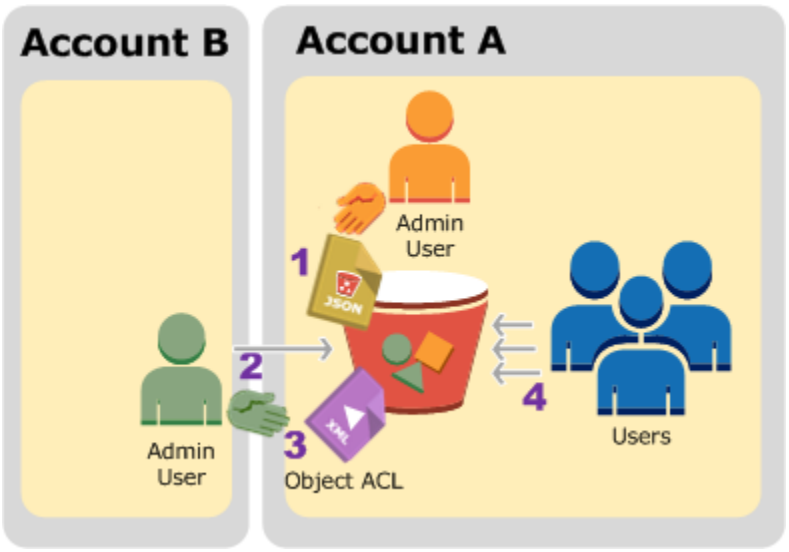
值區擁有者可以讓其他 AWS 帳戶 人上傳物件。預設情況下，儲存貯體擁有者不會擁有由另一個 AWS 帳戶寫入儲存貯體的物件。物件由將其寫入 S3 儲存貯體的帳戶擁有。如果值區擁有者不擁有值區中的物件，物件擁有者必須先使用物件存取控制清單 (ACL) 將權限授與值區擁有者。然後，值區擁有者可以將權限授與他們不擁有的物件。如需詳細資訊，請參閱 Amazon S3 儲存貯體和物件擁有權。

如果儲存貯體擁有者為儲存貯體套用儲存貯體擁有者強制執行的 S3 物件所有權設定，則儲存貯體擁有者會擁有儲存貯體中的所有物件，包括由另一個 AWS 帳戶寫入的物件。此方法可解決物件不屬於值區擁有者的問題。然後，您可以將許可委派給您自己帳戶中的使用者或其他 AWS 帳戶。

> **Note**
>
> S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來同時控制上傳至儲存貯體之物件的擁有權，以及停用或啟用 ACL。根據預設，物件擁有權設定為「儲存貯體擁有者強制執行」設定，而且所有 ACL 都會停用。停用 ACL 時，儲存貯體擁有者會擁有儲存貯體中的所有物件，並使用存取管理政策專門管理對這些物件的存取。

Amazon S3 中的大多數新式使用案例不再需要使用 ACL。建議您將 ACL 保持停用狀態，除非在異常情況下必須個別控制每個物件的存取。停用 ACL 後，您可以使用政策來控制對儲存貯體中所有物件的存取，無論是誰將物件上傳到您的儲存貯體。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

在此範例中，假設儲存貯體擁有者尚未套用儲存貯體擁有者強制執行的「物件擁有權」設定。儲存貯體擁有者向其自己帳戶中的使用者委派許可。以下是逐步解說步驟的摘要：



1. 帳戶 A 管理員使用者使用兩個陳述式來連接儲存貯體政策。
   - 允許帳戶 B 上傳物件的跨帳戶許可。
   - 允許其自己帳戶中的使用者存取儲存貯體中的物件。
2. 帳戶 B 管理員使用者將物件上傳至帳戶 A 所擁有的儲存貯體。
3. 帳戶 B 管理員會更新物件 ACL，以新增將物件完全控制許可給予儲存貯體擁有者的授予。
4. 帳戶 A 中的使用者會存取儲存貯體中的物件來進行驗證，而不管其擁有者為何。

在此範例中，您需要兩個帳戶。下表顯示如何參照這些帳戶與這些帳戶中的管理員使用者。在此演練中，您不會根據建議的 IAM 指導方針使用帳戶根使用者憑證。如需詳細資訊，請參閱 關於使用管理員使用者來建立資源並授予許可。相反地，您可以在每個帳戶中建立管理員，並使用這些憑證來建立資源以及向其授予許可。

AWS 帳戶 身份證	帳戶稱為	帳戶中的管理員
1111-1111-1111	帳戶 A	AccountAadmin
2222-2222-2222	帳戶 B	AccountBadmin

所有建立使用者與授予許可的任務都是在 AWS Management Console完成。若要驗證權限，逐步解說會使用指令行工具 AWS Command Line Interface (AWS CLI) 和 AWS Tools for Windows PowerShell，因此您不需要撰寫任何程式碼。

步驟 0：準備演練

1. 請確定您有兩個帳戶， AWS 帳戶 而且每個帳戶都有一個管理員，如前一節的表格所示。

   a. 如果需要 AWS 帳戶，請註冊。

   b. 使用帳戶 A 登入資料登入 IAM 主控台，然後執行下列動作以建立管理員使用者：

      - 建立使用者**AccountAadmin**並記下使用者的安全認證。如需有關新增使用者的詳細資訊，請參閱《IAM 使用者指南》中的在您的 AWS 帳戶中建立 IAM 使用者。
      - 附加可提供完整存AccountAadmin取權的使用者原則，以授與管理員權限。如需說明，請參閱《IAM 使用者指南》中的管理 IAM 政策。
      - 在 IAM 主控台儀表板中，記下 IAM 使用者登入 URL。登入 AWS Management Console 時，此帳戶中的使用者必須使用此 URL。如需詳細資訊，請參閱《IAM 使用者指南》中的使用者如何登入您的帳戶。

   c. 使用帳戶 B 認證重複上述步驟，並建立系統管理員使用者**AccountBadmin**。

2. 設定 AWS CLI 或視窗工具 PowerShell。請確定您如下所示儲存系統管理員憑證：

   - 如果使用 AWS CLI，請建立兩個設定檔`AccountBadmin`，`AccountAadmin`並在組態檔案中建立。
   - 如果使用 Windows 適用的工具 PowerShell，請確定您將工作階段的認證儲存為`AccountAadmin`和`AccountBadmin`。

   如需說明，請參閱設定逐步解說的工具。

步驟 1：執行帳戶 A 任務

針對帳號 A 執行以下步驟：

步驟 1.1：登入主控台

使用帳戶 A 的 IAM 使用者登入 URL，以使用**AccountAadmin**者 AWS Management Console 身分登入。此使用者將建立儲存貯體並連接其政策。

步驟 1.2：建立儲存貯體和使用者，並新增授予使用者許可的儲存貯體政策

1. 在 Amazon S3 主控台中建立儲存貯體。本練習假設值區是在美國東部 (維吉尼亞北部) 建立的 AWS 區域，名稱為*example-s3-bucket1*。

   如需說明，請參閱建立儲存貯體。
2. 在 IAM 主控台中建立使用者**Dave**。

   如需 step-by-step 指示，請參閱 IAM 使用者指南中的建立 IAM 使用者 (主控台)。
3. 請注意使用者 Dave 認證。
4. 在 Amazon S3 主控台中，將下列儲存貯體政策連接至 *example-s3-bucket1* 儲存貯體。如需說明，請參閱使用 Amazon S3 主控台新增儲存貯體政策。遵循新增儲存貯體政策的步驟。如需如何尋找帳號 ID 的詳細資訊，請參閱尋找您的 AWS 帳戶 ID。

   政策會將 `s3:PutObject` 與 `s3:ListBucket` 許可授予帳戶 B。此策略也會授與使用Dave者`s3:GetObject`權限。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Statement1",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:PutObject",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::example-s3-bucket1/*",
```

```
            "arn:aws:s3:::example-s3-bucket1"
         ]
      },
      {
         "Sid": "Statement3",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
         },
         "Action": [
            "s3:GetObject"
         ],
         "Resource": [
            "arn:aws:s3:::example-s3-bucket1/*"
         ]
      }
   ]
}
```

步驟 2：執行帳戶 B 任務

現在帳戶 B 具有對帳戶 A 儲存貯體執行作業的權限，帳戶 B 管理員會執行下列動作：

- 將物件上傳至帳戶 A 的值區
- 在物件 ACL 中新增授權，以允許帳戶 A (值區擁有者) 完全控制

使用 AWS CLI

1. 使用put-object AWS CLI 指令上載物件。命令中的 ---body 參數會識別要上傳的來源檔案。例如，如果檔案位於機器的C:磁碟Windows機上，請指定c:\HappyFace.jpg。--key 參數提供物件的金鑰名稱。

   ```
   aws s3api put-object --bucket example-s3-bucket1 --key HappyFace.jpg --body
    HappyFace.jpg --profile AccountBadmin
   ```

2. 新增物件 ACL 的授予許可，以允許儲存貯體擁有者可完全控制物件。如需如何尋找規範使用者 ID 的詳細資訊，請參閱《AWS 帳戶管理參考指南》 AWS 帳戶中的「尋找您的標準使用者 ID」。

   ```
   aws s3api put-object-acl --bucket example-s3-bucket1 --key HappyFace.jpg --grant-
   full-control id="AccountA-CanonicalUserID" --profile AccountBadmin
   ```

使用視窗工具 PowerShell

1. 使用Write-S3Object指令上載物件。

```
Write-S3Object -BucketName example-s3-bucket1 -key HappyFace.jpg -file
 HappyFace.jpg -StoredCredentials AccountBadmin
```

2. 新增物件 ACL 的授予許可，以允許儲存貯體擁有者可完全控制物件。

```
Set-S3ACL -BucketName example-s3-bucket1 -Key HappyFace.jpg -CannedACLName "bucket-
owner-full-control" -StoredCreden
```

步驟 3：測試許可

現在，驗證帳戶 A 中的使用者 Dave 可以存取帳戶 B 所擁有的物件。

使用 AWS CLI

1. 將使用者 Dave 認證新增至設 AWS CLI 定檔，並建立新的設定檔UserDaveAccountA。如需詳細資訊，請參閱 設定逐步解說的工具。

```
[profile UserDaveAccountA]
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
region = us-east-1
```

2. 執行 get-object CLI 命令來下載 HappyFace.jpg，並將其儲存在本機。您可以新增 --profile 參數，來提供使用者 Dave 憑證。

```
aws s3api get-object --bucket example-s3-bucket1 --key HappyFace.jpg Outputfile.jpg
 --profile UserDaveAccountA
```

使用視窗工具 PowerShell

1. 將使用者 Dave AWS 認證儲存為UserDaveAccountA永久性存放區。

```
Set-AWSCredentials -AccessKey UserDave-AccessKey -SecretKey UserDave-
SecretAccessKey -storeas UserDaveAccountA
```

2. 執行 Read-S3Object 命令來下載 HappyFace.jpg 物件，並將其儲存在本機。您可以新增 -StoredCredentials 參數，來提供使用者 Dave 憑證。

```
Read-S3Object -BucketName example-s3-bucket1 -Key HappyFace.jpg -file HappyFace.jpg
  -StoredCredentials UserDaveAccountA
```

步驟 4：清理

1. 完成測試後，您可以執行以下操作來清理：
   - 使用帳戶 A 憑證登入 AWS Management Console，並執行下列操作：
     - 在 Amazon S3 主控台中，移除附加至## s3-儲存貯體 1 的儲存貯體政策。在儲存貯體 Properties (屬性) 中，刪除 Permissions (許可) 區段中的政策。
     - 如果儲存貯體是為此練習而建立，請在 Amazon S3 主控台中刪除物件，然後刪除儲存貯體。
     - 在 IAM 主控台中，移除使AccountAadmin用者。如需step-by-step 指示，請參閱 IAM 使用者指南中的刪除 IAM 使用者。
2. 使用帳戶 B 憑證登入 AWS Management Console。在 IAM 主控台中，刪除使用者AccountBadmin。

範例 4-值區擁有者將跨帳戶權限授與其不擁有的物件

主題

- 了解跨帳戶許可和使用 IAM 角色
- 步驟 0：準備演練
- 步驟 1：執行帳戶 A 任務
- 步驟 2：執行帳戶 B 任務
- 步驟 3：執行帳戶 C 任務
- 步驟 4：清理
- 相關資源

在此範例案例中，您擁有值區，而且您已啟用其他 AWS 帳戶 值區上傳物件。如果您為儲存貯體套用儲存貯體擁有者強制執行的 S3 物件所有權設定，則會擁有儲存貯體中的所有物件，包括由另一個 AWS 帳戶寫入的物件。此方法可解決物件不屬於您 (值區擁有者) 的問題。然後，您可以將許可委派

給您自己帳戶中的使用者或其他 AWS 帳戶。假設未啟用儲存貯體擁有者強制執行的 S3 物件擁有權設定。亦即，您的儲存貯體可以有其他 AWS 帳戶 所擁有的物件。

現在，假設身為儲存貯體擁有者，不論誰是擁有者，您都需要將物件的跨帳戶許可授予另一個帳戶中的使用者。例如，該使用者可以是需要存取物件中繼資料的帳單應用程式。有兩個核心問題：

- 儲存貯體擁有者並不擁有其他 AWS 帳戶所建立物件的許可。值區擁有者若要授與非擁有物件的權限，物件擁有者必須先將權限授與值區擁有者。物件擁有者 AWS 帳戶 是建立物件的人。儲存貯體擁有者接著可以委派這些許可。
- 值區擁有者帳戶可以將權限委派給自己帳戶中的使用者 (請參閱範例 3：授予對其未擁有之物件的許可的儲存貯體擁有者)。但是，值區擁有者帳戶無法將權限委派給其他人， AWS 帳戶 因為不支援跨帳戶委派。

在這種情況下，值區擁有者可以建立具有存取物件權限的 AWS Identity and Access Management (IAM) 角色。然後，值區擁有者可以授予另一個 AWS 帳戶 權限來擔任該角色，暫時允許其存取值區中的物件。

> **Note**
>
> S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來同時控制上傳至儲存貯體之物件的擁有權，以及停用或啟用 ACL。根據預設，物件擁有權設定為「儲存貯體擁有者強制執行」設定，而且所有 ACL 都會停用。停用 ACL 時，儲存貯體擁有者會擁有儲存貯體中的所有物件，並使用存取管理政策專門管理對這些物件的存取。
>
> Amazon S3 中的大多數新式使用案例不再需要使用 ACL。建議您將 ACL 保持停用狀態，除非在異常情況下必須個別控制每個物件的存取。停用 ACL 後，您可以使用政策來控制對儲存貯體中所有物件的存取，無論是誰將物件上傳到您的儲存貯體。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

了解跨帳戶許可和使用 IAM 角色

IAM 角色可讓數個案例委派資源存取，而跨帳戶存取是其中一個重要案例。在此範例中，儲存貯體擁有者帳戶 A 使用 IAM 角色暫時將物件存取跨帳戶委派給另一個帳戶 C 中的使用者 AWS 帳戶，即您建立的每個 IAM 角色都附加了下列兩個政策：

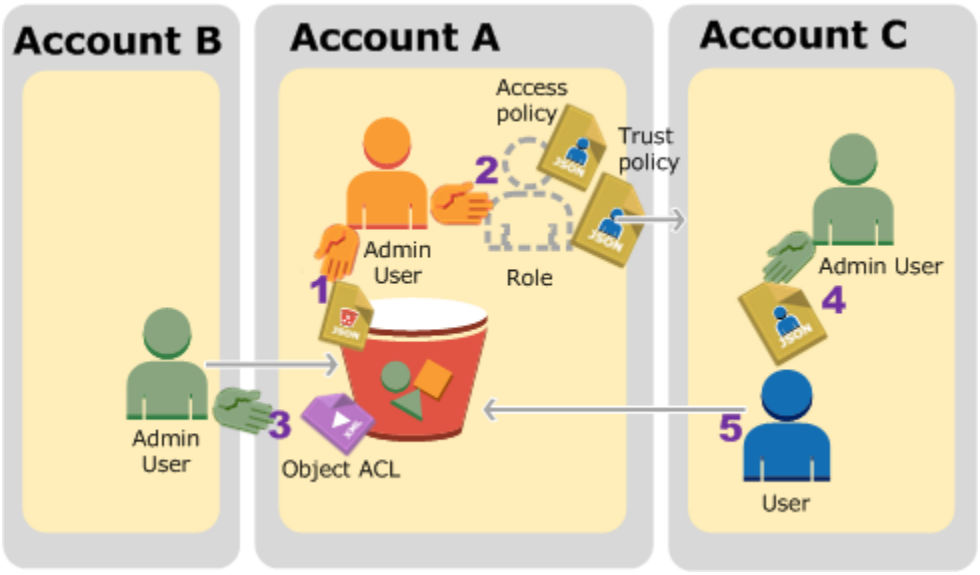
- 識別另一個可以擔任 AWS 帳戶 該角色的信任原則。
- 定義有人採用角色時允許之許可 (例如，`s3:GetObject`) 的存取政策。如需您可在政策中指定的許可清單，請參閱「Amazon S3 的政策動作」。

信任原則中所 AWS 帳戶 識別的接著會授與其使用者權限來擔任該角色。使用者接著可以執行下列操作來存取物件：

- 採用角色，並據此取得暫時性安全憑證。
- 使用暫時性安全憑證，存取儲存貯體中的物件。

如需 IAM 角色的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 角色。

以下是逐步解說步驟的摘要：



1. 帳戶 A 管理員使用者會連接儲存貯體政策，以將上傳物件的條件式許可授予帳戶 B。
2. 帳戶 A 管理員會建立 IAM 角色，以建立與帳戶 C 的信任，因此該帳戶中的使用者可以存取帳戶 A。連接至角色的存取政策會限制帳戶 C 中的使用者在使用者存取帳戶 A 時可以執行的操作。
3. 帳戶 B 管理員會讓物件上傳至帳戶 A 所擁有的儲存貯體，以將完全控制許可授予儲存貯體擁有者。
4. 帳戶 C 管理員會建立使用者，並連接允許使用者採用角色的使用者政策。
5. 帳戶 C 中的使用者會先採用角色，以傳回使用者暫時性安全憑證。使用這些暫時性憑證，使用者則會存取儲存貯體中的物件。

在此範例中，您需要三個帳戶。下表顯示如何參照這些帳戶與這些帳戶中的管理員使用者。根據 IAM 準則 (請參閱關於使用管理員使用者來建立資源並授予許可)，我們不會在本逐步解說中使用 AWS 帳戶根使用者 認證。相反地，您可以在每個帳戶中建立管理員使用者，並使用這些憑證來建立資源以及將許可授予它們。

AWS 帳戶 身份證	帳戶稱為	帳戶中的管理員使用者
1111-1111-1111	帳戶 A	AccountAadmin
2222-2222-2222	帳戶 B	AccountBadmin
3333-3333-3333	帳戶 C	AccountCadmin

步驟 0：準備演練

> **Note**
>
> 您可能想要開啟文字編輯器，並在執行這些步驟時寫下部分資訊。特別的是，您需要每個帳戶的帳戶 ID、正式使用者 ID、IAM 使用者登入 URL 才能連接至主控台，以及 IAM 使用者和角色的 Amazon Resource Names (ARN)。

1. 請確定您有三個帳戶， AWS 帳戶 而且每個帳戶都有一位系統管理員使用者，如前一節的表格所示。

   a. 根據需要註冊。 AWS 帳戶我們的這些帳戶指的是帳戶 A、帳戶 B 和帳戶 C。

   b. 使用帳戶 A 憑證來登入 IAM 主控台，然後執行下列操作以建立管理員使用者：

      - 創建用戶**AccountAadmin**並記下其安全憑據。如需有關新增使用者的詳細資訊，請參閱《IAM 使用者指南》中的在您的 AWS 帳戶中建立 IAM 使用者。
      - 附加提供完整存取權AccountAadmin的使用者原則，以授予管理員權限。如需說明，請參閱《IAM 使用者指南》中的管理 IAM 政策。
      - 在 IAM 主控台儀表板中，記下 IAM 使用者登入 URL。登入 AWS Management Console 時，此帳戶中的使用者必須使用此 URL。AWS Management Console 如需詳細資訊，請參閱 IAM 使用者指南中的以 IAM 使用者身分登入。

   c. 重複前一個步驟，以在帳戶 B 和帳戶 C 中建立管理員使用者。

2. 對於帳戶 C，請記下規範的使用者 ID。

   當您在帳戶 A 中建立 IAM 角色時，信任政策透過指定帳戶 ID，以將採用角色的許可授予帳戶 C。您可以如下尋找帳戶資訊：

a. 使用您的 AWS 帳戶 ID 或帳戶別名、IAM 使用者名稱和密碼登入 Amazon S3 主控台。

b. 選擇 Amazon S3 儲存貯體的名稱，以檢視該儲存貯體的詳細資訊。

c. 選擇 Permissions (許可) 標籤，然後選擇 Access Control List (存取控制清單)。

d. 在 Access for your AWS 帳戶 account (存取您的 AWS 帳戶帳戶) 區段的 Account (帳戶) 欄中是長識別碼，例如 `c1daexampleaaf850ea79cf0430f33d72579fd1611c97f7ded193374c0b163b6`。這是您的正式使用者 ID。

3. 建立儲存貯體政策時，您需要下列資訊。請注意這些值：

- 帳戶 A 的正式使用者 ID – 當帳戶 A 管理員將條件式上傳物件許可授予帳戶 B 管理員時，針對必須完全控制物件的帳戶 A 使用者，此條件指定其正式使用者 ID。

> **Note**
>
> 正式使用者 ID 是 Amazon S3 才有的概念。它是帳戶 ID 的 64 字元混淆版本。

- 帳戶 B 管理員的使用者 ARN — 您可以在 IAM 主控台中找到使用者 ARN。您必須選取使用者，並在「摘要」索引標籤中找到使用者的 ARN。

  在儲存貯體原則中，您授與上傳物件的`AccountBadmin`權限，並使用 ARN 指定使用者。以下是 ARN 值範例：

```
arn:aws:iam::AccountB-ID:user/AccountBadmin
```

4. 設定 AWS Command Line Interface (CLI) 或 AWS Tools for Windows PowerShell. 請確定您儲存系統管理員使用者認證，如下所示：

- 如果使用 AWS CLI，請在配置文件中創建配置文件`AccountAadmin`和`AccountBadmin`。
- 如果使用 AWS Tools for Windows PowerShell，請確定您將工作階段的認證儲存為`AccountAadmin`和`AccountBadmin`。

如需說明，請參閱設定逐步解說的工具。

步驟 1：執行帳戶 A 任務

在此範例中，帳戶 A 是儲存貯體擁有者。因此，帳戶 A AccountAadmin 中的用戶將執行以下操作：

- 建立儲存貯體。
- 附加值區政策，授與帳戶 B 管理員上載物件的權限。
- 建立 IAM 角色，授予帳戶 C 權限來擔任該角色，以便其可存取值區中的物件。

步驟 1.1：登入 AWS Management Console

使用帳戶 A 的 IAM 使用者登入 URL，首先以使用**AccountAadmin**者 AWS Management Console 身分登入。此使用者將建立儲存貯體並連接其政策。

步驟 1.2：建立儲存貯體並連接儲存貯體政策

在 Amazon S3 主控台中，執行下列操作：

1. 建立儲存貯體。此練習採用儲存貯體名稱 *example-s3-bucket1*。

   如需說明，請參閱建立儲存貯體。

2. 附加下列值區政策。此原則會將條件式權限授與帳戶 B 管理員上載物件的權限。

   透過為*example-s3-bucket1*、*AccountB-ID*和提供您自己的值來更新原則*CanonicalUserId-of-AWSaccountA-BucketOwner*。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "111",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::example-s3-bucket1/*"
        },
        {
            "Sid": "112",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::example-s3-bucket1/*",
```

```
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=CanonicalUserId-of-
AWSaccountA-BucketOwner"
                }
            }
        }
    ]
}
```

步驟 1.3：建立 IAM 角色以允許帳戶 A 中的帳戶 C 跨帳戶存取

在 IAM 主控台中，建立 IAM 角色 (**examplerole**) 以授與帳戶 C 擔任該角色的權限。請確定您仍以帳戶 A 系統管理員身分登入，因為必須在帳戶 A 中建立角色。

1. 建立角色之前，請準備定義角色所需許可的受管政策。您可以在稍後的步驟將此政策連接至角色。

   a. 在左側的導覽窗格中，選擇 [原則]，然後選擇 [建立原則]。

   b. 在建立您自己的政策旁邊，選擇選取。

   c. 在 Policy Name (政策名稱) 欄位中，輸入 **access-accountA-bucket**。

   d. 複製下列存取政策，並將其貼入 Policy Document (政策文件) 欄位。存取原則會授與角色`s3:GetObject`權限，因此當帳戶 C 使用者擔任該角色時，它只能執行`s3:GetObject`作業。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::example-s3-bucket1/*"
    }
  ]
}
```

   e. 選擇建立政策。

      新的政策會出現在受管政策清單中。

2. 在左側的導覽窗格中，選擇 [角色]，然後選擇 [建立新角色]。

3. 在 [選取角色類型] 下，選取 [跨帳戶存取的角色]，然後選擇 [提供 AWS 帳戶 您自己的存取權限] 旁邊的 [選取] 按鈕。
4. 輸入帳戶 C 帳戶 ID。

   在本逐步解說中，您不需要使用者擁有多重要素驗證 (MFA) 即可擔任角色，因此請不要選取該選項。
5. 選擇 [下一步] 以設定將與角色相關聯的權限。
6. 選取您建立的存取帳戶-儲存貯體原則旁邊的核取方塊，然後選擇 [下一步]。

   即會出現 Review (檢閱) 頁面，讓您可以在建立角色之前確認角色的設定。此頁面上要注意的一個極重要項目是可傳送給需要使用此角色之使用者的連結。使用此連結的使用者會直接前往 [切換角色] 頁面，其中 [帳戶 ID] 和 [角色名稱] 欄位已填入。您也可以稍後在任何跨帳戶角色的「角色摘要」頁面上看到此連結。
7. 輸入`examplerole`角色名稱，然後選擇 [下一步]。
8. 複查角色之後，請選擇建立角色。

   `examplerole` 角色即會顯示在角色清單中。
9. 選擇角色名稱`examplerole`。
10. 選取 Trust Relationships (信任關係) 標籤。
11. 選擇 [顯示原則文件]，然後確認顯示的信任原則符合下列原則。

    下列信任政策會建立與帳戶 C 的信任，方法是讓它執行 `sts:AssumeRole` 動作。如需詳細資訊，請參閱 AWS Security Token Service API 參考中的 AssumeRole。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountC-ID:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

12. 請記下您建立的examplerole角色的 Amazon 資源名稱 (ARN)。

    稍後在下列步驟中，您會連接使用者政策以允許 IAM 使用者採用此角色，並且依 ARN 值來識別角色。

步驟 2：執行帳戶 B 任務

帳戶 A 所擁有的範例儲存貯體需要其他帳戶所擁有的物件。在此步驟中，帳戶 B 管理員會使用命令列工具來上傳物件。

- 使用指put-object AWS CLI 令，將物件上載至*example-s3-bucket1*。

  ```
  aws s3api put-object --bucket example-s3-bucket1 --key HappyFace.jpg --
  body HappyFace.jpg --grant-full-control id="canonicalUserId-ofTheBucketOwner" --
  profile AccountBadmin
  ```

  注意下列事項：

  - --Profile參數會指定AccountBadmin設定檔，因此物件由帳戶 B 所擁有。
  - grant-full-control 參數會根據儲存貯體政策要求，將物件的完全控制許可授予儲存貯體擁有者。
  - --body 參數會識別要上傳的來源檔案。例如，如果檔案位於Windows電腦的 C: 磁碟機上，您可以指定c:\HappyFace.jpg。

步驟 3：執行帳戶 C 任務

在上述步驟中，帳戶 A 已建立角色examplerole，並使用帳戶 C 建立信任。此角色可讓帳戶 C 中的使用者存取帳戶 A。在此步驟中，帳戶 C 管理員會建立使用者 (Dave) 並委派他從帳戶 A 收到的sts:AssumeRole權限。此方法可讓 Dave 假設帳戶 A 附加至角色的存取權限，特別是 Dave 在他取得物件時可以存取帳戶 A。examplerole *example-s3-bucket1*

步驟 3.1：在帳戶 C 中創建用戶並委派假設權限 examplerole

1. 使用帳戶 C 的 IAM 使用者登入 URL，首先以使用**AccountCadmin**者 AWS Management Console 身分登入。

2. 在 IAM 主控台中，建立使用者 Dave。

如需 step-by-step 指示，請參閱 IAM 使用者指南中的建立 IAM 使用者 (AWS Management Console)。

3. 請注意戴夫認證。Dave 需要這些憑證，才能採用 examplerole 角色。
4. 為 Dave IAM 使用者建立內嵌政策，以將帳戶 A 中examplerole角色的sts:AssumeRole權限委派給 Dave。

   a. 在左側導覽窗格中，選擇 Users (使用者)。

   b. 選擇使用者名稱戴夫。

   c. 在使用者詳細資訊頁面上，選取 Permissions (許可) 標籤，然後展開 Inline Policies (內嵌政策) 區段。

   d. 選擇 click here (按一下這裡) (或 Create User Policy (建立使用者政策))。

   e. 選擇 Custom Policy (自訂政策)，然後選擇 Select (選取)。

   f. 在 Policy Name (政策名稱) 欄位中輸入政策的名稱。

   g. 將下列政策複製到 Policy Document (政策文件) 欄位。

      您必須提供*AccountA-ID*.

      ```
      {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Action": ["sts:AssumeRole"],
            "Resource": "arn:aws:iam::AccountA-ID:role/examplerole"
          }
        ]
      }
      ```

   h. 選擇 Apply Policy (套用政策)

5. 透過新增另一個設定檔，將 Dave 的 AWS CLI 認證儲存到的組態檔AccountCDave。

   ```
   [profile AccountCDave]
   aws_access_key_id = UserDaveAccessKeyID
   aws_secret_access_key = UserDaveSecretAccessKey
   region = us-west-2
   ```

步驟 3.2：假設角色（examplerole）和訪問對象

現在，Dave 可以存取帳戶 A 所擁有之儲存貯體中的物件，如下所示：

- Dave 先使用他自己的憑證來採用 examplerole。這會傳回暫時性憑證。
- 使用暫時性憑證，Dave 接著將存取帳戶 A 之儲存貯體中的物件。

1. 在命令提示字元中，使用AccountCDave設定檔執行下列 AWS CLI assume-role命令。

   您必須透過提供定義的*AccountA-ID*位examplerole置來更新指令中的 ARN 值。

   ```
   aws sts assume-role --role-arn arn:aws:iam::AccountA-ID:role/examplerole --profile
    AccountCDave --role-session-name test
   ```

   作為響應， AWS Security Token Service （AWS STS）返回臨時安全憑據（訪問密鑰 ID，秘密訪問密鑰和會話令牌）。

2. 將臨時安全登入資料儲存在設 AWS CLI 定檔下的TempCred設定檔中。

   ```
   [profile TempCred]
   aws_access_key_id = temp-access-key-ID
   aws_secret_access_key = temp-secret-access-key
   aws_session_token = session-token
   region = us-west-2
   ```

3. 在命令提示字元中，執行下列 AWS CLI 命令，以使用暫時認證存取物件。例如，此命令指定 head-object API 來擷取 HappyFace.jpg 物件的物件中繼資料。

   ```
   aws s3api get-object --bucket example-s3-bucket1 --key HappyFace.jpg SaveFileAs.jpg
    --profile TempCred
   ```

   因為連接至 examplerole 的存取政策允許這些動作，所以 Amazon S3 會處理要求。您可以嘗試對儲存貯體中的其他任何物件執行其他任何動作。

   如果您嘗試任何其他動作（例如，），get-object-acl您將獲得拒絕權限，因為該角色不允許該操作。

   ```
   aws s3api get-object-acl --bucket example-s3-bucket1 --key HappyFace.jpg --profile
    TempCred
   ```

我們已使用使用者 Dave 來採用角色，以及使用暫時性憑證來存取物件。它也可以是帳戶 C 中存取 *example-s3-bucket1* 中物件的應用程式。應用程式可以取得暫時性安全憑證，而且帳戶 C 可以將採用 examplerole 的許可委派給應用程式。

步驟 4：清理

1. 完成測試後，您可以執行以下操作來清理：
   - 使用帳戶 A 憑證登入 AWS Management Console，並執行下列操作：
     - 在 Amazon S3 主控台中，移除連接至 *example-s3-bucket1* 的儲存貯體政策。在儲存貯體 Properties (屬性) 中，刪除 Permissions (許可) 區段中的政策。
     - 如果儲存貯體是為此練習而建立，請在 Amazon S3 主控台中刪除物件，然後刪除儲存貯體。
     - 在 IAM 主控台中，移除examplerole您在帳戶 A 中建立的項目。如需 step-by-step 指示，請參閱 IAM 使用者指南中的刪除 IAM 使用者。
     - 在 IAM 主控台中，移除使AccountAadmin用者。
2. 使用帳戶 B 登入資料登入 IAM 主控台。刪除使用者AccountBadmin。
3. 使用帳戶 C 登入資料登入 IAM 主控台。刪除AccountCadmin和用戶戴夫。

相關資源

如需與本逐步解說相關的詳細資訊，請參閱 IAM 使用者指南中的下列資源：

- 建立角色以將許可委派給 IAM 使用者
- 教學課程： AWS 帳戶 使用 IAM 角色委派存取權
- 管理 IAM 政策

Amazon S3 如何授權要求

當 Amazon S3 收到要求時 (例如儲存貯體或物件操作)，它會先確認要求者具備必要的許可。Amazon S3 會在決定是否授權請求時，評估所有相關的存取政策、使用者政策和資源型政策 (儲存貯體政策、儲存貯體存取控制清單 (ACL) 和物件 ACL)。

> **Note**
>
> 如果 Amazon S3 權限檢查找不到有效的許可，則會傳回拒絕存取 (403 禁止) 權限拒絕錯誤。如需詳細資訊，請參閱疑難排解 Amazon S3 中的拒絕存取 (403 禁止) 錯誤。

為了確定請求者是否有執行特定操作的許可，Amazon S3 在收到請求時依序執行以下操作：

1. 在執行階段將所有相關的存取原則 (使用者原則、儲存貯體原則和 ACL) 轉換為一組用於評估的原則。
2. 執行下列步驟以評估這組產生的政策。在每個步驟中，Amazon S3 會根據內容授權單位，在特定內容中評估政策子集。
   a. 使用者內容 – 在使用者內容中，使用者所屬的父帳戶即為內容授權單位。

      Amazon S3 會評估父帳戶所擁有的政策子集。此子集包含父帳戶連接至使用者的使用者政策。如果父項也擁有請求 (儲存貯體或物件) 中的資源，Amazon S3 也會同時評估對應的資源政策 (儲存貯體政策、儲存貯體 ACL 和物件 ACL)。

      使用者必須具備父帳戶的執行操作許可。

      只有在請求是由 AWS 帳戶使用者提出的情況下，才需要執行此步驟。如果使用的根使用者登入資料提出請求 AWS 帳戶，Amazon S3 會略過此步驟。

   b. 儲存貯體內容 — 在儲存貯體環境中，Amazon S3 會評估擁 AWS 帳戶 有儲存貯體的政策。

      如果要求是針對儲存貯體操作，要求者必須具備儲存貯體擁有者的許可。如果要求是針對物件，Amazon S3 會評估儲存貯體擁有者所擁有的全部政策，檢查儲存貯體擁有者是否未明確拒絕對物件的存取。如已設定明確拒絕，Amazon S3 就不會授權要求。

   c. 物件內容 – 如果請求是針對物件，Amazon S3 會評估物件擁有者所擁有的政策子集。

以下是說明 Amazon S3 如何授權請求的一些範例案例。

Example — 請求者是 IAM 主體

如果請求者是 IAM 主體，Amazon S3 必須判斷主體所屬的父系 AWS 帳戶 是否已授與主體執行作業所需的必要許可。此外，如果要求是針對儲存貯體操作 (例如要求列出儲存貯體內容)，Amazon S3 必須確認儲存貯體擁有者已授予要求者執行操作的許可。若要對資源執行特定作業，IAM 主體需要其所屬父項 AWS 帳戶 和擁有 AWS 帳戶 該資源的父項的權限。

Example — 請求者是 IAM 主體 — 如果請求是針對值區擁有者不擁有的物件上的作業

如果請求是針對儲存貯體擁有者不擁有的物件進行操作，除了確保請求者具有物件擁有者的許可外，Amazon S3 還必須檢查儲存貯體政策，以確保儲存貯體擁有者未對物件設定明確拒絕。儲存貯體擁有者 (付費者) 可以明確拒絕對儲存貯體中物件的存取，無論誰是其擁有者皆是如此。儲存貯體擁有者也可刪除儲存貯體中的任何物件。

根據預設，當其他人將物件 AWS 帳戶 上傳到您的 S3 儲存貯體時，該帳戶 (物件寫入器) 擁有該物件、可以存取該物件，並且可以透過存取控制清單 (ACL) 授與其他使用者存取該物件。您可以使用「物件擁有權」變更此預設行為，以便停用 ACL，並且您身為儲存貯體擁有者，自動擁有儲存貯體中的每個物件。因此，資料的存取控制基於政策，例如 IAM 使用者政策、S3 儲存貯體政策、虛擬私有雲端 (VPC) 端點政策和 AWS Organizations 服務控制政策 (SCP)。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

如需 Amazon S3 如何評估存取原則以授權或拒絕儲存貯體操作和物件操作要求的詳細資訊，請參閱下列主題：

主題

- Amazon S3 如何授權儲存貯體操作要求
- Amazon S3 如何授權物件操作要求

Amazon S3 如何授權儲存貯體操作要求

當 Amazon S3 收到儲存貯體操作的要求時，Amazon S3 會將所有相關許可轉換為一組政策，以便在執行階段進行評估。相關許可包含以資源為基礎的許可 (例如儲存貯體政策和儲存貯體存取控制清單) 和使用者政策 (如果要求來自 IAM 委託人)。然後，Amazon S3 會根據特定的上下文 (使用者內容或儲存貯體內容)，以一系列步驟評估產生的政策集：

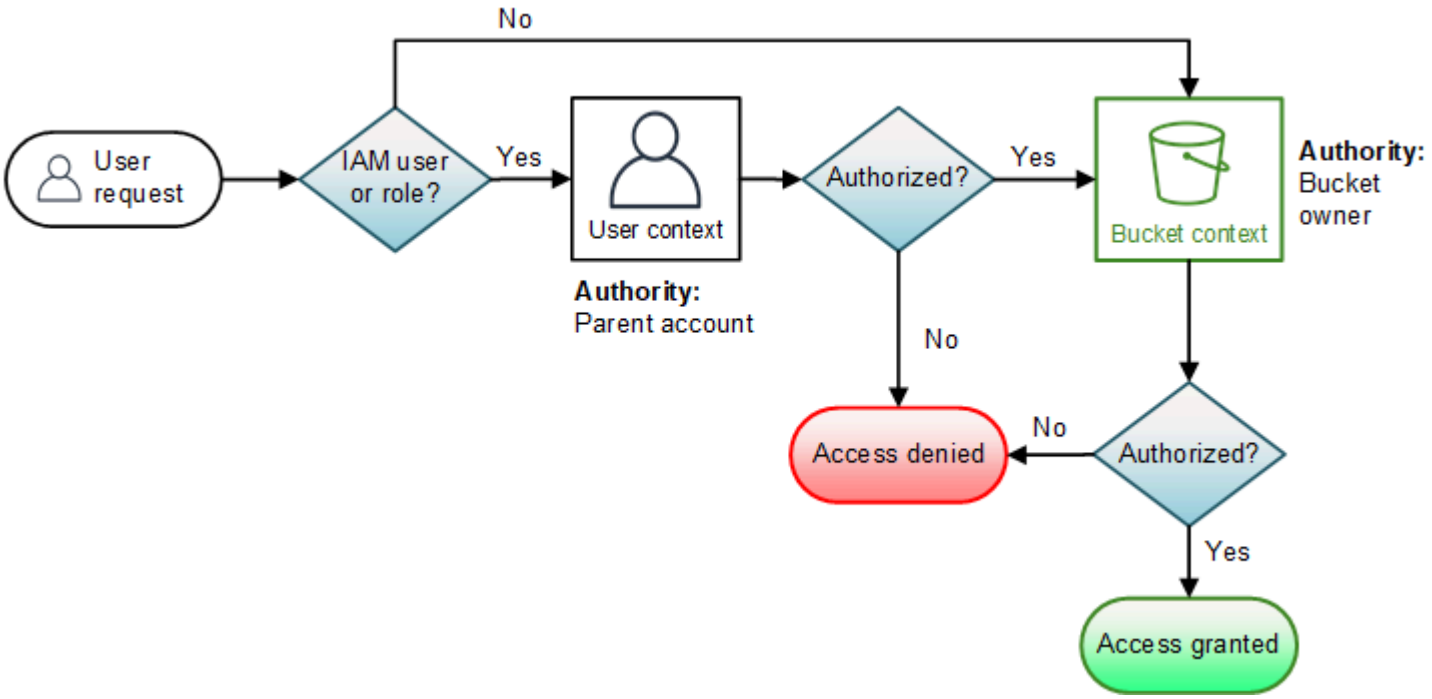
1. 使用者內容 — 如果請求者是 IAM 主體，則主體必須具有其所屬父項 AWS 帳戶 的權限。在此步驟中，Amazon S3 會評估父帳戶 (亦稱為內容授權單位) 所擁有的政策子集。此政策子集包含父帳戶連接至委託人的使用者政策。如果父帳戶也擁有要求中的資源 (在本例中為儲存貯體)，Amazon S3 也

會同時評估對應的資源政策 (儲存貯體政策與儲存貯體 ACL)。每當提出儲存貯體操作要求時，伺服器存取日誌都會記錄要求者的正式 ID。如需詳細資訊，請參閱 使用伺服器存取記錄記錄要求。

2. 儲存貯體內容 – 請求者必須取得儲存貯體擁有者的許可，才能執行特定的儲存貯體操作。在此步驟中，Amazon S3 會評估擁有儲存貯體的政策子集。 AWS 帳戶

   儲存貯體擁有者可以使用儲存貯體政策或儲存貯體 ACL 來授予許可。如果擁 AWS 帳戶 有儲存貯體的也是 IAM 主體的父帳戶，則可以在使用者政策中設定值區許可。

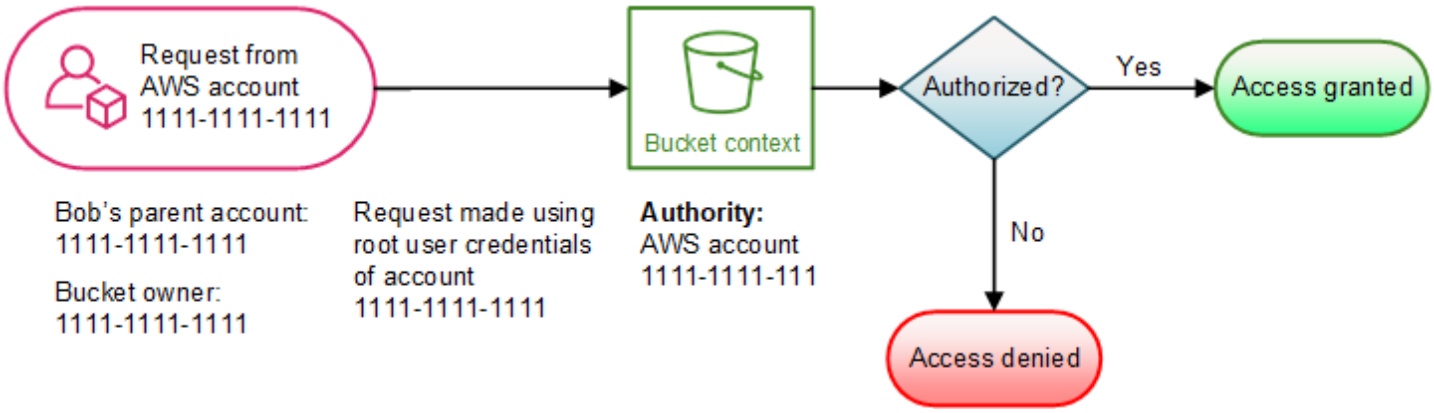
下圖說明儲存貯體操作的內容評估。



下列範例說明評估邏輯。

範例 1：由儲存貯體擁有者所要求的儲存貯體操作

在此範例中，儲存貯體擁有者使用 AWS 帳戶的根憑證，傳送儲存貯體操作請求。

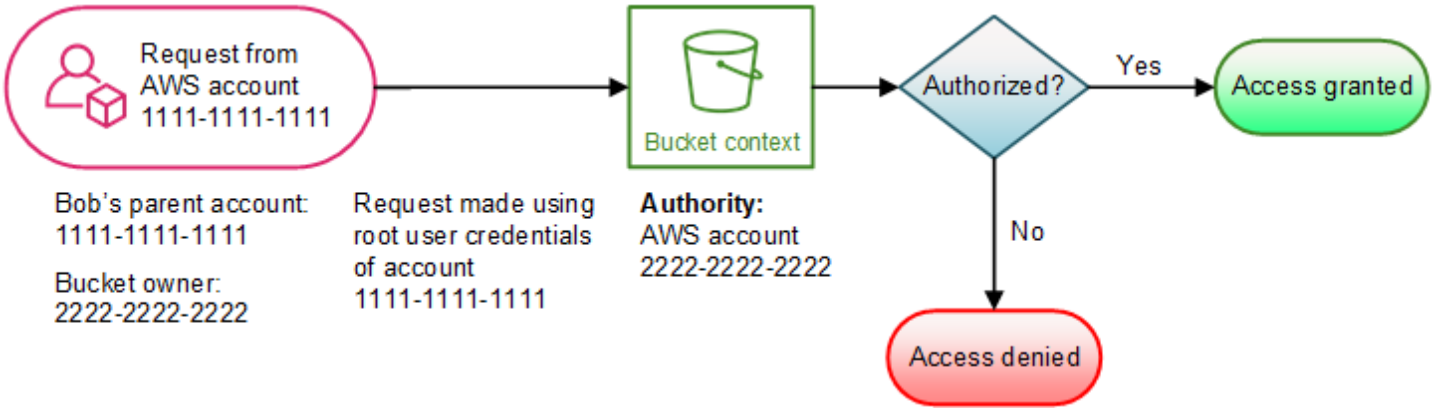


Amazon S3 會執行下列內容評估：

1. 由於請求是透過 AWS 帳戶的根使用者憑證提出，因此不會評估使用者內容。
2. 在儲存貯體內容中，Amazon S3 會檢閱儲存貯體政策，判斷要求者是否具備執行操作的許可。然後 Amazon S3 會授權要求。

範例 2：非值區擁有者所要求的值區作業 AWS 帳戶

在此範例中，使用 AWS 帳戶 1111-1111-1111 的根使用者憑證提出了一個儲存貯體操作請求，但儲存貯體擁有者為 AWS 帳戶 2222-2222-2222。此要求未涉及任何 IAM 使用者。

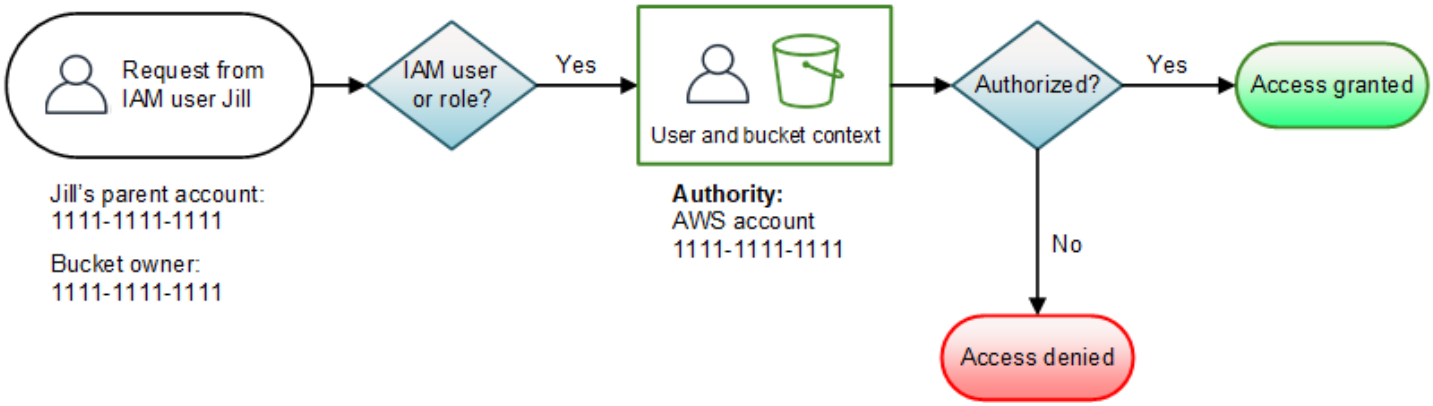


在此範例中，Amazon S3 會評估上下文，如下所示：

1. 由於要求是使用的根使用者認證來提出 AWS 帳戶，因此不會評估使用者前後關聯。
2. 在儲存貯體內容中，Amazon S3 會檢查儲存貯體政策。如果儲存貯體擁有者 (AWS 帳戶 2222-2222-2222) 未授權 AWS 帳戶 1111-1111 執行要求的作業，Amazon S3 會拒絕該要求。否則，Amazon S3 會授權要求並執行操作。

範例 3：IAM 主體要求的儲存貯體作業，其父項也 AWS 帳戶 是值區擁有者

在此範例中，請求的傳送者 Jill 是 AWS 帳戶 1111-1111-1111 的 IAM 使用者，該帳戶同時也是儲存貯體擁有者。



Amazon S3 會執行下列內容評估：

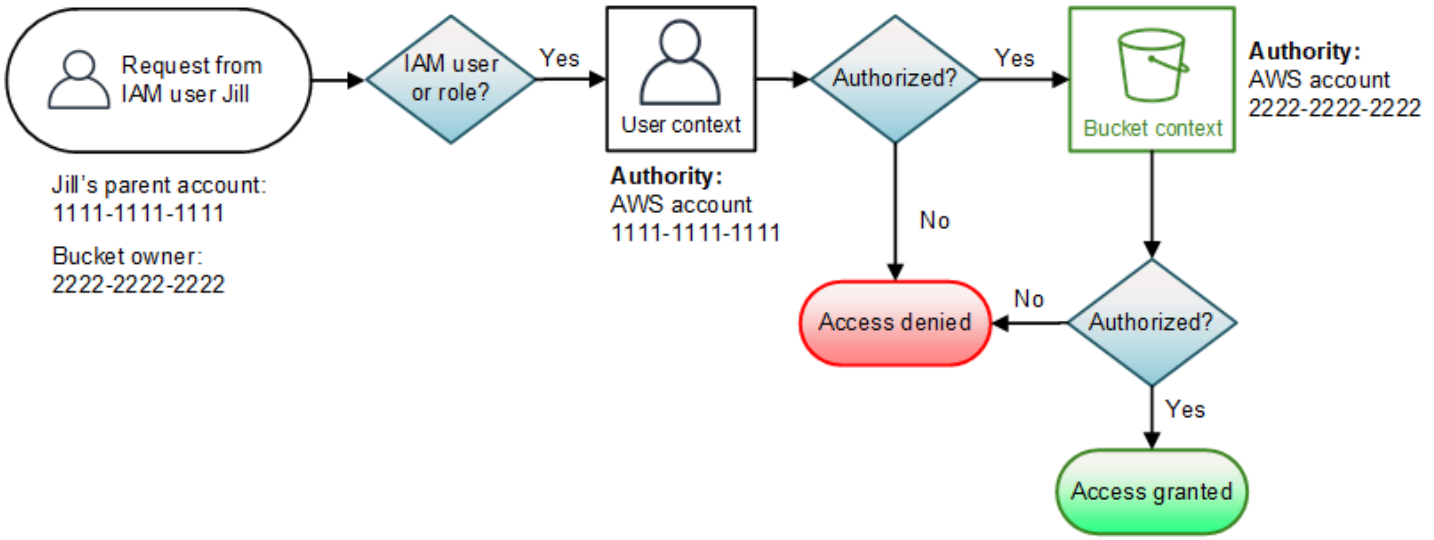
1. 由於請求來自 IAM 主體，因此在使用者內容中，Amazon S3 會評估屬於父項的所有政策， AWS 帳戶 以判斷 Jill 是否有執行操作的權限。

   在此範例中，主體所屬的父項 AWS 帳戶 1111-1111-1111 也是值區擁有者。因此，除了使用者政策之外，Amazon S3 還會在相同內容中評估儲存貯體政策和儲存貯體 ACL，因為它們屬於同一個帳戶。

2. 由於 Amazon S3 在評估使用者內容的過程中已評估儲存貯體政策與儲存貯體 ACL，因此不會評估儲存貯體內容。

範例 4：由父項不 AWS 帳戶 是值區擁有者的 IAM 主體要求的儲存貯體作業

在此範例中，請求是由身分與存取權管理系統的身分與存取權管理使用者，其父項 AWS 帳戶 為 1111-1111，但儲存貯體是由另一個 2222-2222-2222 所擁有的。 AWS 帳戶



Jill 將需要父項 AWS 帳戶 和值區擁有者的權限。Amazon S3 評估內容的方式如下：

1. 由於要求是來自 IAM 委託人，因此 Amazon S3 會檢閱帳戶所撰寫的政策以確認 Jill 具備必要的許可，藉此評估使用者內容。如果 Jill 擁有許可，則 Amazon S3 會繼續評估儲存貯體內容。如果 Jill 沒有權限，它會拒絕該請求。
2. 在儲存貯體環境中，Amazon S3 會驗證儲存貯體擁有者 2222-2222-2222 是否已授與 Jill (或她的父輩 AWS 帳戶) 權限來執行所要求的作業。如果她具有該權限，Amazon S3 會授予請求並執行操作。否則，Amazon S3 會拒絕要求。

Amazon S3 如何授權物件操作要求

Amazon S3 收到對物件操作的請求時，就會將所有相關許可，包括以資源為基礎的許可 (物件存取控制清單 (ACL)、儲存貯體政策、儲存貯體 ACL) 及 IAM 使用者政策，轉換成一組要在執行時間評估的政策。然後，它會執行一連串的步驟以評估這組產生的政策。在每個步驟中，它會評估三個特定前後關聯 (使用者前後關聯、值區前後關聯和物件前後關聯) 中的政策子集：

1. 使用者內容 — 如果請求者是 IAM 主體，則主體必須具有其所屬父項 AWS 帳戶 的權限。在此步驟中，Amazon S3 會評估父帳戶 (亦稱為內容授權單位) 所擁有的政策子集。此政策子集包含父帳戶連接至委託人的使用者政策。如果父項也擁有請求 (儲存貯體或物件) 中的資源，Amazon S3 會同時評估對應的資源政策 (儲存貯體政策、儲存貯體 ACL 和物件 ACL)。

> **Note**
>
> 如果父項 AWS 帳戶 擁有資源 (值區或物件)，則可以使用使用者政策或資源政策，將資源許可授與其 IAM 主體。

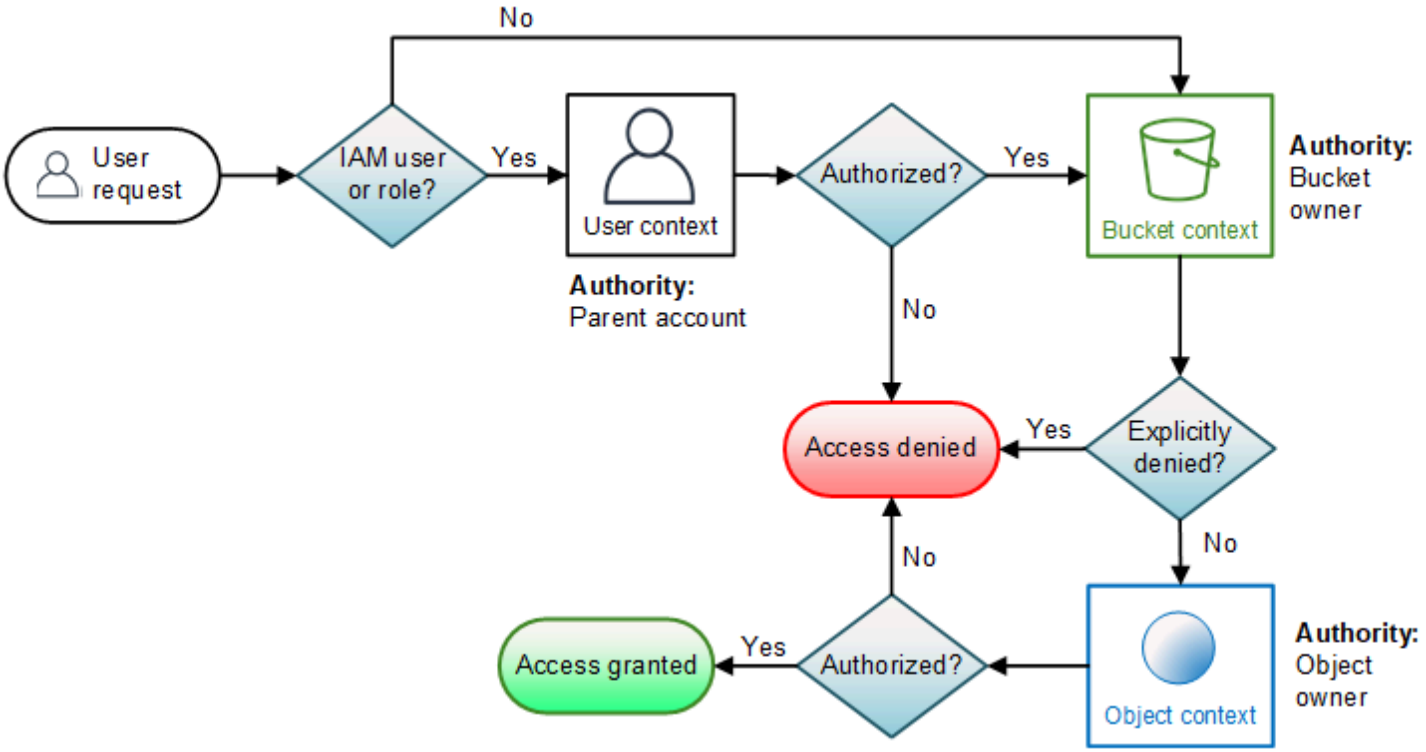
2. 儲存貯體內容 – 在此內容中，Amazon S3 會評估 AWS 帳戶 (儲存貯體擁有者) 所擁有的政策。

   如果擁 AWS 帳戶 有請求中物件的物件與儲存貯體擁有者不同，Amazon S3 會檢查儲存貯體擁有者是否明確拒絕存取物件的政策。如已在物件上設定明確拒絕，Amazon S3 就不會授權要求。

3. 物件內容 – 請求者必須取得物件擁有者的許可，才能執行特定的物件操作。在此步驟中，Amazon S3 會評估物件 ACL。

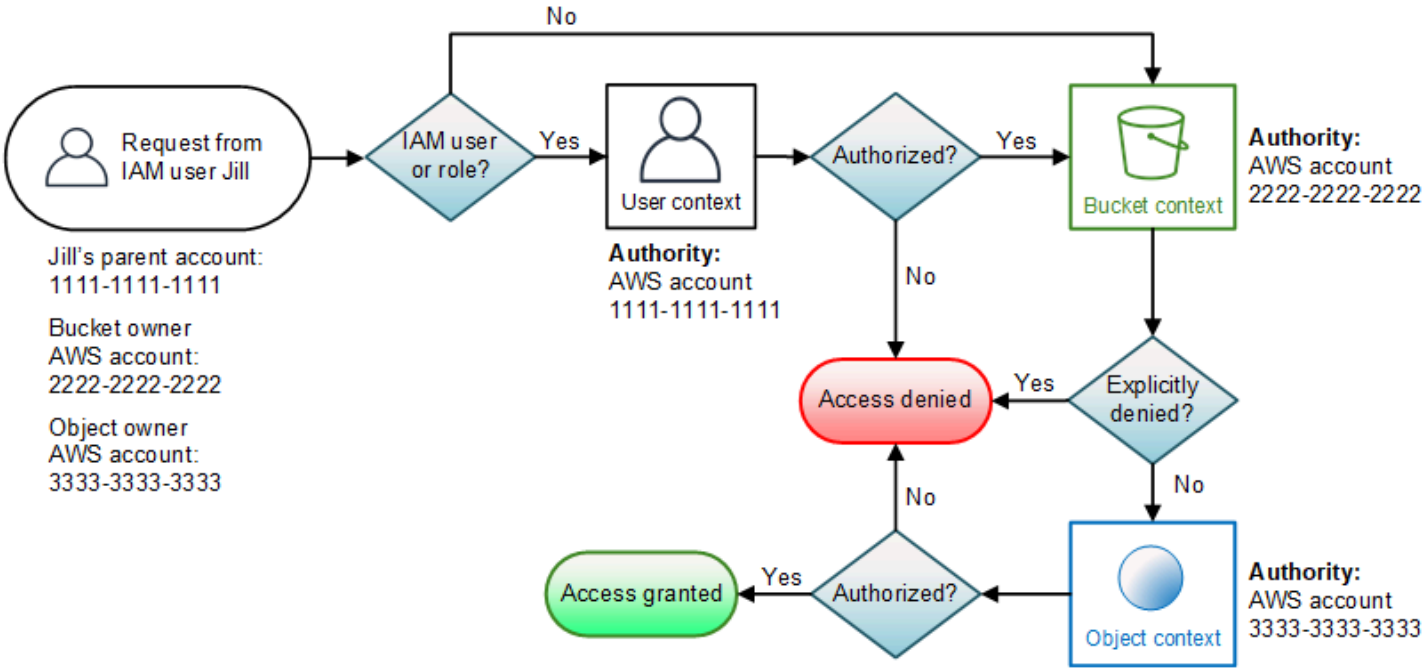
> **Note**
>
> 如果儲存貯體擁有者與物件擁有者相同，則可以在儲存貯體政策中授予對物件的存取，並在儲存貯體內容中進行評估。如果擁有者不同，物件擁有者必須使用物件 ACL 來授予許可。如果擁有 AWS 帳戶 該物件的物件也是 IAM 主體所屬的父項帳戶，則可以在使用者政策中設定物件許可，該政策會根據使用者內容進行評估。如需使用這些存取政策替代方案的詳細資訊，請參閱「使用政策管理 Amazon S3 資源存取權的逐步解說」。
>
> 如果您身為值區擁有者想要擁有值區中的所有物件，並使用以 IAM 為基礎的值區政策或政策若要管理這些物件的存取權，您可以針對物件擁有權套用值區擁有者強制執行設定。使用此設定，您身為儲存貯體擁有者會自動擁有並完全控制儲存貯體中的每個物件。無法編輯儲存貯體和物件 ACL，也不再考慮存取。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

下圖說明物件操作的內容評估。



對象操作請求的示例

在此範例中，IAM 使用者吉爾 (其父項 AWS 帳戶 為 1111-1111) 會傳送物件作業要求 (例如)，針對 AWS 帳戶 3333-3333-3333 所擁有的物件，在 2222-2222 擁有的值區中傳送物件作業要求 (例如`GetObject`)。 AWS 帳戶



Jill 需要父項 AWS 帳戶、值區擁有者和物件擁有者的許可。Amazon S3 評估內容的方式如下：

1. 由於請求來自 IAM 主體，因此 Amazon S3 會評估使用者內容，以確認父項 AWS 帳戶 1111-1111 是否已授予 Jill 執行所要求作業的權限。如果她具備該許可，Amazon S3 會評估儲存貯體內容。否則，Amazon S3 會拒絕要求。
2. 在值區內容中，值區擁有者 AWS 帳戶 2222-2222-2222 是內容權限。Amazon S3 會評估儲存貯體政策，判斷儲存貯體擁有者是否已明確拒絕 Jill 存取物件。
3. 在物件內容中，內容授權單位是 AWS 帳戶 3333-3333-3333，也就是物件擁有者。Amazon S3 會評估物件 ACL，判斷 Jill 是否具備存取物件的許可。如果具備，Amazon S3 會授權要求。

AWS Amazon S3 的受管政策

受 AWS 管理的策略是由建立和管理的獨立策略 AWS。 AWS 受管理的策略旨在為許多常見使用案例提供權限，以便您可以開始將權限指派給使用者、群組和角色。

請記住， AWS 受管理的政策可能不會為您的特定使用案例授與最低權限權限，因為這些權限可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法變更受 AWS 管理策略中定義的權限。如果 AWS 更新 AWS 受管理原則中定義的權限，則此更新會影響附加原則的所有主體識別 (使用者、群組和角色)。 AWS 當新的啟動或新 AWS 服務 的 API 操作可用於現有服務時，最有可能更新 AWS 受管理策略。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

AWS 受管理策略：AmazonS3FullAccess

您可將 `AmazonS3FullAccess` 政策連接到 IAM 身分。此政策授與允許 Amazon S3 完整存取的許可。

若要檢視此政策的許可，請參閱 AWS Management Console中的 AmazonS3FullAccess。

AWS 受管理策略：AmazonS3ReadOnlyAccess

您可將 `AmazonS3ReadOnlyAccess` 政策連接到 IAM 身分。此政策授與允許 Amazon S3 唯讀存取的許可。

若要檢視此政策的許可，請參閱 AWS Management Console中的 AmazonS3ReadOnlyAccess。

AWS 受管政策：AmazonS3ObjectLambdaExecutionRolePolicy

提供 AWS Lambda 函數在向 S3 物件 Lambda 存取點發出請求時，將資料傳送至 S3 物件 Lambda 所需的權限。此外，還授予 Lambda 許可以寫入 Amazon CloudWatch 日誌。

若要檢視此政策的許可，請參閱 AWS Management Console中的 AmazonS3ObjectLambdaExecutionRolePolicy。

Amazon S3 更新受 AWS 管政策

檢視有關 Amazon S3 AWS 受管政策更新的詳細資訊，因為此服務開始追蹤這些變更。

變更	描述	日期
Amazon S3 新增了 Describe 許可至 AmazonS3ReadOnlyAccess	Amazon S3 新增了 s3:Describe* 許可至 AmazonS3ReadOnlyAccess 。	2023 年 8 月 11 日
Amazon S3 將 S3 Object Lambda 許可新增至 AmazonS3FullAccess 和 AmazonS3ReadOnlyAccess	Amazon S3 更新了 AmazonS3FullAccess 和 AmazonS3ReadOnlyAccess 政策來包含 S3 Object Lambda 的許可。	2021 年 9 月 27 日
Amazon S3 新增了 AmazonS3ObjectLambdaExecutionRolePolicy	Amazon S3 新增了名為的新 AWS受管政策AmazonS3ObjectLambdaExecutionRolePolicy ，提供 Lambda 函數許可以與 S3 物件 Lambda 互動並寫入 CloudWatch 日誌。	2021 年 8 月 18 日
Amazon S3 開始追蹤變更	Amazon S3 開始追蹤其 AWS 受管政策的變更。	2021 年 8 月 18 日

讓 Amazon S3 Storage Lens 使用服務連結角色

若要使用 Amazon S3 Storage Lens 來收集和彙總 AWS Organizations中所有帳戶的指標，您必須首先確保 S3 Storage Lens 具有由組織管理帳戶啟用的受信任存取權。S3 儲存鏡頭會建立服務連結角色 (SLR)，讓其取得 AWS 帳戶 屬於您組織的清單。S3 Storage Lens 會在建立或更新 S3 Storage Lens 儀表板或組態時，使用此帳戶清單來收集所有成員帳戶中 S3 資源的指標。

Amazon S3 儲存透鏡使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是特殊類型的 IAM 角色，此角色可直接連結到 S3 Storage Lens。S3 Storage Lens 預先定義服務連結角色，並包含服務代表您呼叫其他人所需 AWS 服務 的所有許可。

服務連結角色可讓設定 S3 Storage Lens 更為簡單，因為您不必手動新增必要的許可。S3 Storage Lens 定義其服務連結角色的許可，除非另有定義，否則僅有 S3 Storage Lens 可以擔任其角色。定義的許可包括信任政策和許可政策，並且該許可政策不能連接到任何其他 IAM 實體。

您必須先刪除相關的資源，才能刪除服務連結角色。如此可保護您 S3 Storage Lens 的資源，避免您不小心移除資源的存取許可。

如需關於支援服務連結角色的其他服務資訊，請參閱《可搭配 IAM 運作的AWS 服務》，尋找 Service-linked roles (服務連結角色) 欄中顯示為 Yes (是) 的服務。選擇具有連結的 Yes (是)，以檢視該服務的服務連結角色文件。

Amazon S3 Storage Lens 的服務連結角色許可

S3 儲存鏡頭使用名為的服務連結角色 AWSServiceRoleForS3StorageLens— 這可讓您存取 S3 儲存鏡頭所使用或管理的 AWS 服務和資源。這可讓 S3 儲存鏡頭代表您存取 AWS Organizations 資源。

S3 Storage Lens 服務連結角色信任組織儲存裝置上的下列服務：

- `storage-lens.s3.amazonaws.com`

角色許可政策允許 S3 Storage Lens 完成下列動作。

- `organizations:DescribeOrganization`

  `organizations:ListAccounts`

  `organizations:ListAWSServiceAccessForOrganization`

  `organizations:ListDelegatedAdministrators`

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

為 S3 Storage Lens 建立服務連結角色

您不需要手動建立一個服務連結角色。當您在登入 AWS Organizations 管理或委派管理員帳戶時完成下列其中一項工作時，S3 Storage Lens 會為您建立服務連結角色：

- 在 Amazon S3 主控台中為組織建立 S3 Storage Lens 儀表板組態。
- PUT您組織使用 REST API AWS CLI 和開發套件的 S3 儲存鏡頭組態。

> **Note**
>
> S3 儲存鏡頭每個組織最多可支援五個委派管理員。

如果您刪除此服務連結角色，上述動作會視需要重新建立該角色。

S3 Storage Lens 服務連結角色的範例政策

Example S3 Storage Lens 服務連結角色的許可政策

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AwsOrgsAccess",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization",
                "organizations:ListAccounts",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

編輯 Amazon S3 Storage Lens 的服務連結角色

S3 儲存鏡頭不允許您編輯AWSServiceRoleForS3StorageLens服務連結角色。因為可能有各種實體會參考服務連結角色，所以您無法在建立角色之後變更其名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱《IAM 使用者指南》中的編輯服務連結角色。

刪除 Amazon S3 Storage Lens 的服務連結角色

如果您不再需要使用服務連結角色，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。

> **Note**
>
> 如果 Amazon S3 Storage Lens 服務在您試圖刪除資源時正在使用該角色，刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

若要刪除，AWSServiceRoleForS3StorageLens您必須 AWS 區域 使用 AWS Organizations 管理或委派管理員帳戶刪除所有組織層級 S3 Storage Lens 組態。

這些資源為組織層級 S3 Storage Lens 組態。使用 S3 儲存鏡頭清理資源，然後使用 IAM 主控台、CLI、REST API 或 AWS 開發套件刪除角色。

在 REST API 和開發套件中，您可以在組織建立 S3 儲存鏡頭組態的所有區域`ListStorageLensConfigurations`中探索 S3 儲存鏡頭組態。 AWS CLI使用動作 `DeleteStorageLensConfiguration` 刪除這些組態，以便您可以刪除角色。

> **Note**
>
> 若要刪除服務連結角色，您必須刪除它們存在的所有區域中的所有組織層級 S3 Storage Lens 組態。

若要刪除AWSServiceRoleForS3StorageLens單反相機使用的 Amazon S3 儲存鏡頭資源

1. 若要取得組織層級組態的清單，您必須`ListStorageLensConfigurations`在具有 S3 儲存鏡頭組態的每個區域中使用。您也可以從 Amazon S3 主控台取得此清單。
2. 呼叫 `DeleteStorageLensConfiguration` API 呼叫或使用 Amazon S3 主控台，從適當的區域端點刪除這些組態。

使用 IAM 手動刪除服務連結角色

刪除設定之後，請從 IAM 主控台或叫用 IAM API `DeleteServiceLinkedRole` 或使用或 SDK 刪除 AWSServiceRoleForS3StorageLens單鏡反光相機。 AWS CLI AWS 如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

支援 S3 Storage Lens 服務連結角色的區域

S3 儲存鏡頭支援在所有可用服務的 AWS 區域 地方使用服務連結角色。如需詳細資訊，請參閱「Amazon S3 區域和端點」。

疑難排解 Amazon S3 身分識別和存取

使用下列資訊協助您診斷和修正使用 Amazon S3 和 IAM 時可能遇到的常見問題。

主題

- 我收到訪問被拒絕的錯誤
- 我沒有授權在 Amazon S3 中執行操作
- 我沒有授權執行 iam：PassRole
- 我想允許我以外的人訪 AWS 帳戶 問我的 Amazon S3 資源

我收到訪問被拒絕的錯誤

確認您嘗試在值區政策或以身分識別為基礎的政策中授予權限的請求者沒有明確的`Deny`陳述式。

如需疑難排解拒絕存取錯誤的詳細資訊，請參閱針對 Amazon S3 中的拒絕存取 (403 禁止) 錯誤進行疑難排解。

我沒有授權在 Amazon S3 中執行操作

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在`mateojackson` IAM 使用者嘗試使用主控台檢視一個虛構 *`my-example-widget`* 資源的詳細資訊，但卻無虛構 `s3:`*`GetWidget`* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 s3:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 `mateojackson` 使用者的政策，允許使用 `s3:`*`GetWidget`* 動作存取 *`my-example-widget`* 資源。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行`iam:PassRole`動作的錯誤訊息，則必須更新您的政策以允許您將角色傳遞給 Amazon S3。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的 IAM 使用者`marymajor`嘗試使用主控台在 Amazon S3 中執行動作時，會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 `iam:PassRole` 動作。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想允許我以外的人訪 AWS 帳戶 問我的 Amazon S3 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 Amazon S3 是否支援這些功能，請參閱Amazon S3 如何與 IAM 配合使用。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱《IAM 使用者指南》中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解跨帳戶存取使用角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。

使用 S3 Access Grants 管理存取

為了遵守最低權限原則，您可以根據應用程式、人物角色、群組或組織單位來定義對 Amazon S3 資料的精細存取權。您可以根據存取模式的規模和複雜性，使用各種方法來達到對 Amazon S3 中資料的精細存取。

透過 AWS Identity and Access Management (IAM) 主體管理 Amazon S3 中資料集 small-to-medium 數量存取的最簡單方法是定義 IAM 許可政策和 S3 儲存貯體政策。只要必要的政策符合 S3 儲存貯體政策 (20 KB) 和 IAM 政策 (5 KB) 的政策大小限制，並且在每個帳戶允許的 IAM 主體數量範圍內，此策略就有效。

隨著資料集和使用案例數量的擴展，您可能需要更多政策空間。能夠為政策陳述式提供更多空間的方法，是使用 S3 存取點作為額外的 S3 儲存貯體端點，因為每個存取點都可以有自己的政策。您可以定義相當精細的存取控制模式，因為每個帳戶可以有數千個存取點， AWS 區域 每個存取點的政策大小最多可達 20 KB。雖然 S3 存取點會增加可用的政策空間量，但它需有一套機制，讓用戶端能夠為適當的資料集找到適當的存取點。

第三種方法是實作 IAM 工作階段中介裝置模式，您可以在其中針對每個存取工作階段實作存取決策邏輯，並動態產生短期 IAM 工作階段憑證。雖然 IAM 工作階段中介裝置方法可支援任意動態許可模式並有效擴展，但您必須建置存取模式邏輯。

您可以使用 S3 Access Grants 來管理 Amazon S3 資料的存取權，而不使用上述方法。S3 Access Grants 提供簡化的模型，可透過字首、儲存貯體或物件定義 Amazon S3 中資料的存取許可。此外，您可以使用 S3 Access Grants 來授予 IAM 主體的存取權，以及直接從公司目錄授予使用者或群組的存取權。

您時常透過將使用者和群組映射至資料集來定義 Amazon S3 中資料的許可。您可以使用 S3 Access Grants 來定義 Amazon S3 儲存貯體和物件內，使用者和角色與 S3 字首的直接存取映射。透過 S3 Access Grants 中簡化的存取結構描述，您可以依每個 S3 字首為基礎將唯讀、唯寫或讀寫存取權授予 IAM 主體，以及直接從公司目錄將存取權授予使用者或群組。應用程式可使用這些 S3 Access Grants 功能，以應用程式目前已驗證的使用者身分向 Amazon S3 請求資料。

當您將 S3 Access Grants 與的受信任身分傳播功能整合時 AWS IAM Identity Center，您的應用程式可以直接代表經過驗證的公司目錄使用者向 AWS 服務 (包括 S3 Access Grants) 提出請求。您的應用程式不再需要先將使用者映射至 IAM 主體。此外，由於最終使用者身分會一直傳播到 Amazon S3，因此，哪個使用者存取了哪個 S3 物件的稽核工作就更簡單。您不再需要重建不同使用者和 IAM 工作階段之間的關係。當您將 S3 Access Grants 與 IAM Identity Center 信任的身分傳播搭配使用時，Amazon S3 的每個 AWS CloudTrail 資料事件都包含代表其存取資料的最終使用者的直接參考。

如需有關 S3 Access Grants 的詳細資訊，請參閱下列主題。

主題

S3 Access Grants 概念

S3 Access Grants 導入下列概念來簡化其存取結構描述：

S3 Access Grants 執行個體

S3 Access Grants 執行個體是個別授權的邏輯容器，負責定義誰擁有何種 Amazon S3 資料的何種層級存取權。每個 AWS 區域 的每個 AWS 帳戶可以擁有一個 S3 Access Grants 執行個體。您可以使用此 S3 Access Grants 執行個體來控制對相同帳戶和中所有儲存貯體的存取 AWS 區域。如果您想要使用 S3 存取授權授與公司目錄中使用者和群組身分的存取權，您還必須將 S3 Access Grants 執行個體與 AWS Identity and Access Management (IAM) 身分中心執行個體建立關聯。

位置

位置定義 S3 Access Grants 執行個體可以授予存取權的資料。S3 Access Grants 的運作方式是供應存取權範圍限於特定 S3 字首、儲存貯體或物件的 IAM 憑證。您可以將 S3 Access Grants 位置與 IAM 角色建立關聯，從中建立這些臨時工作階段。最常見的位置組態是整個 S3 Access Grants 執行個體使用單一位置 `s3://`，可涵蓋帳戶和 AWS 區域中所有 S3 儲存貯體的存取權。您也可以在 S3 Access Grants 執行個體中建立多個位置。例如，您可以將儲存貯體註冊為位置 `s3://example-s3-bucket1`，並將授權限於此儲存貯體，您也可以註冊預設位置 `s3://`。

授權

若要縮小某個位置內的存取範圍，您可以建立個別授權。S3 Access Grants 執行個體中的個別授權可讓特定實體 (IAM 主體或公司目錄中的使用者或群組) 存取 Amazon S3 字首、儲存貯體或物件。您可以針對每個授權定義不同的範圍 (字首、儲存貯體或物件) 和存取層級 (READ、WRITE 或 READWRITE)。例如，您可能擁有可讓特定公司目錄群組以 01234567-89ab-cdef-0123-456789abcdef READ 方式存取 s3://*example-s3-bucket1*/projects/items/* 的授權。此授權可讓該群組中的使用者以 READ 方式存取名為 *example-s3-bucket1* 的儲存貯體中索引鍵名稱字首為 projects/items/ 的每個物件。

S3 Access Grants 臨時憑證

應用程式可透過呼叫新的 S3 API 作業來請求 just-in-time 存取登入資料 GetDataAccess，以請求存取權限層級為、或的單一物件READWRITE、前置詞或儲存貯體READWRITE。S3 Access Grants 執行個體會根據自有的授權來評估 GetDataAccess 請求。如果有相符的授權，S3 Access Grants 便會擔任與相符授權的位置相關聯的 IAM 角色。然後，S3 Access Grants 會將 IAM 工作階段的許可範圍明確設定為授權範圍所指定的 S3 儲存貯體、字首或物件。臨時存取認證的到期時間預設為 1 小時，但您可以將其設定為 15 分鐘到 12 小時的任何值。

運作方式

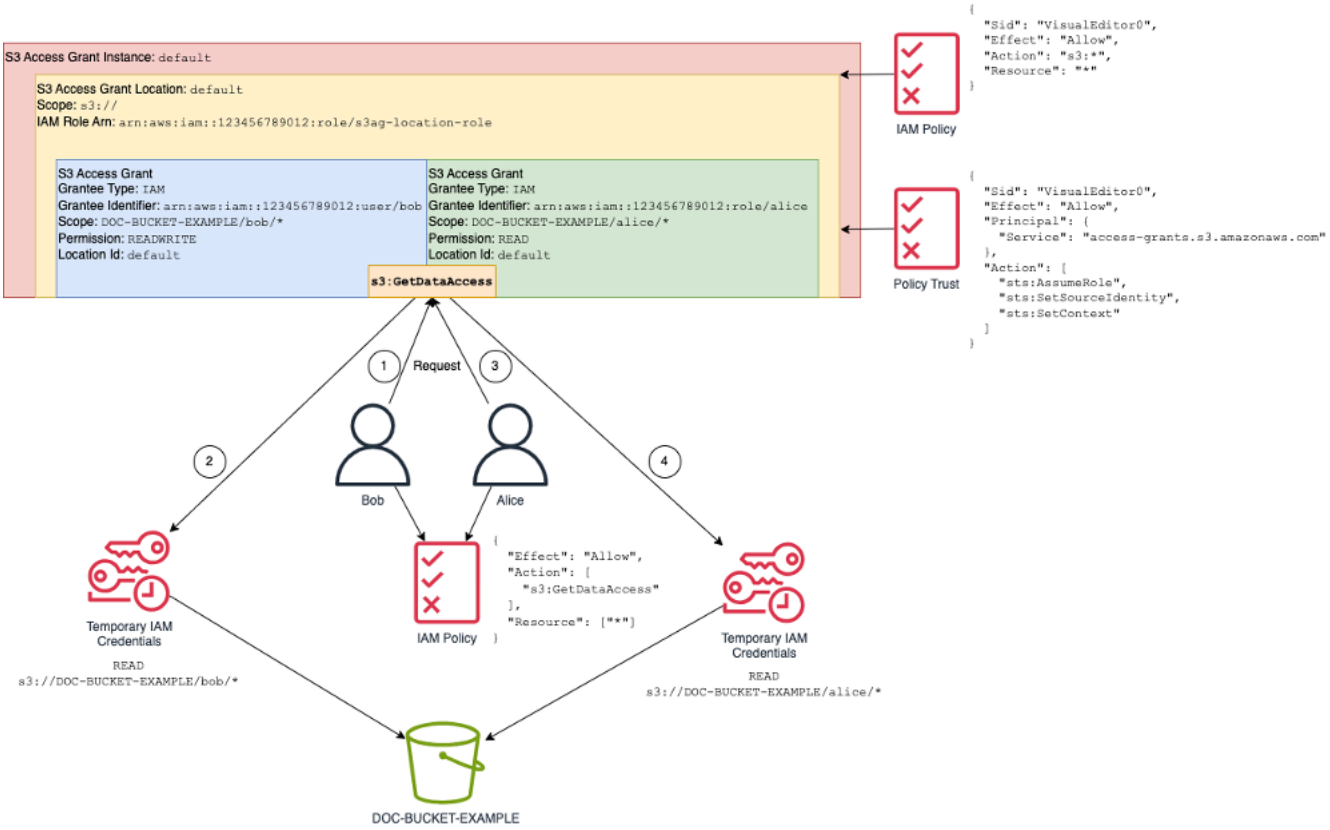
在下圖中，已向 IAM 角色 s3ag-location-role 註冊具有範圍 s3:// 的預設 Amazon S3 位置。當透過 S3 Access Grants 取得憑證時，此 IAM 角色具有在帳戶內執行 Amazon S3 動作的許可。

在此位置中，會為兩個 IAM 使用者建立兩個個別存取授權。IAM 使用者 Bob 會同時獲得 DOC-BUCKET-EXAMPLE 儲存貯體中 bob/ 字首的 READ 和 WRITE 存取權。另一個 IAM 角色 Alice 只會授與DOC-BUCKET-EXAMPLE值區中alice/前置詞的READ存取權。藍色的授權定義為讓 Bob 存取 DOC-BUCKET-EXAMPLE 儲存貯體中的 bob/ 字首。綠色的授權定義為讓 Alice 存取 DOC-BUCKET-EXAMPLE 儲存貯體中的 alice/ 字首。

Bob 需要READ資料時，與其授權位置相關聯的 IAM 角色會呼叫 S3 存取授予 GetDataAccessAPI 作業。如果 Bob 嘗試對開頭為 s3://DOC-BUCKET-EXAMPLE/bob/* 的任何 S3 字首或物件執行 READ 操作，則 GetDataAccess 請求會傳回一組具有 s3://DOC-BUCKET-EXAMPLE/bob/* 許可的臨時 IAM 工作階段憑證。同樣地，Bob 可以對開頭為 s3://DOC-BUCKET-EXAMPLE/bob/* 的任何 S3 字首或物件執行 WRITE 操作，因為授權也允許此操作。

Alice 也可以對開頭為 s3://DOC-BUCKET-EXAMPLE/alice/ 的任何項目執行 READ。然而，如果她嘗試對 s3:// 中的任何儲存貯體、字首或物件執行 WRITE 操作，則會收到拒絕存取 (403 禁止) 錯誤，因為她沒有可對任何資料進行 WRITE 存取的授權。此外，如果 Alice 針對 s3://DOC-BUCKET-

EXAMPLE/alice/ 以外的資料提出任何層級的存取權 (READ 或 WRITE) 請求，她將會再次收到「拒絕存取」錯誤。



此模式可擴展到大量的使用者和儲存貯體，並簡化這些許可的管理工作。您可以新增和移除個別的獨立授權，而不需每次想要新增或移除個別使用者字首存取關係時，都得編輯可能很龐大的 S3 儲存貯體政策。

S3 Access Grants 和公司目錄身分

您可以使用 Amazon S3 存取授權來授與 AWS Identity and Access Management (IAM) 主體 (使用者或角色) 的存取權，無論是在同 AWS 帳戶 一位置還是其他人。不過，在許多情況下，存取資料的實體是公司目錄中的最終使用者。您可以使用 S3 Access Grants 直接將存取權授予公司使用者和群組，而不是將存取權授予 IAM 主體。透過 S3 Access Grants，您不再需要將公司身分映射至中繼 IAM 主體，即可透過公司應用程式存取 S3 資料。

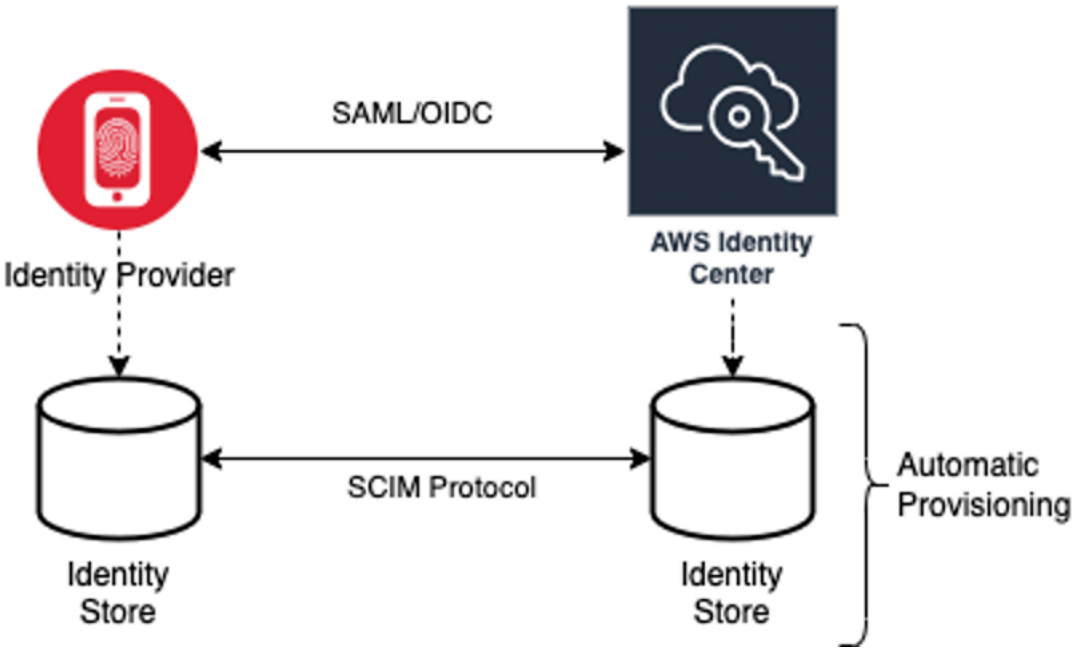
這項新功能 — 支援使用最終使用者身分存取資料 — 透過將 S3 Access Grants 執行個體與執行個體建立關聯而提供。 AWS IAM Identity Center IAM 身分中心支援標準型身分識別提供者，也是支援使用者身分的任何服務或功能 (包括 S3 存取授權) 的中樞。 AWS IAM Identity Center 透過其信任的身分傳播功能為企業身分提供身分驗證支援。如需詳細資訊，請參閱跨應用程式的信任身分傳播。

若要開始使用 S3 Access Grants 中的人力身分支援，您必須先在 IAM Identity Center 中設定企業身分提供者與 IAM Identity Center 之間的身分佈建。IAM Identity Center 支援企業身分提供者，例如 Okta、Microsoft Entra ID (舊稱為 Azure Active Directory)，或任何其他支援跨網域身分識別管理系統 (SCIM) 通訊協定的外部身分提供者 (IdP)。當您將 IAM Identity Center 連線到 IdP 並啟用自動佈建時，IdP 中的使用者和群組會同步到 IAM Identity Center 的身分存放區。完成此步驟之後，IAM Identity Center 會擁有自己的使用者和群組檢視，因此您可以使用其他 AWS 服務 和功能 (例如 S3 存取權授與) 來參考使用者和群組。如需設定 IAM Identity Center 自動佈建的詳細資訊，請參閱《AWS IAM Identity Center 使用者指南》中的自動佈建。

IAM 身分中心與整合， AWS Organizations 因此您可以集中管理多個帳戶的許可， AWS 帳戶 而無需手動設定每個帳戶。在一般組織中，您的身分管理員會為整個組織設定一個 IAM Identity Center 執行個體，作為單一身分同步點。這個 IAM 身分中心執行個體通常是在組織中的專用執行個體 AWS 帳戶 中執行。在此通用組態中，您可以參考組織中任何 AWS 帳戶 一個 S3 存取授與中的使用者和群組身分。

不過，如果您的 AWS Organizations 管理員尚未設定中央 IAM 身分中心執行個體，您可以在與 S3 存取授權執行個體相同的帳戶中建立本機執行個體。這種配置對於本地開發用例 proof-of-concept 或本地開發用例更為常見。在所有情況下，IAM 身分中心執行個體必須與要關聯 AWS 區域 的 S3 存取授權執行個體位於相同的情況下。

下圖中的 IAM Identity Center 組態具有外部 IdP，該 IdP 設定為使用 SCIM 將 IdP 的身分存放區同步到 IAM Identity Center 的身分存放區。



若要使用公司目錄身分搭配 S3 Access Grants，請執行下列操作：

- 在 IAM Identity Center 中設定自動佈建，以將使用者和群組資訊從 IdP 同步到 IAM Identity Center。
- 在 IAM Identity Center 內將您的外部身分來源設定為可信權杖發行者。如需詳細資訊，請參閱《AWS IAM Identity Center 使用者指南》中的跨應用程式的受信任身分傳播。
- 將您的 S3 Access Grants 執行個體與 IAM Identity Center 執行個體建立關聯。您可以在建立 S3 Access Grants 執行個體時執行此操作。如果您已建立 S3 Access Grants 執行個體，請參閱 關聯和取消關聯 IAM Identity Center 執行個體。

目錄身分如何存取 S3 資料

假設您的公司目錄使用者需要透過公司應用程式 (例如文件檢視器應用程式) 存取 S3 資料，且該應用程式與您的外部 IdP (例如 Okta) 整合以驗證使用者身分。這些應用程式中的使用者身分驗證通常是透過使用者 Web 瀏覽器中的重新導向來完成。由於目錄中的使用者不是 IAM 主體，因此您的應用程式需有可用來呼叫 S3 Access Grants `GetDataAccess` API 操作的 IAM 憑證，才能代表使用者取得 S3 資料的存取憑證。與自行取得憑證的 IAM 使用者和角色不同的是，您的應用程式需要一種方式來代表未映射至 IAM 角色的目錄使用者，如此使用者才能透過 S3 Access Grants 取得資料存取權。

從已驗證目錄使用者轉換成可代表目錄使用者向 S3 Access Grants 提出請求的 IAM 呼叫者，這個過程是由應用程式透過 IAM Identity Center 的可信權杖發行者功能來完成。在驗證目錄使用者之後，應用程式會擁有來自 IdP 的身分權杖 (例如 Okta)，該權杖會根據 Okta 代表目錄使用者。IAM Identity Center 中的可信權杖發行者組態可讓應用程式用此 Okta 權杖 (Okta 租用戶設定為「信任的發行者」) 交換來自 IAM Identity Center 的不同身分權杖，藉此安全地代表 AWS 服務內的目錄使用者。然後，資料應用程式將擔任 IAM 角色，從 IAM Identity Center 提供目錄使用者的權杖作為額外內容。應用程式可以使用產生的 IAM 工作階段來呼叫 S3 Access Grants。權杖代表應用程式的身分 (IAM 主體本身) 以及目錄使用者的身分。

此轉換過程的主要步驟是權杖交換。應用程式透過在 IAM Identity Center 中呼叫 `CreateTokenWithIAM` API 操作來執行此權杖交換。當然，這也是一個 AWS API 調用，需要 IAM 主體進行簽名。提出此請求的 IAM 主體通常是與應用程式相關聯的 IAM 角色。例如，如果應用程式在 Amazon EC2 上執行，則 `CreateTokenWithIAM` 請求通常是由與應用程式執行所在的 EC2 執行個體相關聯的 IAM 角色執行。成功`CreateTokenWithIAM`調用的結果是一個新的身份令牌，它將在其中識別 AWS 服務。

下一步是要在應用程式代表目錄使用者呼叫 `GetDataAccess` 之前，讓應用程式取得包含目錄使用者身分的 IAM 工作階段。應用程式會透過 AWS Security Token Service (AWS STS) `AssumeRole` 請求執行此作業，該請求也包含目錄使用者的 IAM 身分中心憑證，做為其他身分內容。這個額外內容可讓 IAM Identity Center 將目錄使用者的身分傳播至下一個步驟。應用程式擔任的 IAM 角色需有 IAM 許可才能呼叫 `GetDataAccess` 操作。

擔任具有目錄使用者的 IAM Identity Center 權杖作為額外內容的身分持有人 IAM 角色後，應用程式現在一切就緒，可代表已驗證目錄使用者向 `GetDataAccess` 提出經簽署的請求。

權杖傳播的步驟如下：

建立 IAM Identity Center 應用程式

首先，在 IAM Identity Center 建立新的應用程式。此應用程式將使用範本，以允許 IAM Identity Center 識別您可以使用的應用程式設定類型。建立應用程式的命令會要求您提供 IAM Identity Center 執行個體 Amazon Resource Name (ARN)、應用程式名稱和應用程式提供者 ARN。應用程式提供者是應用程式將用來呼叫 IAM Identity Center 的 SAML 或 OAuth 應用程式提供者。

若要使用下列範例命令，請將 *`user input placeholders`* 取代為您自己的資訊：

```
aws sso-admin create-application \
 --instance-arn "arn:aws:sso:::instance/ssoins-ssoins-1234567890abcdef" \
 --application-provider-arn "arn:aws:sso::aws:applicationProvider/custom" \
 --name MyDataApplication
```

回應：

```
{
   "ApplicationArn": "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d"
}
```

建立可信權杖發行者

現在您已有 IAM Identity Center 應用程式，下一步就是設定可信權杖發行者，它將用來交換 IdP 的 `IdToken` 值與 IAM Identity Center 權杖。您需要在此步驟中提供下列項目：

- 身分提供者發行者 URL
- 可信權杖發行者名稱
- 宣告屬性路徑
- 身分存放區屬性路徑
- JSON Web 金鑰組 (JWKS) 擷取選項

宣告屬性路徑是身分提供者屬性，將用來映射身分存放區屬性。通常，宣告屬性路徑是使用者的電子郵件地址，但您可以使用其他屬性來執行映射。

建立稱為 oidc-configuration.json 的檔案並包含下列資訊。若要使用此檔案，請將 *user input placeholders* 取代為您自己的資訊。

```
{
  "OidcJwtConfiguration":
     {
      "IssuerUrl": "https://login.microsoftonline.com/a1b2c3d4-abcd-1234-b7d5-
b154440ac123/v2.0",
      "ClaimAttributePath": "preferred_username",
      "IdentityStoreAttributePath": "userName",
      "JwksRetrievalOption": "OPEN_ID_DISCOVERY"
     }
}
```

若要建立可信權杖發行者，請執行下列命令。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws sso-admin create-trusted-token-issuer \
  --instance-arn "arn:aws:sso:::instance/ssoins-1234567890abcdef" \
  --name MyEntraIDTrustedIssuer \
  --trusted-token-issuer-type OIDC_JWT \
  --trusted-token-issuer-configuration file://./oidc-configuration.json
```

回應

```
{
  "TrustedTokenIssuerArn": "arn:aws:sso::123456789012:trustedTokenIssuer/
ssoins-1234567890abcdef/tti-43b4a822-1234-1234-1234-a1b2c3d41234"
}
```

將 IAM Identity Center 應用程式與可信權杖發行者連線

可信權杖發行者會需要進行幾項額外的組態設定，以便正常運作。設定可信權杖發行者將信任的對象。對象是 IdToken 內金鑰所識別的值，可在身分提供者設定中找到。例如：

```
1234973b-abcd-1234-abcd-345c5a9c1234
```

建立名為 grant.json 且包含下列內容的檔案。若要使用此檔案，請變更對象以符合您的身分提供者設定，並提供上一個命令傳回的可信權杖發行者 ARN。

```
{
```

```
   "JwtBearer":
     {
       "AuthorizedTokenIssuers":
         [
           {
             "TrustedTokenIssuerArn": "arn:aws:sso::123456789012:trustedTokenIssuer/
ssoins-1234567890abcdef/tti-43b4a822-1234-1234-1234-a1b2c3d41234",
               "AuthorizedAudiences":
                 [
                   "1234973b-abcd-1234-abcd-345c5a9c1234"
                 ]
            }
         ]
     }
 }
```

執行下列範例命令。若要使用此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws sso-admin put-application-grant \
  --application-arn "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d" \
  --grant-type "urn:ietf:params:oauth:grant-type:jwt-bearer" \
  --grant file://./grant.json \
```

此命令會使用組態設定來設定可信權杖發行者，以信任 `grant.json` 檔案中的對象，並將此對象與第一個步驟中建立的應用程式連結，以交換 `jwt-bearer` 類型的權杖。字串 `urn:ietf:params:oauth:grant-type:jwt-bearer` 不是任意字串。它是 OAuth JSON Web 權杖 (JWT) 聲明設定檔中註冊的命名空間。您可以在 RFC 7523 中找到有關此命名空間的詳細資訊。

接下來，使用下列命令設定可信權杖發行者從身分提供者交換 `IdToken` 值時要包括哪些範圍。對於 S3 Access Grants，`--scope` 參數的值為 `s3:access_grants:read_write`。

```
aws sso-admin put-application-access-scope \
  --application-arn "arn:aws:sso::111122223333:application/ssoins-
ssoins-111122223333abcdef/apl-abcd1234a1b2c3d" \
  --scope "s3:access_grants:read_write"
```

最後一個步驟是將資源政策連接到 IAM Identity Center 應用程式。此政策可讓應用程式 IAM 角色向 API 操作 `sso-oauth:CreateTokenWithIAM` 發出請求，並從 IAM Identity Center 接收 `IdToken` 值。

建立名為 authentication-method.json 且包含下列內容的檔案。使用您的帳戶 ID 取代 *123456789012*。

```
{
   "Iam":
       {
         "ActorPolicy":
             {
                "Version": "2012-10-17",
                    "Statement":
                    [
                        {
                           "Effect": "Allow",
                            "Principal":
                            {
                              "AWS": "arn:aws:iam::123456789012:role/webapp"
                            },
                           "Action": "sso-oauth:CreateTokenWithIAM",
                            "Resource": "*"
                        }
                    ]
                }
            }
        }
```

若要將政策連接至 IAM Identity Center 應用程式，請執行下列命令：

```
aws sso-admin put-application-authentication-method \
   --application-arn "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d" \
   --authentication-method-type IAM \
   --authentication-method file://./authentication-method.json
```

這樣就完成了透過 Web 應用程式搭配目錄使用者使用 S3 Access Grants 的組態設定。您可以直接在應用程式中測試此設定，也可以從 IAM Identity Center 應用程式政策中允許的 IAM 角色使用下列命令來呼叫 CreateTokenWithIAM API 操作：

```
aws sso-oidc create-token-with-iam \
   --client-id "arn:aws:sso::123456789012:application/ssoins-ssoins-1234567890abcdef/
apl-abcd1234a1b2c3d"  \
   --grant-type urn:ietf:params:oauth:grant-type:jwt-bearer \
```

```
    --assertion IdToken
```

回應類似以下內容：

```
{
    "accessToken": "<suppressed long string to reduce space>",
    "tokenType": "Bearer",
    "expiresIn": 3600,
    "refreshToken": "<suppressed long string to reduce space>",
    "idToken": "<suppressed long string to reduce space>",
    "issuedTokenType": "urn:ietf:params:oauth:token-type:refresh_token",
    "scope": [
      "sts:identity_context",
      "s3:access_grants:read_write",
      "openid",
      "aws"
    ]
}
```

如果您將使用 base64 編碼的 IdToken 值解碼，則會看到 JSON 格式的金鑰值對。金鑰 sts:identity_context 包含應用程式需要在 sts:AssumeRole 請求中傳送的值，以包含目錄使用者的身分資訊。以下是解碼的 IdToken 範例：

```
{
    "aws:identity_store_id": "d-996773e796",
    "sts:identity_context": "AQoJb3JpZ2luX2VjEOTtl;<SUPRESSED>",
    "sub": "83d43802-00b1-7054-db02-f1d683aacba5",
    "aws:instance_account": "123456789012",
    "iss": "https://identitycenter.amazonaws.com/ssoins-1234567890abcdef",
    "sts:audit_context": "AQoJb3JpZ2luX2VjEOT<SUPRESSED>==",
    "aws:identity_store_arn": "arn:aws:identitystore::232642235904:identitystore/
d-996773e796",
    "aud": "abcd12344U0gi7n4Yyp0-WV1LWNlbnRyYWwtMQ",
    "aws:instance_arn": "arn:aws:sso:::instance/ssoins-6987d7fb04cf7a51",
    "aws:credential_id": "EXAMPLEHI5glPh40y9TpApJn8...",
    "act": {
       "sub": "arn:aws:sso::232642235904:trustedTokenIssuer/
ssoins-6987d7fb04cf7a51/43b4a822-1020-7053-3631-cb2d3e28d10e"
    },
    "auth_time": "2023-11-01T20:24:28Z",
    "exp": 1698873868,
    "iat": 1698870268
```

```
}
```

您可以從 sts:identity_context 取得值，並在 sts:AssumeRole 呼叫中傳遞此資訊。以下是語法的 CLI 範例。要擔任的角色是具有調用 s3:GetDataAccess 許可的臨時角色。

```
aws sts assume-role \
   --role-arn "arn:aws:iam::123456789012:role/temp-role" \
   --role-session-name "TempDirectoryUserRole" \
   --provided-contexts ProviderArn="arn:aws:iam::aws:contextProvider/
IdentityCenter",ContextAssertion="value from sts:identity_context"
```

您現在可以使用隨此呼叫收到的憑證調用 s3:GetDataAccess API 操作，並接收具有 S3 資源存取權的最終憑證。

開始使用 S3 Access Grants

Amazon S3 Access Grants 是 Amazon S3 的一項功能，可為您的 S3 資料提供可擴展的存取控制解決方案。S3 Access Grants 是 S3 憑證供應方，這表示您會在 S3 Access Grants 註冊您的授權清單和層級。之後，當使用者或用戶端需要存取您的 S3 資料時，必須先向 S3 Access Grants 取得憑證。如果有授權存取的對應授權，S3 Access Grants 會供應臨時、最低權限的存取憑證。然後使用者或用戶端就可以使用 S3 Access Grants 供應的憑證來存取您的 S3 資料。務必記住，如果您的 S3 資料需求強制要求複雜或大型許可組態，您可以使用 S3 Access Grants 來擴展使用者、群組、角色和應用程式的 S3 資料許可。

對於大多數使用案例，您可以使用 AWS Identity and Access Management (IAM) 搭配儲存貯體政策或 IAM 身分型政策來管理 S3 資料的存取控制。

但是，如果您有複雜的 S3 存取控制需求 (如下所述)，那麼使用 S3 Access Grants 將十分有益：

- 您的儲存貯體政策大小限制為 20 KB。
- 您可以授予人員身分 (例如 Microsoft Entra ID，舊稱為 Azure Active Directory)、Okta 或 Ping 使用者和群組 S3 資料的存取權，以進行分析和大數據。
- 您必須提供跨帳戶存取權，而不需頻繁更新 IAM 政策。
- 您的資料是非結構化的物件層級資料，而非採用結構化的列和欄格式。

S3 Access Grants 工作流程如下：

步驟	描述
1	建立 S3 Access Grants 執行個體 若要開始進行，請啟動將包含個別存取授權的 S3 Access Grants 執行個體。
2	註冊位置 其次，註冊 S3 資料位置 (例如，預設 `s3://`)，然後指定 S3 Access Grants 提供 S3 資料位置存取權時所擔任的預設 IAM 角色。您也可以將自訂位置新增至特定儲存貯體或字首，並將這些位置映射至自訂 IAM 角色。
3	建立授權 建立個別許可授權。在這些許可授權中指定已註冊的 S3 位置、位置內的資料存取範圍、承授者的身分及其存取層級 (`READ`、`WRITE` 或 `READWRITE`)。
4	請求存取 S3 資料 當使用者、應用程式和 AWS 服務 想要存取 S3 資料時，他們會先提出存取要求。S3 Access Grants 會決定是否應授權請求。如果有授權存取的對應授權，S3 Access Grants 會使用與該授權相關聯的已註冊位置的 IAM 角色，將臨時憑證供應給請求者。
5	存取 S3 資料 應用程式使用 S3 Access Grants 供應的臨時憑證來存取 S3 資料。

建立 S3 Access Grants 執行個體

若要開始使用 Amazon S3 Access Grants，您必須先建立 S3 Access Grants 執行個體。每個帳戶只能建立一個 S3 存取授與執行 AWS 區域 個體。S3 Access Grants 執行個體可作為 S3 Access Grants 資源的容器，其中包括註冊的位置和授權。

使用 S3 存取授權，您可以為 AWS Identity and Access Management (IAM) 使用者和角色建立 S3 資料的許可授與。如果您已將公司身分目錄新增至 AWS IAM Identity Center，您可以將公司目錄的這個 IAM 身分中心執行個體與 S3 Access Grants 執行個體建立關聯。完成此操作之後，就可以為公司使用者和群組建立存取授權。如果您尚未將公司目錄新增至 IAM Identity Center，您可以稍後再將 S3 Access Grants 執行個體與 IAM Identity Center 執行個體建立關聯。

您可以使用 Amazon S3 主控台 AWS Command Line Interface (AWS CLI)、Amazon S3 REST API 和 AWS 開發套件來建立 S3 存取授與執行個體。

使用 S3 主控台

您必須先建立與 S3 資料相同的 S3 存取授與執行個體，才能使用 S3 存取授權授予對 S3 資料 AWS 區域 的存取權限。

必要條件

如果您想要使用公司目錄中的身分來授予 S3 資料的存取權，請新增公司身分目錄至 AWS IAM Identity Center。如果您尚未準備好這樣做，可以稍後再將 S3 Access Grants 執行個體與 IAM Identity Center 執行個體建立關聯。

建立 S3 Access Grants 執行個體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在導覽列中，選擇目前顯示的名稱 AWS 區域。接下來，選擇要切換到的區域。
3. 在左側導覽窗格中，選擇 Access Grants。
4. 在 S3 Access Grants 頁面上，選擇建立 S3 Access Grants 執行個體。

    a. 在設定 Access Grants 執行個體精靈的步驟 1 中，確認您要在目前的 AWS 區域中建立執行個體。請確定這與 S3 資料所 AWS 區域 在的位置相同。您可以為每個帳戶建立一個 S3 存取授與執行 AWS 區域 個體。

    b. (選擇性) 如果您已將公司身分目錄新增至 AWS IAM Identity Center，您可以將公司目錄的這個 IAM 身分中心執行個體與 S3 Access Grants 執行個體建立關聯。

    若要這樣做，請選取在**##**中新增 IAM Identity Center 執行個體。然後輸入 IAM Identity Center 執行個體 Amazon Resource Name (ARN)。

    如果您尚未將公司目錄新增至 IAM Identity Center，您可以稍後再將 S3 Access Grants 執行個體與 IAM Identity Center 執行個體建立關聯。

c. 若要建立 S3 Access Grants 執行個體，請選擇下一步。若要註冊位置，請參閱步驟 2 - 註冊位置。

5. 如果下一步或建立 S3 Access Grants 執行個體為停用狀態：

   無法建立執行個體

   - 您可能在相同 AWS 區域中已有 S3 Access Grants 執行個體。在左側導覽窗格中，選擇 Access Grants。在 S3 Access Grants 頁面上，向下捲動至您帳戶中的 S3 Access Grants 執行個體區段，以判斷執行個體是否已存在。
   - 您可能沒有建立 S3 Access Grants 執行個體所需的 `s3:CreateAccessGrantsInstance` 許可。請聯絡您的帳戶管理員。如需將 IAM Identity Center 執行個體與 S3 Access Grants 執行個體建立關聯所需的其他許可，請參閱 CreateAccessGrantsInstance。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

若要使用下列範例命令，請以您自己的資訊取代 *user input placeholders*。

Example 建立 S3 Access Grants 執行個體

```
aws s3control create-access-grants-instance \
--account-id 111122223333 \
--region us-east-2
```

回應：

```
{
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00",
    "AccessGrantsInstanceId": "default",
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default"
}
```

使用 REST API

您可以使用 Amazon S3 REST API 建立 S3 Access Grants 執行個體。如需有關管理 S3 Access Grants 執行個體的 REST API 支援資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列各節：

- AssociateAccessGrantsIdentityCenter
- CreateAccessGrantsInstance
- DeleteAccessGrantsInstance
- DissociateAccessGrantsIdentityCenter
- GetAccessGrantsInstance
- GetAccessGrantsInstanceForPrefix
- GetAccessGrantsInstanceResourcePolicy
- ListAccessGrantsInstances
- PutAccessGrantsInstanceResourcePolicy

使用 AWS 軟體開發套件

本節提供如何使用 AWS SDK 建立 S3 Access Grants 執行個體的範例。

Java

此範例會建立 S3 Access Grants 執行個體，作為個別存取授權的容器。您的帳戶 AWS 區域 中每個可以有一個 S3 存取授與執行個體。回應包括執行個體 ID `default`，以及針對 S3 Access Grants 執行個體產生的 Amazon Resource Name (ARN)。

Example 建立 S3 Access Grants 執行個體請求

```
public void createAccessGrantsInstance() {
CreateAccessGrantsInstanceRequest createRequest =
 CreateAccessGrantsInstanceRequest.builder().accountId("111122223333").build();
CreateAccessGrantsInstanceResponse createResponse =
 s3Control.createAccessGrantsInstance(createRequest);LOGGER.info("CreateAccessGrantsInstance
 " + createResponse);
}
```

回應：

```
CreateAccessGrantsInstanceResponse(
CreatedAt=2023-06-07T01:46:20.507Z,
AccessGrantsInstanceId=default,
AccessGrantsInstanceArn=arn:aws:s3:us-east-2:111122223333:access-grants/default)
```

主題

檢視 S3 Access Grants 執行個體的詳細資訊

您可以檢視定 AWS 區域中 Amazon S3 Access Grants 執行個體的詳細資訊。您也可以列出 S3 存取授與執行個體，包括透過 AWS Resource Access Manager (AWS RAM) 與您共用的執行個體。

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、Amazon S3 REST API 和 AWS 開發套件，檢視 S3 存取授予執行個體的詳細資訊，或列出 S3 存取授與執行個體。

使用 S3 主控台

檢視 S3 Access Grants 執行個體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Grants。
3. 在 S3 Access Grants 頁面上，選擇包含您要使用之 S3 Access Grants 執行個體的區域。
4. S3 Access Grants 頁面會列出您的 S3 Access Grants 執行個體，以及與您的帳戶共用的任何跨帳戶執行個體。若要檢視執行個體的詳細資訊，請選擇檢視詳細資訊。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

若要使用下列範例命令，請以您自己的資訊取代 *user input placeholders*。

Example – 取得 S3 Access Grants 執行個體的詳細資訊

```
aws s3control get-access-grants-instance \
 --account-id 111122223333 \
 --region us-east-2
```

回應：

```
{
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default",
    "AccessGrantsInstanceId": "default",
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00"
}
```

Example – 列出帳戶的所有 S3 Access Grants 執行個體

此動作會列出帳戶的 S3 Access Grants 執行個體。每個執行個體只能有一個 S3 存取授與執行個體 AWS 區域。此動作也會列出您的帳戶可存取的其他跨帳戶 S3 Access Grants 執行個體。

```
aws s3control list-access-grants-instances \
 --account-id 111122223333 \
 --region us-east-2
```

回應：

```
{
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default",
    "AccessGrantsInstanceId": "default",
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00"
}
```

使用 REST API

如需有關管理 S3 Access Grants 執行個體的 Amazon S3 REST API 支援資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列各節：

- GetAccessGrantsInstance
- GetAccessGrantsInstanceForPrefix
- ListAccessGrantsInstances

使用 AWS 軟體開發套件

本節提供如何使用 AWS SDK 取得 S3 存取授與執行個體詳細資訊的範例。

若要使用下列範例，請以您自己的資訊取代 *user input placeholders*。

Java

Example – 取得 S3 Access Grants 執行個體

```
public void getAccessGrantsInstance() {
GetAccessGrantsInstanceRequest getRequest = GetAccessGrantsInstanceRequest.builder()
.accountId("111122223333")
.build();
GetAccessGrantsInstanceResponse getResponse =
 s3Control.getAccessGrantsInstance(getRequest);
LOGGER.info("GetAccessGrantsInstanceResponse: " + getResponse);
}
```

回應：

```
GetAccessGrantsInstanceResponse(
AccessGrantsInstanceArn=arn:aws:s3:us-east-2: 111122223333:access-grants/default,
CreatedAt=2023-06-07T01:46:20.507Z)
```

Example – 列出帳戶的所有 S3 Access Grants 執行個體

此動作會列出帳戶的 S3 Access Grants 執行個體。每個區域只能擁有一個 S3 Access Grants 執行個體。此動作也可列出您的帳戶可存取的其他跨帳戶 S3 Access Grants 執行個體。

```
public void listAccessGrantsInstances() {
ListAccessGrantsInstancesRequest listRequest =
 ListAccessGrantsInstancesRequest.builder()
.accountId("111122223333")
.build();
ListAccessGrantsInstancesResponse listResponse =
 s3Control.listAccessGrantsInstances(listRequest);
LOGGER.info("ListAccessGrantsInstancesResponse: " + listResponse);
}
```

回應：

```
ListAccessGrantsInstancesResponse(
AccessGrantsInstancesList=[
ListAccessGrantsInstanceEntry(
AccessGrantsInstanceId=default,
AccessGrantsInstanceArn=arn:aws:s3:us-east-2:111122223333:access-grants/default,
```

```
CreatedAt=2023-06-07T04:28:11.728Z
)
]
)
```

關聯和取消關聯 IAM Identity Center 執行個體

在 Amazon S3 存取授權中，您可以將公司身分目錄的 AWS IAM Identity Center 執行個體與 S3 存取授權執行個體建立關聯。完成此操作之後，除了 AWS Identity and Access Management (IAM) 使用者和角色之外，您還可以為公司目錄使用者和群組建立存取權授與。

如果您不想再為公司目錄使用者和群組建立存取授權，可以將 IAM Identity Center 執行個體與 S3 Access Grants 執行個體取消關聯。

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、Amazon S3 REST API 和 AWS SDK，建立或取消 IAM Identity Center 執行個體的關聯。

使用 S3 主控台

如果您將 IAM Identity Center 執行個體與 S3 Access Grants 執行個體建立關聯，則必須新增公司身分目錄至 IAM Identity Center。如需詳細資訊，請參閱 the section called “S3 Access Grants 和公司目錄身分”。

將 IAM Identity Center 執行個體與 S3 Access Grants 執行個體建立關聯

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Grants。
3. 在 S3 Access Grants 頁面上，選擇包含您要使用之 S3 Access Grants 執行個體的區域。
4. 選擇執行個體的檢視詳細資訊。
5. 在詳細資訊頁面的 IAM Identity Center 區段中，選擇新增 IAM Identity Center 執行個體或取消註冊已關聯的 IAM Identity Center 執行個體。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

若要使用下列範例命令，請以您自己的資訊取代 *user input placeholders*。

Example – 將 IAM Identity Center 執行個體與 S3 Access Grants 執行個體建立關聯

```
aws s3control associate-access-grants-identity-center \
 --account-id 111122223333 \
 --identity-center-arn arn:aws:sso:::instance/ssoins-1234a567bb89012c \
 --profile access-grants-profile \
 --region eu-central-1

// No response body
```

Example – 將 IAM Identity Center 執行個體與 S3 Access Grants 執行個體取消關聯

```
aws s3control dissociate-access-grants-identity-center \
 --account-id 111122223333 \
 --profile access-grants-profile \
 --region eu-central-1

// No response body
```

使用 REST API

如需有關管理 IAM Identity Center 執行個體與 S3 Access Grants 執行個體之間關聯的 Amazon S3 REST API 支援資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列各節：

- AssociateAccessGrantsIdentityCenter
- DissociateAccessGrantsIdentityCenter

刪除 S3 Access Grants 執行個體

您可以從帳戶中刪除 Amazon S3 存取授與執 AWS 區域 行個體。不過，您必須先執行下列操作，才能刪除 S3 Access Grants 執行個體：

- 刪除 S3 Access Grants 執行個體內的所有資源，包括所有授權和位置。如需詳細資訊，請參閱刪除授權和刪除位置。
- 如果您已將 AWS IAM Identity Center 執行個體與 S3 存取授權執行個體建立關聯，則必須取消 IAM 身分中心執行個體的關聯。如需詳細資訊，請參閱關聯和取消關聯 IAM Identity Center 執行個體。

> **Important**
>
> 若您刪除 S3 Access Grants 執行個體，則會將其永久刪除且無法復原。透過此 S3 Access Grants 執行個體中的授權獲得存取權的所有承授者都將失去 S3 資料的存取權。

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、Amazon S3 REST API 和 AWS 開發套件刪除 S3 存取授與執行個體。

使用 S3 主控台

刪除 S3 Access Grants 執行個體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Grants。
3. 在 S3 Access Grants 頁面上，選擇包含您要使用之 S3 Access Grants 執行個體的區域。
4. 選擇執行個體的檢視詳細資訊。
5. 在執行個體詳細資訊頁面上，選擇右上角的刪除執行個體。
6. 在出現的對話方塊中，選擇刪除。這個動作無法復原。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

若要使用下列範例命令，請以您自己的資訊取代 *user input placeholders*。

> **Note**
>
> 您必須先刪除 S3 Access Grants 執行個體內建立的所有授權和位置，才能刪除 S3 Access Grants 執行個體。如果您已將 IAM Identity Center 執行個體與 S3 Access Grants 執行個體建立關聯，則必須先取消其關聯。

Example – 刪除 S3 Access Grants 執行個體

```
aws s3control delete-access-grants-instance \
--account-id 111122223333 \
--profile access-grants-profile \
```

```
--region us-east-2 \
--endpoint-url https://s3-control.us-east-2.amazonaws.com \

 // No response body
```

使用 REST API

如需有關刪除 S3 Access Grants 執行個體的 Amazon S3 REST API 支援資訊，請參閱《Amazon Simple Storage Service API 參考》中的 DeleteAccessGrantsInstance。

使用 AWS 軟體開發套件

本節提供如何使用 AWS SDK 刪除 S3 Access Grants 執行個體的範例。

若要使用下列範例，請以您自己的資訊取代 *user input placeholders*。

Java

> **Note**
>
> 您必須先刪除 S3 Access Grants 執行個體內建立的所有授權和位置，才能刪除 S3 Access Grants 執行個體。如果您已將 IAM Identity Center 執行個體與 S3 Access Grants 執行個體建立關聯，則必須先取消其關聯。

Example – 刪除 S3 Access Grants 執行個體

```
public void deleteAccessGrantsInstance() {
DeleteAccessGrantsInstanceRequest deleteRequest =
 DeleteAccessGrantsInstanceRequest.builder()
.accountId("111122223333")
.build();
DeleteAccessGrantsInstanceResponse deleteResponse =
 s3Control.deleteAccessGrantsInstance(deleteRequest);
LOGGER.info("DeleteAccessGrantsInstanceResponse: " + deleteResponse);
}
```

註冊位置

在您的帳戶中建立 Amazon S3 存取授與執行個體後，您可以 AWS 區域 在該執行個體中註冊 S3 位置。位置是 S3 資源，當中包含您要授予存取權的資料。您可以註冊預設位置 (即中的所有值區) AWS

區域，然後在稍後建立個別存取權授與時縮小存取範圍。s3://您也可以註冊特定儲存貯體或儲存貯體和字首作為位置。

您必須先在 S3 Access Grants 執行個體至少註冊一個位置，然後才能建立存取授權。當您註冊位置時，也必須指定 S3 Access Grants 擔任的 AWS Identity and Access Management (IAM) 角色，以便履行該位置的執行期請求，並將許可範圍縮小到執行期的特定授權。

S3 URI	IAM 角色	描述
s3://	*Default-IAM-role*	預設位置 s3:// 包括 AWS 區域中的所有儲存貯體。
s3://*example-s3-bucket1* /	*IAM-role-For-bucket*	此位置包括所指定儲存貯體中的所有物件。

請先確認執行下列作業，才能註冊位置：

- 建立一或多個儲存貯體，其中包含您要授予存取權的資料。這些儲存貯體必須與 S3 存取授權執行個體位於相同 AWS 區域 的位置。如需詳細資訊，請參閱建立儲存貯體。

  若要新增字首至儲存貯體，請參閱建立物件索引鍵名稱。
- 建立 IAM 角色，並在資源政策檔案中將 S3 Access Grants 服務主體存取權授予此角色。若要完成此操作，您可以建立包含下列陳述式的 JSON 檔案。若要將資源政策新增至您的帳戶，請參閱建立並連接您的第一個客戶管理政策。

  TestRolePolicy.json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1234567891011",
      "Action": ["sts:AssumeRole", "sts:SetSourceIdentity", "sts:SetContext"],
      "Effect": "Allow",
      "Principal": {"Service":"access-grants.s3.amazonaws.com"}
    }
  ]
}
```

- 建立 IAM 政策以將 Amazon S3 許可連接至 IAM 角色。請參閱下列範例 `iam-policy.json` 檔案，並將 *`user input placeholders`* 取代為您自己的資訊。

> **Note**
>
> - 如果您使用伺服器端加密搭配 AWS Key Management Service (AWS KMS) 金鑰來加密資料，下列範例會包含政策中 IAM 角色的必要 AWS KMS 許可。如果不使用此功能，可以從 IAM 政策中移除這些許可。
> - 只有在登入資料由 S3 存取授權提供資源時，您才可以限制 IAM 角色存取 S3 資料。此範例說明如何為特定 S3 存取授與執行個體新增`Condition`陳述式。若要這麼做，請將條件陳述式中的 S3 存取授與執行個體 ARN 取代為 S3 存取授與執行個體 ARN，格式如下：`arn:aws:s3:`*`region`*`:`*`accountId`*`:access-grants/default`

家居政策

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
         "Sid": "ObjectLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:GetObjectAcl",
            "s3:GetObjectVersionAcl",
            "s3:ListMultipartUploadParts"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:region:accountId:access-
grants/default"]
            }
        }
      },
```

```
      {
         "Sid": "ObjectLevelWritePermissions",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:PutObjectVersionAcl",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:AbortMultipartUpload"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ##:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "BucketLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ##:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "KMSPermissions",
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt",
```

```
            "kms:GenerateDataKey"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、Amazon S3 REST API 或 AWS 開發套件，在 S3 存取授予執行個體中註冊位置。

使用 S3 主控台

您必須至少先註冊一個位置，才能透過 S3 Access Grants 授予 S3 資料的存取權。

在 S3 Access Grants 執行個體中註冊位置

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Grants。
3. 在 S3 Access Grants 頁面上，選擇包含您要使用之 S3 Access Grants 執行個體的區域。

   如果您是第一次使用 S3 Access Grants 執行個體，請確定您已完成步驟 1 - 建立 S3 Access Grants 執行個體，並瀏覽至設定 Access Grants 執行個體精靈的步驟 2。如果您已有 S3 Access Grants 執行個體，請選擇檢視詳細資訊，然後從位置索引標籤選擇註冊位置。

   a. 針對位置範圍，選擇瀏覽 S3，或輸入要註冊之位置的 S3 URI 路徑。有關 S3 URI 格式，請參閱位置格式表。輸入 URI 之後，您可以選擇檢視以瀏覽位置。

   b. 針對 IAM 角色，選擇以下其中一個選項：

      - 從現有的 IAM 角色中選擇

        從下拉式清單中選擇 IAM 角色。選擇角色後，請選擇檢視以確定此角色具有管理您要註冊之位置的必要許可。具體而言，請確認此角色將 `sts:AssumeRole` 和 `sts:SetSourceIdentity` 許可授予 S3 Access Grants。

      - 輸入 IAM 角色 ARN

        導覽至 IAM 主控台。複製 IAM 角色的 Amazon 資源名稱 (ARN)，並將其粘貼到此框中。

c. 選擇下一步或註冊位置完成此操作。

4. 故障診斷：

無法註冊位置

- 位置可能已註冊。

  您可能沒有註冊位置的 `s3:CreateAccessGrantsLocation` 許可。請聯絡您的帳戶管理員。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

您可以在 S3 Access Grants 執行個體中註冊預設位置 `s3://` 或自訂位置。務必先建立具有該位置的主要存取權的 IAM 角色，然後確實授予 S3 Access Grants 許可來擔任此角色。

若要使用下列範例命令，請將 *`user input placeholders`* 取代為您自己的資訊。

Example 建立資源政策

建立允許 S3 Access Grants 擔任 IAM 角色的政策。若要完成此操作，您可以建立包含下列陳述式的 JSON 檔案。若要將資源政策新增至您的帳戶，請參閱建立並連接您的第一個客戶管理政策。

TestRolePolicy.json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1234567891011",
      "Action": ["sts:AssumeRole", "sts:SetSourceIdentity"],
      "Effect": "Allow",
      "Principal": {"Service":"access-grants.s3.amazonaws.com"}
    }
  ]
}
```

Example 建立角色

執行下列 IAM 命令以建立角色。

```
aws iam create-role --role-name accessGrantsTestRole \
 --region us-east-2 \
 --assume-role-policy-document file://TestRolePolicy.json
```

執行 `create-role` 命令會傳回政策：

```
{
    "Role": {
        "Path": "/",
        "RoleName": "accessGrantsTestRole",
        "RoleId": "AROASRDGX4WM4GH55GIDA",
        "Arn": "arn:aws:iam::111122223333:role/accessGrantsTestRole",
        "CreateDate": "2023-05-31T18:11:06+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "Stmt1685556427189",
                    "Action": [
                        "sts:AssumeRole",
                        "sts:SetSourceIdentity"
                    ],
                    "Effect": "Allow",
                    "Principal": {
                        "Service":"access-grants.s3.amazonaws.com"
                    }
                }
            ]
        }
    }
}
```

Example

建立 IAM 政策以將 Amazon S3 許可連接至 IAM 角色。請參閱下列範例 `iam-policy.json` 檔案，並將 *`user input placeholders`* 取代為您自己的資訊。

> **Note**
>
> 如果您使用伺服器端加密搭配 AWS Key Management Service (AWS KMS) 金鑰來加密資料，下列範例會在政策中新增 IAM 角色的必要 AWS KMS 許可。如果不使用此功能，可以從 IAM 政策中移除這些許可。

為了確保 IAM 角色只能在憑證由 S3 Access Grants 供應時用來存取 S3 中的資料，此範例將示範如何在 IAM 政策中新增 Condition 陳述式來指定 S3 Access Grants 執行個體 (s3:AccessGrantsInstance: *InstanceArn*)。使用下列範例政策時，請將 *user input placeholders* 取代為您自己的資訊。

家居政策

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
         "Sid": "ObjectLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:GetObjectAcl",
            "s3:GetObjectVersionAcl",
            "s3:ListMultipartUploadParts"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:region:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "ObjectLevelWritePermissions",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:PutObjectVersionAcl",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
```

```
            "s3:AbortMultipartUpload"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ##:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "BucketLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ##:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "KMSPermissions",
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt",
            "kms:GenerateDataKey"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Example

執行以下命令：

```
aws iam put-role-policy \
--role-name accessGrantsTestRole \
--policy-name accessGrantsTestRole \
--policy-document file://iam-policy.json
```

Example 註冊預設位置

```
aws s3control create-access-grants-location \
 --account-id 111122223333 \
 --location-scope s3:// \
 --iam-role-arn arn:aws:iam::111122223333:role/accessGrantsTestRole
```

回應：

```
{"CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "LocationScope": "s3://”
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

Example 註冊自訂位置

```
aws s3control create-access-grants-location \
 --account-id 111122223333 \
 --location-scope s3://DOC-BUCKET-EXAMPLE/ \
 --iam-role-arn arn:aws:iam::123456789012:role/accessGrantsTestRole
```

回應：

```
{"CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb456",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://DOC-BUCKET-EXAMPLE/",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
```

```
}
```

使用 REST API

如需有關管理 S3 Access Grants 執行個體的 Amazon S3 REST API 支援資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列各節：

- CreateAccessGrantsLocation
- DeleteAccessGrantsLocation
- GetAccessGrantsLocation
- ListAccessGrantsLocations
- UpdateAccessGrantsLocation

使用 AWS 軟體開發套件

本節提供如何使用 AWS SDK 註冊位置的範例。

若要使用下列範例，請以您自己的資訊取代 *user input placeholders*。

Java

您可以在 S3 Access Grants 執行個體中註冊預設位置 `s3://` 或自訂位置。務必先建立具有該位置的主要存取權的 IAM 角色，然後確實授予 S3 Access Grants 許可來擔任此角色。

若要使用下列範例命令，請將 *user input placeholders* 取代為您自己的資訊。

Example 註冊預設位置

要求:

```
public void createAccessGrantsLocation() {
CreateAccessGrantsLocationRequest createRequest =
 CreateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.locationScope("s3://")
.iamRoleArn("arn:aws:iam::123456789012:role/accessGrantsTestRole")
.build();
CreateAccessGrantsLocationResponse createResponse =
 s3Control.createAccessGrantsLocation(createRequest);
LOGGER.info("CreateAccessGrantsLocationResponse: " + createResponse);
}
```

回應：

```
CreateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:11.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope=s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

Example 註冊自訂位置

要求:

```
public void createAccessGrantsLocation() {
CreateAccessGrantsLocationRequest createRequest =
 CreateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.locationScope("s3://DOC-BUCKET-EXAMPLE/")
.iamRoleArn("arn:aws:iam::111122223333:role/accessGrantsTestRole")
.build();
CreateAccessGrantsLocationResponse createResponse =
 s3Control.createAccessGrantsLocation(createRequest);
LOGGER.info("CreateAccessGrantsLocationResponse: " + createResponse);
}
```

回應：

```
CreateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=18cfe6fb-eb5a-4ac5-aba9-8d79f04c2012,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/18cfe6fb-eb5a-4ac5-aba9-8d79f04c2666,
LocationScope= s3://test-bucket-access-grants-user123/,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

主題

- 檢視註冊位置的詳細資訊

- 更新註冊的位置
- 刪除註冊的位置

檢視註冊位置的詳細資訊

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、Amazon S3 REST API 和 AWS 開發套件，取得在 S3 存取授權執行個體中註冊之位置的詳細資訊。

使用 S3 主控台

檢視在 S3 Access Grants 執行個體中註冊的位置

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Grants。
3. 在 S3 Access Grants 頁面上，選擇包含您要使用之 S3 Access Grants 執行個體的區域。
4. 選擇執行個體的檢視詳細資訊。
5. 在執行個體的詳細資訊頁面上，選擇位置索引標籤。
6. 尋找您要檢視的註冊位置。若要篩選註冊位置的清單，請使用搜尋方塊。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

若要使用下列範例命令，請以您自己的資訊取代 *user input placeholders*。

Example – 取得註冊位置的詳細資訊

```
aws s3control get-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id default
```

回應：

```
{
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
```

```
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

Example – 列出在 S3 Access Grants 執行個體中註冊的所有位置

若要將結果限於 S3 字首或儲存貯體，您可以選擇使用 `--location-scope s3://bucket-and-or-prefix` 參數。

```
aws s3control list-access-grants-locations \
--account-id 111122223333 \
--region us-east-2
```

回應：

```
{"AccessGrantsLocationsList": [
  {
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "LocationScope": "s3://"
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
     },
  {
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb456",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://DOC-EXAMPLE-BUCKET/prefixA*",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
     }
   ]
  }
```

使用 REST API

如需有關取得已註冊位置的詳細資訊，或列出在 S3 Access Grants 執行個體註冊的所有位置的 Amazon S3 REST API 支援資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列各節：

- GetAccessGrantsLocation

- ListAccessGrantsLocations

使用 AWS 軟體開發套件

本節提供如何使用 AWS SDK 取得 S3 Access Grants 執行個體中已註冊位置的詳細資訊，或列出所有註冊位置的範例。

若要使用下列範例，請以您自己的資訊取代 *user input placeholders*。

Java

Example – 取得註冊位置的詳細資訊

```
public void getAccessGrantsLocation() {
GetAccessGrantsLocationRequest getAccessGrantsLocationRequest =
 GetAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("default")
.build();
GetAccessGrantsLocationResponse getAccessGrantsLocationResponse =
 s3Control.getAccessGrantsLocation(getAccessGrantsLocationRequest);
LOGGER.info("GetAccessGrantsLocationResponse: " + getAccessGrantsLocationResponse);
}
```

回應：

```
GetAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope= s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

Example – 列出 S3 Access Grants 執行個體中的所有註冊位置

若要將結果限於 S3 字首或儲存貯體，您可以選擇在 `LocationScope` 參數中傳遞 S3 URI (例如 `s3://bucket-and-or-prefix`)。

```
public void listAccessGrantsLocations() {
```

```
ListAccessGrantsLocationsRequest listRequest =
 ListAccessGrantsLocationsRequest.builder()
.accountId("111122223333")
.build();

ListAccessGrantsLocationsResponse listResponse =
 s3Control.listAccessGrantsLocations(listRequest);
LOGGER.info("ListAccessGrantsLocationsResponse: " + listResponse);
}
```

回應：

```
ListAccessGrantsLocationsResponse(
AccessGrantsLocationsList=[
ListAccessGrantsLocationsEntry(
CreatedAt=2023-06-07T04:35:11.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope=s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
),
ListAccessGrantsLocationsEntry(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=635f1139-1af2-4e43-8131-a4de006eb456,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/635f1139-1af2-4e43-8131-a4de006eb888,
LocationScope=s3://DOC-EXAMPLE-BUCKET/prefixA*,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
]
)
```

更新註冊的位置

您可以更新在 Amazon S3 存取授權執行個體中註冊的位置的 AWS Identity and Access Management (IAM) 角色。對於您在 S3 Access Grants 中用來註冊位置的每個新 IAM 角色，請務必將 S3 Access Grants 服務主體 (`access-grants.s3.amazonaws.com`) 存取權授予此角色。若要完成此操作，請在您第一次註冊位置時使用的信任政策 JSON 檔案中，為新的 IAM 角色新增一個項目。

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、Amazon S3 REST API 和 AWS 開發套件來更新 S3 存取授權執行個體中的位置。

使用 S3 主控台

更新在 S3 Access Grants 執行個體註冊之位置的 IAM 角色

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Grants。
3. 在 S3 Access Grants 頁面上，選擇包含您要使用之 S3 Access Grants 執行個體的區域。
4. 選擇執行個體的檢視詳細資訊。
5. 在執行個體的詳細資訊頁面上，選擇位置索引標籤。
6. 尋找您要更新的位置。若要篩選位置的清單，請使用搜尋方塊。
7. 選擇您要更新的註冊位置旁的選項按鈕。
8. 更新 IAM 角色，然後選擇儲存變更。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

若要使用下列範例命令，請以您自己的資訊取代 *user input placeholders*。

Example – 更新註冊位置的 IAM 角色

```
aws s3control update-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id 635f1139-1af2-4e43-8131-a4de006eb999 \
--iam-role-arn arn:aws:iam::777788889999:role/accessGrantsTestRole
```

回應：

```
{
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb999",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:777788889999:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://DOC-EXAMPLE-BUCKET/prefixB*",
    "IAMRoleArn": "arn:aws:iam::777788889999:role/accessGrantsTestRole"
}
```

使用 REST API

如需有關在 S3 Access Grants 執行個體中更新位置的 Amazon S3 REST API 支援資訊，請參閱《Amazon Simple Storage Service API 參考》中的 UpdateAccessGrantsLocation。

使用 AWS 軟體開發套件

本節提供如何使用 AWS SDK 更新已註冊位置的 IAM 角色的範例。

若要使用下列範例，請以您自己的資訊取代 *user input placeholders*。

Java

Example – 更新註冊位置的 IAM 角色

```
public void updateAccessGrantsLocation() {
UpdateAccessGrantsLocationRequest updateRequest =
 UpdateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("635f1139-1af2-4e43-8131-a4de006eb999")
.iamRoleArn("arn:aws:iam::777788889999:role/accessGrantsTestRole")
.build();
UpdateAccessGrantsLocationResponse updateResponse =
 s3Control.updateAccessGrantsLocation(updateRequest);
LOGGER.info("UpdateAccessGrantsLocationResponse: " + updateResponse);
}
```

回應：

```
UpdateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=635f1139-1af2-4e43-8131-a4de006eb999,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:777788889999:access-grants/default/
location/635f1139-1af2-4e43-8131-a4de006eb888,
LocationScope=s3://DOC-EXAMPLE-BUCKET/prefixB*,
IAMRoleArn=arn:aws:iam::777788889999:role/accessGrantsTestRole
)
```

刪除註冊的位置

您可以從 Amazon S3 Access Grants 執行個體中刪除位置註冊。刪除位置會從 S3 Access Grants 執行個體取消註冊該位置。

您必須先刪除與此位置相關聯的所有授權，才能從 S3 Access Grants 執行個體移除位置註冊。如需有關如何刪除授權的資訊，請參閱刪除授權。

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、Amazon S3 REST API 和 AWS 開發套件刪除 S3 存取授權執行個體中的位置。

使用 S3 主控台

從 Amazon S3 Access Grants 執行個體刪除位置註冊

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Grants。
3. 在 S3 Access Grants 頁面上，選擇包含您要使用之 S3 Access Grants 執行個體的區域。
4. 選擇執行個體的檢視詳細資訊。
5. 在執行個體的詳細資訊頁面上，選擇位置索引標籤。
6. 尋找您要更新的位置。若要篩選位置的清單，請使用搜尋方塊。
7. 選擇您要刪除的註冊位置旁的選項按鈕。
8. 選擇 Deregister (取消註冊)。
9. 此時會出現對話方塊，警告您此動作無法復原。若要刪除位置，請選擇取消註冊。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

若要使用下列範例命令，請以您自己的資訊取代 *user input placeholders*。

Example – 刪除位置註冊

```
aws s3control delete-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id  a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
 // No response body
```

使用 REST API

如需有關從 S3 Access Grants 執行個體刪除位置的 Amazon S3 REST API 支援資訊，請參閱《Amazon Simple Storage Service API 參考》中的 DeleteAccessGrantsLocation。

使用 AWS 軟體開發套件

本節提供如何使用 AWS SDK 刪除位置的範例。

若要使用下列範例，請以您自己的資訊取代 *user input placeholders*。

Java

Example – 刪除位置註冊

```
public void deleteAccessGrantsLocation() {
DeleteAccessGrantsLocationRequest deleteRequest =
 DeleteAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("a1b2c3d4-5678-90ab-cdef-EXAMPLE11111")
.build();
DeleteAccessGrantsLocationResponse deleteResponse =
 s3Control.deleteAccessGrantsLocation(deleteRequest);
LOGGER.info("DeleteAccessGrantsLocationResponse: " + deleteResponse);
}
```

回應：

```
DeleteAccessGrantsLocationResponse()
```

建立授權

您必須先在 Amazon S3 Access Grants 執行個體中至少註冊一個位置，才能建立存取授權。存取授權可給予承授者存取註冊位置的許可。

受權者可以是 AWS Identity and Access Management (IAM) 使用者或角色，也可以是目錄使用者或群組。目錄使用者是您新增至與 S3 Access Grants 執行個體相關聯之 AWS IAM Identity Center 執行個體的公司目錄或外部身分來源的使用者。若要從 IAM Identity Center 為特定使用者或群組建立授權，請在 IAM Identity Center 尋找 IAM Identity Center 用來識別該使用者的 GUID，例如 `a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`。

您可以授予儲存貯體、字首或物件的存取權。Amazon S3 中的字首是物件索引鍵名稱開頭的一串字元，用於組織儲存貯體中的物件。字首可以是任何允許的字元字串，例如儲存貯體中以 `engineering/` 字首開頭的物件索引鍵名稱。

子字首

授予註冊位置的存取權時，您可以使用 Subprefix 欄位將範圍縮小為儲存貯體內的特定字首，或是儲存貯體中的特定物件。

您無法建立預設位置 s3:// 的存取授權，這樣會讓承授者存取區域中的所有儲存貯體。如果您選擇預設 s3:// 位置作為授權位置，則必須使用 Subprefix 欄位指定下列其中一項來縮小授權範圍：

- 儲存貯體 — s3://*bucket*/*
- 儲存貯體內的字首 — s3://*bucket*/*prefix**
- 字首內的字首 — s3://*bucket*/*prefixA*/*prefixB**
- 物件 — s3://*bucket*/*object-key-name*

如果您建立的存取授權中註冊位置是儲存貯體，則可在 Subprefix 欄位中傳遞下列其中一項：

- 儲存貯體內的字首 — *prefix**
- 字首內的字首 — *prefixA/prefixB**
- 物件 — /*object-key-name*

Amazon S3 主控台中顯示的授權範圍或 API 或 AWS Command Line Interface (AWS CLI) 回應中傳回的授權範圍是將位置路徑與. GrantScope Subprefix 請確定此串連路徑正確映射至您要授予存取權的 S3 儲存貯體、字首或物件。

如果您要建立的存取授權僅授予一個物件的存取權，請在 API 呼叫或 CLI 命令中指定 s3PrefixType 為 Object。

> **Note**
>
> 如果儲存貯體不存在，則您無法建立該儲存貯體的授權。不過，您可以為尚不存在的字首建立授權。

您可以使用 Amazon S3 主控台 AWS CLI、Amazon S3 REST API 和 AWS 開發套件來建立存取授與。

使用 S3 主控台

建立存取授權

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Grants。
3. 在 S3 Access Grants 頁面上，選擇包含您要使用之 S3 Access Grants 執行個體的區域。

   如果您是第一次使用 S3 Access Grants 執行個體，請確定您已完成步驟 2 - 註冊位置，並瀏覽至設定 Access Grants 執行個體精靈的步驟 3。如果您已有 S3 Access Grants 執行個體，請選擇檢視詳細資訊，然後從授權索引標籤選擇建立授權。

   a. 在授權範圍區段中，選取或輸入註冊位置。

      如果您選取預設的 `s3://` 位置，請使用子字首方塊縮小存取授權的範圍。如需詳細資訊，請參閱子字首。如果您只授予一個物件的存取權，請選取授權範圍是物件。

   b. 在權限和存取底下，選取許可層級讀取或寫入，或兩者都選取。

      然後選擇承授者類型。如果您已將公司目錄新增至 IAM Identity Center，並將此 IAM Identity Center 執行個體與 S3 Access Grants 執行個體建立關聯，則可以選擇 IAM Identity Center 的目錄身分。如果您選擇此選項，請從 IAM Identity Center 取得使用者或群組的 ID，然後在此區段中輸入。

      如果承授者類型是 IAM 使用者或角色，請選擇 IAM 主體。在 IAM 主體類型下，選擇使用者或角色。然後，在 IAM 主體使用者下，從清單中選擇或輸入身分 ID。

   c. 若要建立 S3 Access Grants 授權，請選擇下一步或建立授權。
4. 如果下一步或建立授權為停用狀態：

   無法建立授權

   - 您可能需要先在 S3 Access Grants 執行個體中註冊位置。
   - 您可能沒有建立存取授權的 `s3:CreateAccessGrant` 許可。請聯絡您的帳戶管理員。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

下列範例說明如何建立 IAM 主體的存取授權請求，以及如何為公司目錄使用者或群組建立存取授權請求。

若要使用下列範例命令，請將 *user input placeholders* 取代為您自己的資訊。

> **Note**
>
> 如果您要建立的存取授權僅授予一個物件的存取權，請包含必要的參數 `--s3-prefix-type Object`。

Example 建立 IAM 主體的存取授權請求

```
aws s3control create-access-grant \
--account-id 111122223333 \
--access-grants-location-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222 \
--access-grants-location-configuration S3SubPrefix=prefixB* \
--permission READ \
--grantee GranteeType=IAM,GranteeIdentifier=arn:aws:iam::123456789012:user/data-consumer-3
```

Example 建立存取授權回應

```
{"CreatedAt": "2023-05-31T18:41:34.663000+00:00",
    "AccessGrantId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/grant/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Grantee": {
        "GranteeType": "IAM",
        "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
    },
    "AccessGrantsLocationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "AccessGrantsLocationConfiguration": {
        "S3SubPrefix": "prefixB*"
    },
    "GrantScope": "s3://DOC-BUCKET-EXAMPLE/prefix*",
    "Permission": "READ"
}
```

建立目錄使用者或群組的存取授權請求

若要建立目錄使用者或群組的存取授權請求，您必須先執行下列其中一個命令，取得目錄使用者或群組的 GUID。

Example 取得目錄使用者或群組的 GUID

您可以透過 IAM 身分中心主控台或使用或 AWS SDK，找到 IAM 身分中心使用者的 GUID。 AWS CLI 下列命令會列出所指定 IAM Identity Center 執行個體中的使用者，包括其名稱和識別符。

```
aws identitystore list-users --identity-store-id d-1a2b3c4d1234
```

此命令會列出所指定 IAM Identity Center 執行個體中的群組。

```
aws identitystore list-groups --identity-store-id d-1a2b3c4d1234
```

Example 建立目錄使用者或群組的存取授權

此命令類似於為 IAM 使用者或角色建立授權，但承授者類型為 DIRECTORY_USER 或 DIRECTORY_GROUP，且承授者識別符是目錄使用者或群組的 GUID。

```
aws s3control create-access-grant \
--account-id 123456789012 \
--access-grants-location-id default \
--access-grants-location-configuration S3SubPrefix="DOC-EXAMPLE-BUCKET/rafael/*" \
--permission READWRITE \
--grantee GranteeType=DIRECTORY_USER,GranteeIdentifier=83d43802-00b1-7054-db02-
f1d683aacba5 \
```

使用 REST API

如需有關管理存取授權的 Amazon S3 REST API 支援資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列各節：

- CreateAccessGrant
- DeleteAccessGrant
- GetAccessGrant
- ListAccessGrants

使用 AWS 軟體開發套件

本節提供如何使用 AWS SDK 建立存取授權的範例。

Java

若要使用下列範例，請將 *user input placeholders* 取代為您自己的資訊：

> **Note**
>
> 如果您要建立的存取授權僅授予一個物件的存取權，請包含必要的參數 `.s3PrefixType(S3PrefixType.Object)`。

Example 建立存取授權請求

```
public void createAccessGrant() {
CreateAccessGrantRequest createRequest = CreateAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa")
.permission("READ")
.accessGrantsLocationConfiguration(AccessGrantsLocationConfiguration.builder().s3SubPrefix("
.grantee(Grantee.builder().granteeType("IAM").granteeIdentifier("arn:aws:iam::111122223333:u
data-consumer-3").build())
.build();
CreateAccessGrantResponse createResponse =
 s3Control.createAccessGrant(createRequest);
LOGGER.info("CreateAccessGrantResponse: " + createResponse);
}
```

Example 建立存取授權回應

```
CreateAccessGrantResponse(
CreatedAt=2023-06-07T05:20:26.330Z,
AccessGrantId=a1b2c3d4-5678-90ab-cdef-EXAMPLE33333,
AccessGrantArn=arn:aws:s3:us-east-2:444455556666:access-grants/default/grant/
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333,
Grantee=Grantee(
GranteeType=IAM,
GranteeIdentifier=arn:aws:iam::111122223333:user/data-consumer-3
),
AccessGrantsLocationId=a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa,
AccessGrantsLocationConfiguration=AccessGrantsLocationConfiguration(
S3SubPrefix=prefixB*
),
GrantScope=s3://DOC-BUCKET-EXAMPLE/prefixB,
```

```
Permission=READ
)
```

主題

- 檢視授權
- 刪除授權

檢視授權

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、Amazon S3 REST API 和 AWS 開發套件，在 Amazon S3 存取授予執行個體中檢視存取授與的詳細資訊。

使用 S3 主控台

檢視存取授權的詳細資訊

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Grants。
3. 在 S3 Access Grants 頁面上，選擇包含您要使用之 S3 Access Grants 執行個體的區域。
4. 選擇執行個體的檢視詳細資訊。
5. 在詳細資訊頁面上，選擇授權索引標籤。
6. 在授權區段中，尋找您要檢視的存取授權。若要篩選授權清單，請使用搜尋方塊。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

若要使用下列範例命令，請將 *user input placeholders* 取代為您自己的資訊。

Example – 取得存取授權的詳細資訊

```
aws s3control get-access-grant \
--account-id 111122223333 \
--access-grant-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222
```

回應：

```
{
    "CreatedAt": "2023-05-31T18:41:34.663000+00:00",
    "AccessGrantId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "Grantee": {
        "GranteeType": "IAM",
        "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
    },
    "Permission": "READ",
    "AccessGrantsLocationId": "12a6710f-5af8-41f5-b035-0bc795bf1a2b",
    "AccessGrantsLocationConfiguration": {
        "S3SubPrefix": "prefixB*"
    },
    "GrantScope": "s3://DOC-EXAMPLE-BUCKET/"
}
```

Example – 列出 S3 Access Grants 執行個體中的所有存取授權

您可以選擇性地使用下列參數將結果限制為 S3 前置詞或 AWS Identity and Access Management (IAM) 身分：

- 子字首 – `--grant-scope s3://bucket-name/prefix*`
- IAM 身分 – `--grantee-type IAM` 和 `--grantee-identifier arn:aws:iam::123456789000:role/accessGrantsConsumerRole`

```
aws s3control list-access-grants \
--account-id 111122223333
```

回應：

```
{
    "AccessGrantsList": [{"CreatedAt": "2023-06-14T17:54:46.542000+00:00",
            "AccessGrantId": "dd8dd089-b224-4d82-95f6-975b4185bbaa",
            "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/dd8dd089-b224-4d82-95f6-975b4185bbaa",
            "Grantee": {
                "GranteeType": "IAM",
                "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
            },
```

```
            "Permission": "READ",
            "AccessGrantsLocationId": "23514a34-ea2e-4ddf-b425-d0d4bfcarda1",
            "GrantScope": "s3://DOC-EXAMPLE-BUCKET/prefixA*"
        },
        {"CreatedAt": "2023-06-24T17:54:46.542000+00:00",
            "AccessGrantId": "ee8ee089-b224-4d72-85f6-975b4185a1b2",
            "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/ee8ee089-b224-4d72-85f6-975b4185a1b2",
            "Grantee": {
                "GranteeType": "IAM",
                "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-9"
            },
            "Permission": "READ",
            "AccessGrantsLocationId": "12414a34-ea2e-4ddf-b425-d0d4bfcacao0",
            "GrantScope": "s3://DOC-EXAMPLE-BUCKET/prefixB*"
        },

    ]
}
```

使用 REST API

您可以使用 Amazon S3 API 操作來檢視存取授權的詳細資訊，並列出 S3 Access Grants 執行個體中的所有存取授權。如需有關管理存取授權的 REST API 支援資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列各節：

- GetAccessGrant
- ListAccessGrants

使用 AWS 軟體開發套件

本節提供如何使用 AWS SDK 取得存取授與詳細資料的範例。

若要使用下列範例，請以您自己的資訊取代 *user input placeholders*。

Java

Example – 取得存取授權的詳細資訊

```
public void getAccessGrant() {
GetAccessGrantRequest getRequest = GetAccessGrantRequest.builder()
.accountId("111122223333")
```

```
.accessGrantId("a1b2c3d4-5678-90ab-cdef-EXAMPLE22222")
.build();
GetAccessGrantResponse getResponse = s3Control.getAccessGrant(getRequest);
LOGGER.info("GetAccessGrantResponse: " + getResponse);
}
```

回應：

```
GetAccessGrantResponse(
CreatedAt=2023-06-07T05:20:26.330Z,
AccessGrantId=a1b2c3d4-5678-90ab-cdef-EXAMPLE22222,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant-fd3a5086-42f7-4b34-9fad-472e2942c70e,
Grantee=Grantee(
GranteeType=IAM,
GranteeIdentifier=arn:aws:iam::111122223333:user/data-consumer-3
),
Permission=READ,
AccessGrantsLocationId=12a6710f-5af8-41f5-b035-0bc795bf1a2b,
AccessGrantsLocationConfiguration=AccessGrantsLocationConfiguration(
S3SubPrefix=prefixB*
),
GrantScope=s3://DOC-EXAMPLE-BUCKET/
)
```

Example – 列出 S3 Access Grants 執行個體中的所有存取授權

您可以選擇使用這些參數將結果限於 S3 字首或 IAM 身分：

- 範圍 – `GrantScope=s3://bucket-name/prefix*`
- 承授者 – `GranteeType=IAM` 和 `GranteeIdentifier=arn:aws:iam::111122223333:role/accessGrantsConsumerRole`

```
public void listAccessGrants() {
ListAccessGrantsRequest listRequest = ListAccessGrantsRequest.builder()
.accountId("111122223333")
.build();
ListAccessGrantsResponse listResponse = s3Control.listAccessGrants(listRequest);
LOGGER.info("ListAccessGrantsResponse: " + listResponse);
}
```

回應：

```
ListAccessGrantsResponse(
AccessGrantsList=[
ListAccessGrantEntry(
CreatedAt=2023-06-14T17:54:46.540z,
AccessGrantId=dd8dd089-b224-4d82-95f6-975b4185bbaa,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/dd8dd089-b224-4d82-95f6-975b4185bbaa,
Grantee=Grantee(
GranteeType=IAM, GranteeIdentifier= arn:aws:iam::111122223333:user/data-consumer-3
),
Permission=READ,
AccessGrantsLocationId=23514a34-ea2e-4ddf-b425-d0d4bfcarda1,
GrantScope=s3://DOC-EXAMPLE-BUCKET/prefixA
),
ListAccessGrantEntry(
CreatedAt=2023-06-24T17:54:46.540Z,
AccessGrantId=ee8ee089-b224-4d72-85f6-975b4185a1b2,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/ee8ee089-b224-4d72-85f6-975b4185a1b2,
Grantee=Grantee(
GranteeType=IAM, GranteeIdentifier= arn:aws:iam::111122223333:user/data-consumer-9
),
Permission=READ,
AccessGrantsLocationId=12414a34-ea2e-4ddf-b425-d0d4bfcacao0,
GrantScope=s3://DOC-EXAMPLE-BUCKET/prefixB*
)
]
)
```

刪除授權

您可以從 Amazon S3 Access Grants 執行個體刪除存取授權。您無法復原存取授權刪除動作。刪除存取授權後，承授者將無法再存取您的 Amazon S3 資料。

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、Amazon S3 REST API 和 AWS 開發套件刪除存取授與。

使用 S3 主控台

刪除存取授權

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Access Grants。
3. 在 S3 Access Grants 頁面上，選擇包含您要使用之 S3 Access Grants 執行個體的區域。
4. 選擇執行個體的檢視詳細資訊。
5. 在詳細資訊頁面上，選擇授權索引標籤。
6. 搜尋您要刪除的授權。找到授權後，選擇該授權旁的選項按鈕。
7. 選擇刪除。此時會出現對話方塊，警告您此動作無法復原。再次選擇刪除，以刪除授權。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

若要使用下列範例命令，請以您自己的資訊取代 *user input placeholders*。

Example – 刪除存取授權

```
aws s3control delete-access-grant \
--account-id 111122223333 \
--access-grant-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111

// No response body
```

使用 REST API

如需有關管理存取授權的 Amazon S3 REST API 支援資訊，請參閱《Amazon Simple Storage Service API 參考》中的 DeleteAccessGrant。

使用 AWS 軟體開發套件

本節提供如何使用 AWS SDK 刪除存取授與的範例。若要使用下列範例，請以您自己的資訊取代 *user input placeholders*。

Java

Example – 刪除存取授權

```
public void deleteAccessGrant() {
DeleteAccessGrantRequest deleteRequest = DeleteAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantId("a1b2c3d4-5678-90ab-cdef-EXAMPLE11111")
.build();
DeleteAccessGrantResponse deleteResponse =
 s3Control.deleteAccessGrant(deleteRequest);
LOGGER.info("DeleteAccessGrantResponse: " + deleteResponse);
}
```

回應：

```
DeleteAccessGrantResponse()
```

透過 S3 Access Grants 請求存取 Amazon S3 資料

使用 Amazon S3 Access Grants 建立存取授權以授予 AWS Identity and Access Management (IAM) 主體、您的公司目錄身分或授權應用程式存取 S3 資料後，受權者可以請求登入資料以存取此資料。

當應用程式或 AWS 服務 使用 `GetDataAccess` API 操作要求 S3 存取授與代表受權者存取 S3 資料時，S3 存取授權會首先驗證您是否已授與此資料的身分存取權。然後，S3 存取授權會使用 AssumeRoleAPI 作業取得臨時登入資料權杖，並將其分配給要求者。這個臨時憑證權杖是 AWS Security Token Service (AWS STS) 權杖。

`GetDataAccess` 請求必須包含 `target` 參數，用來指定臨時憑證套用的 S3 資料範圍。此 `target` 範圍可與授權範圍相同，也可以是該範圍的子集，但 `target` 範圍必須在授予請求者的授權範圍內。請求也必須指定 `permission` 參數，以指出臨時憑證的許可層級，也就是 `READ`、`WRITE` 或 `READWRITE`。

請求者可以在其憑證請求中指定臨時權杖的許可層級。請求者可以使用 `privilege` 參數，在授權範圍內縮小或加大臨時憑證的存取範圍。`privilege` 參數的預設值為 `Default`，這表示所傳回憑證的目標範圍是原始授權範圍。`privilege` 的另一個可能的值為 `Minimal`。如果 `target` 範圍比原始授權範圍小，則臨時憑證的範圍也會縮小，以符合 `target` 範圍，前提是 `target` 範圍在授權範圍內。

下表詳細說明 `privilege` 參數對兩個授權的影響。一個授權的範圍是 `S3://example-s3-bucket1/bob/*`，包括 *example-s3-bucket1* 儲存貯體中的整個 bob/ 字首。另一個授權的範圍

是 S3://*example-s3-bucket1*/bob/reports/*，只包括 *example-s3-bucket1* 儲存貯體中的 bob/reports/ 字首。

授權範圍	請求範圍	權限	傳回範圍	Effect
S3://*example-s3-bucket1* /bob/*	*example-s3-bucket1* /bob/*	Default	*example-s3-bucket1* /bob/*	請求者可以存取 *example-s3-bucket1* 儲存貯體中，具有以字首 *bob/* 開頭的索引鍵名稱的所有物件。
S3://*example-s3-bucket1* /bob/*	*example-s3-bucket1* /bob/	Minimal	*example-s3-bucket1* /bob/	若字首名稱 bob/ 後沒有萬用字元 *，則請求者只能存取 *example-s3-bucket1* 儲存貯體中名為 bob/ 的物件。這類物件並不常見。請求者無法存取任何其他物件，包括具有以 bob/ 字首開頭的索引鍵名稱的物件。
S3://*example-s3-bucket1* /bob/*	*example-s3-bucket1* /bob/images/*	Minimal	*example-s3-bucket1* /bob/images/*	請求者可以存取 *example-s3-bucket1* 儲存貯體中，具有以字首 *bob/images/** 開頭的索引鍵名稱的所有物件。
S3://*example-s3-bucket1* /bob/reports/*	*example-s3-bucket1* /bob/reports/file.txt	Default	*example-s3-bucket1* /bob/reports/*	請求者可以存取 *example-s3-bucket1* 儲存貯體中，具有以 bob/reports 字首開頭的索引鍵名稱的所有物件，這是相符授權的範圍。

授權範圍	請求範圍	權限	傳回範圍	Effect
S3://*example-s3-bucket1* /bob/reports/*	*example-s3-bucket1* /bob/reports/file.txt	Minimal	*example-s3-bucket1* /bob/reports/file.txt	請求者只能存取 *example-s3-bucket1* 儲存貯體中具有索引鍵名稱 bob/reports/file.txt 的物件。請求者無法存取任何其他物件。

durationSeconds 參數會設定臨時憑證的持續時間 (以秒為單位)。預設值為 3600 秒 (1 小時)，但請求者 (承授者) 可以指定從 900 秒 (15 分鐘) 到最長 43200 秒 (12 小時) 的範圍。若承授者請求的值高於此上限，則請求會失敗。

> **Note**
>
> 在您的臨時權杖請求中，如果位置是物件，請將您請求中的 targetType 參數值設定為 Object。只有在位置是物件且權限層級為 Minimal 時，才需要此參數。若位置是儲存貯體或字首，則不需要指定此參數。

如需詳細資訊，請參閱 Amazon 簡易儲存服務 API 參考GetDataAccess中的。

您可以使用 AWS Command Line Interface (AWS CLI)、Amazon S3 REST API 和 AWS 開發套件來請求臨時登入資料。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

若要使用下列範例命令，請以您自己的資訊取代 *user input placeholders*。

Example 請求臨時憑證

要求:

```
aws s3control get-data-access \
--account-id 111122223333 \
```

```
--target s3://example-s3-bucket/prefixA* \
--permission READ \
--privilege Default \
--region us-east-2
```

回應：

```
{
"Credentials": {
 "AccessKeyId": "Example-key-id",
 "SecretAccessKey": "Example-access-key",
 "SessionToken": "Example-session-token",
"Expiration": "2023-06-14T18:56:45+00:00"},
    "MatchedGrantTarget": "s3://example-s3-bucket/prefixA**"
}
```

使用 REST API

如需從 S3 存取授權請求臨時登入資料的 Amazon S3 REST API 支援的相關資訊，請參閱 Amazon 簡單儲存服務 API 參考GetDataAccess中的。

使用 AWS 軟體開發套件

本節提供受權者如何使用 AWS SDK 從 S3 存取授權請求臨時登入資料的範例。

Java

下列程式碼範例會傳回承授者用來存取 S3 資料的臨時憑證。若要使用此程式碼範例，請將 *user input placeholders* 取代為您自己的資訊。

Example 取得臨時憑證

要求:

```
public void getDataAccess() {
GetDataAccessRequest getDataAccessRequest = GetDataAccessRequest.builder()
.accountId("111122223333")
.permission(Permission.READ)
.privilege(Privilege.MINIMAL)
.target("s3://example-s3-bucket/prefixA*")
.build();
GetDataAccessResponse getDataAccessResponse =
 s3Control.getDataAccess(getDataAccessRequest);
```

```
LOGGER.info("GetDataAccessResponse: " + getDataAccessResponse);
}
```

回應：

```
GetDataAccessResponse(
Credentials=Credentials(
AccessKeyId="Example-access-key-id",
SecretAccessKey="Example-secret-access-key",
SessionToken="Example-session-token",
Expiration=2023-06-07T06:55:24Z
))
```

透過存取授權存取 S3 資料

承授者透過其存取授權取得臨時憑證後，可以使用這些臨時憑證呼叫 Amazon S3 API 操作來存取您的資料。

受權者可以使用 AWS Command Line Interface (AWS CLI)、 AWS 開發套件和 Amazon S3 REST API 存取 S3 資料。

使用 AWS CLI

承授者從 S3 Access Grants 取得其臨時憑證後，就可以設定包含這些憑證的設定檔來擷取資料。

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

若要使用下列範例命令，請將 *user input placeholders* 取代為您自己的資訊。

Example – 設定設定檔

```
aws configure set aws_access_key_id "$accessKey" --profile access-grants-consumer-access-profile
aws configure set aws_secret_access_key "$secretKey" --profile access-grants-consumer-access-profile
aws configure set aws_session_token "$sessionToken" --profile access-grants-consumer-access-profile
```

若要使用下列範例命令，請以您自己的資訊取代 *user input placeholders*。

Example – 取得 S3 資料

受權者可以使用命get-object AWS CLI 令來存取資料。受權者也可以使用put-objectls、和其他 S3 AWS CLI 命令。

```
aws s3api get-object \
--bucket example-s3-bucket1 \
--key myprefix \
--region us-east-2 \
--profile access-grants-consumer-access-profile
```

使用 AWS 軟體開發套件

本節提供承授者如何使用 AWS SDK 存取 S3 資料的範例。

Java

如需如何使用臨時登入資料取得 S3 資料的範例，請參閱如何使用 AWS 開發套件和 Amazon S3 程式碼範例取得物件 AWS SDK for Java 2.x。

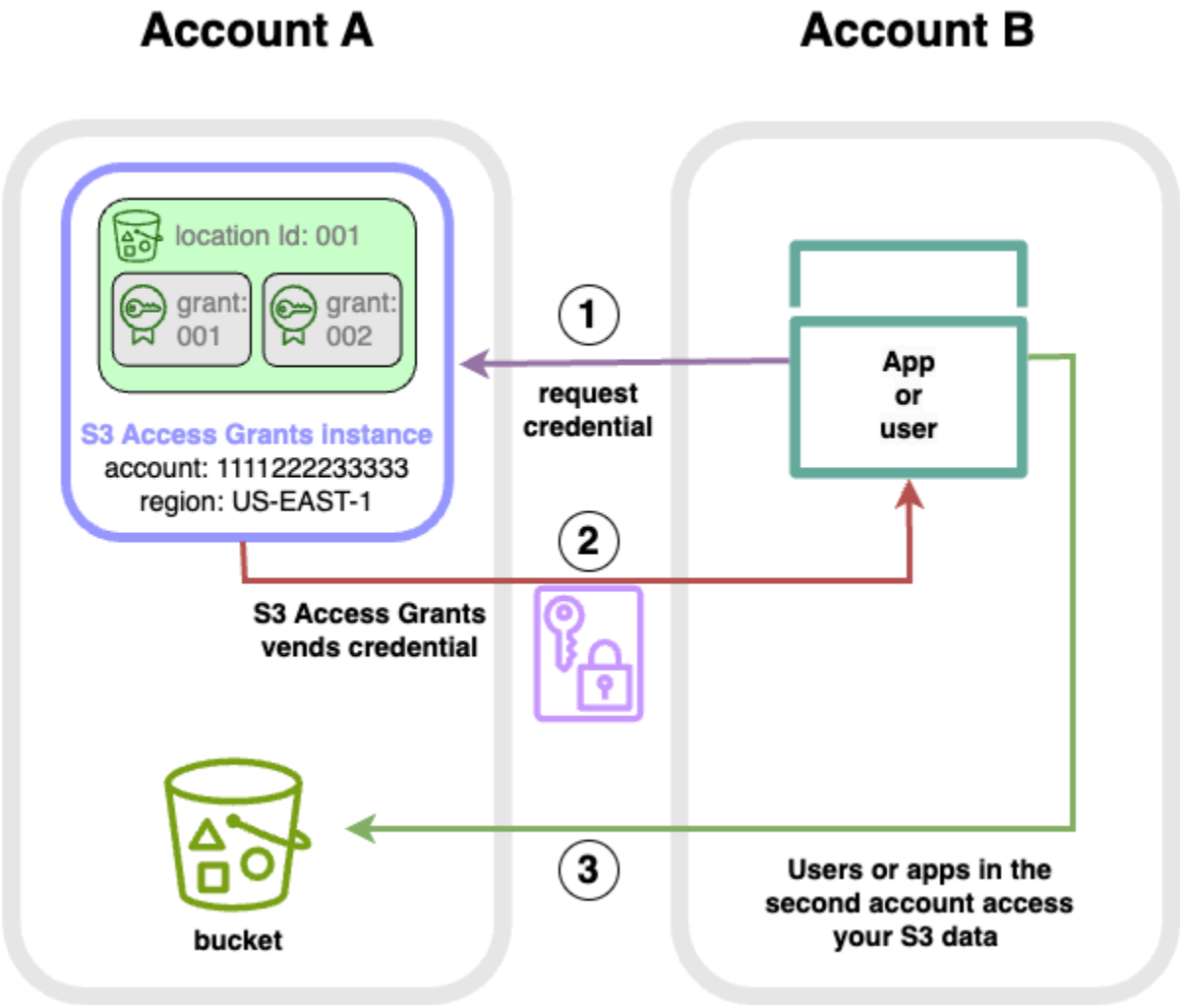
S3 Access Grants 跨帳户存取權

使用 S3 存取授權，您可以授與 Amazon S3 資料存取權給下列項目：

- AWS Identity and Access Management (IAM) 您帳戶內的身分識別
- 其他 AWS 帳戶中的 IAM 身分
- AWS IAM Identity Center 執行個體中的目錄使用者或群組

首先，設定另一個帳戶的跨帳戶存取權。這包括使用資源政策授予對 S3 存取授與執行個體的存取權。然後，使用授權授予對 S3 資料 (儲存貯體、前置詞或物件) 的存取權。

設定跨帳戶存取後，另一個帳戶可以向 S3 存取授權請求對 Amazon S3 資料的臨時存取登入資料。下圖顯示透過 S3 存取授權跨帳戶 S3 存取的使用者流程：



1. 第二個帳戶 (B) 中的使用者或應用程式從帳戶 (A) 中的 S3 存取授予執行個體 (存放 Amazon S3 資料) 請求登入資料。如需詳細資訊，請參閱 透過 S3 Access Grants 請求存取 Amazon S3 資料。
2. 如果有授權可讓第二個帳戶存取您的 Amazon S3 資料，則帳戶中的 S3 存取授權執行個體 (A) 會傳回臨時登入資料。如需詳細資訊，請參閱 the section called “建立授權”。
3. 第二個帳戶 (B) 中的使用者或應用程式會使用由 S3 Access 授予授權的登入資料存取您帳戶 (A) 中的 S3 資料。

設定 S3 存取權授予跨帳戶存取

若要透過 S3 存取授權授與跨帳戶 S3 存取權，請遵循下列步驟：

- 步驟 1：在您的帳戶中設定 S3 存取授與執行個體，例如存放 S3 資料的帳戶 ID 111122223333。

- 步驟 2：設定帳戶中 S3 Access Grants 執行個體的資源政策，111122223333以授予第二個帳戶的存取權，例如帳戶 ID 444455556666。
- 步驟 3：在第二個帳戶中設定 IAM 主體的 IAM 許可，以444455556666向您帳戶中的 S3 Access 授予執行個體請求登入資料111122223333。
- 步驟 4：在您的帳戶中建立授權111122223333，讓第二個帳戶中的 IAM 主體444455556666存取您帳戶中的某些 S3 資料111122223333。

步驟 1：在您的帳戶中設定 S3 存取授與執行個體

首先，您的帳戶中必須有 S3 存取授與執行個體，111122223333才能管理對 Amazon S3 資料的存取。您必須在每個儲存您要共用的 S3 資料的 AWS 區域 位置建立 S3 存取授與執行個體。如果您要共用多個資料 AWS 區域，請針對每個步驟重複上述每個設定步驟 AWS 區域。如果您的 S3 資料存放 AWS 區域 位置已有 S3 存取授與執行個體，請繼續執行下一個步驟。如果您尚未設定 S3 存取授與執行個體，請參建立 S3 Access Grants 執行個體閱完成此步驟。

步驟 2：設定 S3 Access 授與執行個體的資源政策，以授予跨帳戶存取權

在您的帳戶中建立 S3 Access Grants 執行個體以進行跨帳戶存取之後，111122223333請為帳戶中的 S3 Access Grants 執行個體設定以資源為基礎的政策，以授111122223333予跨帳戶存取權。S3 Access Grants 執行個體本身可支援資源型政策。有了正確的資源型政策，您可以將 AWS Identity and Access Management (IAM) 使用者或其他角色的存取權授 AWS 帳戶 予 S3 Access Grants 執行個體。跨帳戶存取僅授予下列權限 (動作)：

- `s3:GetAccessGrantsInstanceForPrefix`— 使用者、角色或應用程式可擷取包含特定前置詞的 S3 Access Grants 執行個體。
- `s3:ListAccessGrants`
- `s3:ListAccessLocations`
- `s3:GetDataAccess`— 使用者、角色或應用程式可以根據您透過 S3 Access Grants 授予的存取權限要求臨時登入資料。使用這些憑證即可存取您已取得存取權的 S3 資料。

您可以選擇要將這些當中的哪些許可包含在資源政策中。S3 Access Grants 執行個體上的這個資源政策是一般的資源型政策，支援 IAM 政策語言支援的一切。在相同的政策中，您可以授予帳戶中特定 IAM 身分的存取權111122223333，例如使用`aws:PrincipalArn`條件，但不需要使用 S3 存取授權來執行此操作。而是在 S3 Access Grants 執行個體中，您可以從您的帳戶以及其他帳戶為個別 IAM 身分建立授權。透過 S3 存取授與管理每個存取授與，您可以擴展許可。

如果您已經使用 AWS Resource Access Manager(AWS RAM)，則可以使用它與其他帳號或組織內部共用s3:AccessGrants資源。如需詳細資訊，請參閱使用共用 AWS 資源。如果您不使用 AWS RAM，您也可以使用 S3 存取授予 API 操作或 AWS Command Line Interface (AWS CLI) 來新增資源政策。

使用 S3 主控台

建議您使用 AWS Resource Access Manager (AWS RAM) Console 與其他帳戶或組織內部共用您的s3:AccessGrants資源。若要跨帳戶共用 S3 存取授與，請執行下列動作：

若要設定 S3 存取授與執行個體資源政策：

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 從選取 AWS 區域 器 AWS 區域 中選取。
3. 在左側導覽窗格中，選取存取權授與。
4. 在 [存取權授與執行個體] 頁面的 [此帳戶中的執行個體] 區段中，選取共用執行個體。這會將您重定向到 AWS RAM 控制台。
5. 選取 [建立資源共用]。
6. 依照下列 AWS RAM 步驟建立資源共用。如需詳細資訊，請參閱在中建立資源共用 AWS RAM。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

您可以使用 put-access-grants-instance-resource-policy CLI 命令新增資源策略。

如果您想要將 S3 Access Grants 執行個體的跨帳戶存取權授予第二個帳戶111122223333444455556666，則您帳戶中 S3 Access Grants 執行個體的資源政策111122223333應授予第二個帳戶執行下列動作的444455556666權限：

- s3:ListAccessGrants
- s3:ListAccessGrantsLocations
- s3:GetDataAccess
- s3:GetAccessGrantsInstanceForPrefix

在 S3 存取授與執行個體資源政策中，將 S3 存取授與執行個體的 ARN 指定為Resource，第二個帳戶指定444455556666Principal為。若要使用下列範例，請以您自己的資訊取代使用#######。

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

若要新增或更新 S3 Access 授與執行個體資源政策，請使用下列命令。當您使用下列範例指令時，請以您自己*user input placeholders*的資訊取代。

Example 新增或更新 S3 存取授與執行個體資源政策

```
aws s3control put-access-grants-instance-resource-policy \
--account-id 111122223333 \
--policy file://resourcePolicy.json \
--region us-east-2
{
 "Policy": "{\n
   \"Version\": \"2012-10-17\",\n
   \"Statement\": [{\n
  \"Effect\": \"Allow\",\n
  \"Principal\": {\n
    \"AWS\": \"444455556666\"\n
  },\n
  \"Action\": [\n
    \"s3:ListAccessGrants\",\n
    \"s3:ListAccessGrantsLocations\",\n
    \"s3:GetDataAccess\",\n
```

```
    \"s3:GetAccessGrantsInstanceForPrefix\"\n
   ],\n
   \"Resource\": \"arn:aws:s3:us-east-2:111122223333:access-grants/default\"\n
     }\n
    ]\n
    }\n",
  "CreatedAt": "2023-06-16T00:07:47.473000+00:00"
 }
```

Example 取得 S3 Access Grants 資源政策

您也可以使用 CLI 取得或刪除 S3 存取授與執行個體的資源政策。

若要取得 S3 存取授與資源政策，請使用下列範例命令。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control get-access-grants-instance-resource-policy \
--account-id 111122223333 \
--region us-east-2

{
"Policy": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",\"Principal
\":{\"AWS\":\"arn:aws:iam::111122223333:root\"},\"Action\":[\"s3:ListAccessGrants\",
\"s3:ListAccessGrantsLocations\",\"s3:GetDataAccess\"],\"Resource\":\"arn:aws:s3:us-
east-2:111122223333:access-grants/default\"}]}",
"CreatedAt": "2023-06-16T00:07:47.473000+00:00"
}
```

Example 刪除 S3 Access Grants 資源政策

若要刪除 S3 存取授與資源政策，請使用下列範例命令。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control delete-access-grants-instance-resource-policy \
--account-id 111122223333 \
--region us-east-2

// No response body
```

使用 REST API

您可以使用 PutAccessGrantsInstanceResourcePolicy API 新增資源策略。

如果您想要將 S3 Access Grants 執行個體的跨帳戶存取權授予第二個帳戶 111122223333444455556666，則您帳戶中 S3 Access Grants 執行個體的資源政策111122223333應授予第二個帳戶執行下列動作的444455556666權限：

- s3:ListAccessGrants
- s3:ListAccessGrantsLocations
- s3:GetDataAccess
- s3:GetAccessGrantsInstanceForPrefix

在 S3 存取授與執行個體資源政策中，將 S3 存取授與執行個體的 ARN 指定為Resource，第二個帳戶指定444455556666Principal為。若要使用下列範例，請以您自己的資訊取代使用#######。

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

然後，您可以使用 PutAccessGrantsInstanceResourcePolicy API 來設定原則。

如需 REST API 支援以更新、取得或刪除 S3 存取授與執行個體的資源政策的相關資訊，請參閱 Amazon 簡單儲存服務 API 參考中的以下各節：

- PutAccessGrantsInstanceResourcePolicy
- GetAccessGrantsInstanceResourcePolicy
- DeleteAccessGrantsInstanceResourcePolicy

使用 AWS 軟體開發套件

本節提供 AWS SDK 範例，說明如何設定 S3 存取授權資源政策，以授予第二個 AWS 帳戶存取部分 S3 資料的權限。

Java

新增、更新、取得或刪除資源政策，以管理對 S3 Access Rents 執行個體的跨帳户存取權。

Example 新增或更新 S3 存取授與執行個體資源政策

如果您想要將 S3 Access Grants 執行個體的跨帳戶存取權授予第二個帳戶 111122223333444455556666，則您帳戶中 S3 Access Grants 執行個體的資源政策111122223333應授予第二個帳戶執行下列動作的444455556666權限：

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

在 S3 存取授與執行個體資源政策中，將 S3 存取授與執行個體的 ARN 指定為`Resource`，第二個帳戶指定444455556666`Principal`為。若要使用下列範例，請以您自己的資訊取代使用*#######*。

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
```

```
}
```

若要新增或更新 S3 Access 授與執行個體資源政策，請使用下列程式碼範例：

```
public void putAccessGrantsInstanceResourcePolicy() {
 PutAccessGrantsInstanceResourcePolicyRequest putRequest =
 PutAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .policy(RESOURCE_POLICY)
 .build();
 PutAccessGrantsInstanceResourcePolicyResponse putResponse =
 s3Control.putAccessGrantsInstanceResourcePolicy(putRequest);
 LOGGER.info("PutAccessGrantsInstanceResourcePolicyResponse: " + putResponse);
 }
```

回應：

```
PutAccessGrantsInstanceResourcePolicyResponse(
 Policy={
 "Version": "2012-10-17",
 "Statement": [{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
 },
 "Action": [
 "s3:ListAccessGrants",
 "s3:ListAccessGrantsLocations",
 "s3:GetDataAccess",
 "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
 }]
 }
 )
```

Example 取得 S3 Access Grants 資源政策

若要取得 S3 存取授與資源政策，請使用下列程式碼範例。若要使用下列範例命令，請以您自己的資訊取代 *user input placeholders*。

```
public void getAccessGrantsInstanceResourcePolicy() {
```

```
 GetAccessGrantsInstanceResourcePolicyRequest getRequest =
 GetAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .build();
 GetAccessGrantsInstanceResourcePolicyResponse getResponse =
 s3Control.getAccessGrantsInstanceResourcePolicy(getRequest);
 LOGGER.info("GetAccessGrantsInstanceResourcePolicyResponse: " + getResponse);
 }
```

回應：

```
GetAccessGrantsInstanceResourcePolicyResponse(
 Policy={"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"AWS":"arn:aws:iam::444455556666:root"},"Action":
["s3:ListAccessGrants","s3:ListAccessGrantsLocations","s3:GetDataAccess"],"Resource":"arn:aw
east-2:111122223333:access-grants/default"}]},
 CreatedAt=2023-06-15T22:54:44.319Z
 )
```

Example 刪除 S3 Access Grants 資源政策

若要刪除 S3 存取授與資源政策，請使用下列程式碼範例。若要使用下列範例命令，請以您自己的資訊取代 *user input placeholders*。

```
public void deleteAccessGrantsInstanceResourcePolicy() {
 DeleteAccessGrantsInstanceResourcePolicyRequest deleteRequest =
 DeleteAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .build();
 DeleteAccessGrantsInstanceResourcePolicyResponse deleteResponse =
 s3Control.putAccessGrantsInstanceResourcePolicy(deleteRequest);
 LOGGER.info("DeleteAccessGrantsInstanceResourcePolicyResponse: " + deleteResponse);
 }
```

回應：

```
DeleteAccessGrantsInstanceResourcePolicyResponse()
```

步驟 3：在第二個帳戶中授予 IAM 身分，以呼叫帳戶中的 S3 存取授權執行個體

Amazon S3 資料的擁有者為帳戶中的 S3 Access Grants 執行個體設定跨帳戶政策後111122223333，第二個帳戶的擁有者444455556666必須為其 IAM 使用者或角色建立以身分識別為基礎的政策，而擁有者必須授與他們 S3 存取授與執行個體的存取權。在以身分為基礎的政策中，根據 S3 Access Grants 執行個體資源政策中授予的內容以及您要授予的許可，包括下列一或多個動作：

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

遵循AWS 跨帳戶存取模式，第二個帳戶中的 IAM 使用者或角色444455556666必須明確具有一或多個這些許可。例如，授予`s3:GetDataAccess`權限，讓 IAM 使用者或角色可以呼叫帳戶中的 S3 Access Grants 執行個體111122223333來請求登入資料。

若要使用此範例命令，請以您自己的資訊取代 *`user input placeholders`*。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
  "Effect": "Allow",
  "Action": [
   "s3:GetDataAccess",
  ],
   "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
  }
 ]
}
```

如需編輯 IAM 身分型政策的相關資訊，請參閱指AWS Identity and Access Management 南中的編輯 IAM 政策。

步驟 4：在帳戶的 S3 存取授權執行個體中建立授權，讓第二個帳戶中的 IAM 身分可存取部分 S3 資料

對於最後一個組態步驟，您可以在帳戶 111122223333 的 S3 存取授權執行個體中建立授權，以便將第二個帳戶 444455556666 中的 IAM 身分存取至您帳戶中的某些 S3 資料。您可以使用 Amazon S3 主控台、CLI、API 和開發套件來執行此操作。如需詳細資訊，請參閱 建立授權。

在授權中，從第二個帳戶指定 IAM 身分的 AWS ARN，並指定要授與存取權的 S3 資料 (儲存貯體、前綴或物件) 中的哪個位置。此位置必須已向 S3 存取授予執行個體註冊。如需詳細資訊，請參閱 註冊位置。您可以選擇性地指定子首碼。例如，如果您授與存取權的位置是值區，而您想要進一步限制存取該值區中特定物件的存取權，請在S3SubPrefix欄位中傳遞物件索引鍵名稱。或者，如果您想使用以特定前綴（例如）開頭的密鑰名稱來限制對值區中對象的訪問權限2024-03-research-results/，則傳遞S3SubPrefix=2024-03-research-results/。

以下是針對第二個帳戶中的身分建立存取權授與的範例 CLI 命令。如需詳細資訊，請參閱建立授權。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control create-access-grant \
--account-id 111122223333 \
--access-grants-location-id default \
--access-grants-location-configuration S3SubPrefix=prefixA* \
--permission READ \
--grantee GranteeType=IAM,GranteeIdentifier=arn:aws:iam::444455556666:role/data-consumer-1
```

設定跨帳戶存取後，第二個帳戶中的使用者或角色可以執行下列動作：

- 呼叫列ListAccessGrantsInstances出透過與其共用的 S3 存取授與執行個體 AWS RAM。如需詳細資訊，請參閱 檢視 S3 Access Grants 執行個體的詳細資訊。
- 向 S3 存取授權請求臨時登入資料。如需如何提出這些要求的詳細資訊，請參閱透過 S3 Access Grants 請求存取 Amazon S3 資料。

將標 AWS 籤與 S3 存取授權搭配使用

Amazon S3 Access Grants 中的標籤與 Amazon S3 中的物件標籤具有類似的特性。每個標籤都是金鑰值對。您可以在 S3 Access Grants 中標記的資源包括 S3 Access Grants 執行個體、位置和授權。

> **Note**
>
> S3 Access Grants 中的標記會使用與物件標記不同的 API 操作。S3 Access Grants 使用 TagResource、UntagResource 和 ListTagsForResource API 操作，其中資源可以是 S3 Access Grants 執行個體、註冊位置或存取授權。

與物件標籤類似的地方在於具有下列限制：

- 您可以在建立新的 S3 Access Grants 資源時，新增標籤至新資源，或是新增標籤至現有資源。
- 一個資源最多可與 10 個標籤相關聯。如果有多個標籤與同一資源相關聯，則這些標籤必須具有唯一的標籤索引鍵。
- 標籤金鑰最長可包含 128 個 Unicode 字元，標籤值最長可包含 256 個 Unicode 字元。標籤在內部是以 UTF-16 表示。在 UTF-16 中，字元會佔用 1 或 2 個字元位置。
- 索引鍵和值區分大小寫。

如需標籤限制的詳細資訊，請參閱《AWS Billing 使用者指南》中的使用者定義的標籤限制。

您可以使用 AWS Command Line Interface (AWS CLI)、Amazon S3 REST API 或 AWS 開發套件，在 S3 存取授權中標記資源。

使用 AWS CLI

若要安裝 AWS CLI，請參閱《AWS Command Line Interface 使用者指南》 AWS CLI中的〈安裝〉。

您可以在建立 S3 Access Grants 資源時或之後進行標記。下列範例顯示如何標記或取消標記 S3 Access Grants 執行個體。您可以對註冊位置和存取授權執行類似的操作。

若要使用下列範例命令，請將 *user input placeholders* 取代為您自己的資訊。

Example - 建立具有標籤的 S3 Access Grants 執行個體

```
aws s3control create-access-grants-instance \
 --account-id 111122223333 \
 --profile access-grants-profile \
 --region us-east-2 \
 --tags Key=tagKey1,Value=tagValue1
```

回應：

```
 {
    "CreatedAt": "2023-10-25T01:09:46.719000+00:00",
    "AccessGrantsInstanceId": "default",
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default"
}
```

Example - 標記已建立的 S3 Access Grants 執行個體

```
aws s3control tag-resource \
```

```
--account-id 111122223333 \
--resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
--profile access-grants-profile \
--region us-east-2 \
--tags Key=tagKey2,Value=tagValue2
```

Example - 列出 S3 Access Grants 執行個體的標籤

```
aws s3control list-tags-for-resource \
--account-id 111122223333 \
--resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
--profile access-grants-profile \
--region us-east-2
```

回應：

```
{
    "Tags": [
        {
            "Key": "tagKey1",
            "Value": "tagValue1"
        },
        {
            "Key": "tagKey2",
            "Value": "tagValue2"
        }
    ]
}
```

Example - 取消標記 S3 Access Grants 執行個體

```
aws s3control untag-resource \
 --account-id 111122223333 \
 --resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
 --profile access-grants-profile \
 --region us-east-2 \
 --tag-keys "tagKey2"
```

使用 REST API

您可以使用 Amazon S3 API 標記、取消標記 S3 Access Grants 執行個體、註冊位置或存取授權，以及列出其標籤。如需有關管理 S3 Access Grants 標籤的 REST API 支援資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列各節：

- TagResource
- UntagResource
- ListTagsForResource

S3 Access Grants 的限制

S3 Access Grants 具有下列限制：

> **Note**
>
> 如果您的使用案例超過這些限制，請聯絡 AWS 支援人員以要求更高的限制。

S3 Access Grants 執行個體

您可以為每 AWS 區域 個帳戶建立 1 個 S3 存取授與執行個體。請參閱建立 S3 Access Grants 執行個體。

S3 Access Grants 位置

每個 S3 Access Grants 執行個體可以註冊 1,000 個 S3 Access Grants 位置。請參閱註冊 S3 Access Grants 位置。

授權

每個 S3 Access Grants 執行個體可以建立 100,000 個授權。請參閱建立授權。

S3 Access Grants 整合

S3 存取授權可搭配下列 AWS 服務和功能使用。此頁面將隨著新的整合推出而更新。

AWS IAM Identity Center

跨應用程式的可信身分傳播

Amazon EMR

- 使用 S3 Access Grants 啟動 Amazon EMR 叢集

Amazon EMR on EKS

- 使用 S3 Access Grants 啟動 EKS 叢集上的 Amazon EMR

Amazon EMR Serverless 應用程式

- 使用 S3 Access Grants 啟動 Amazon EMR Serverless 應用程式

Amazon Athena

- 使用啟用 IAM Identity Center 的 Athena 工作群組

使用 ACL 管理存取

存取控制清單 (ACL) 是資源型選項之一，您可以用來管理值區和物件的存取權。您可以使用 ACL 將基本的讀取/寫入權限授與其他 AWS 帳戶人。使用 ACL 管理許可有其限制。

例如，您只能將權限授與其他人 AWS 帳戶；您無法將權限授與帳戶中的使用者。您無法授予條件式許可，也無法明確拒絕許可。ACL 適用於特定案例。例如，如果值區擁有者允許其他 AWS 帳戶 人上載物件，則只能使用擁有 AWS 帳戶 該物件的物件 ACL 來管理這些物件的權限。

S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來同時控制上傳至儲存貯體之物件的擁有權，以及停用或啟用 ACL。根據預設，物件擁有權設定為「儲存貯體擁有者強制執行」設定，而且所有 ACL 都會停用。停用 ACL 時，儲存貯體擁有者會擁有儲存貯體中的所有物件，並使用存取管理政策專門管理對這些物件的存取。

Amazon S3 中的大多數新式使用案例不再需要使用 ACL。建議您將 ACL 保持停用狀態，除非在異常情況下必須個別控制每個物件的存取。停用 ACL 後，您可以使用政策來控制對儲存貯體中所有物件的存取，無論是誰將物件上傳到您的儲存貯體。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

> **Important**
>
> 如果儲存貯體使用 S3 物件擁有權的儲存貯體擁有者強制執行設定，則您必須使用政策將存取權授予儲存貯體及其中的物件。在啟用儲存貯體擁有者強制執行設定的情況下，請求設定存取控制清單 (ACL) 或更新 ACL 失敗，並傳回 `AccessControlListNotSupported` 錯誤碼。仍支援讀取 ACL 的請求。

如需有關 ACL 的詳細資訊，請參閱以下主題：

主題

存取控制清單 (ACL) 概觀

Amazon S3 存取控制清單 (ACL) 可讓您管理對儲存貯體和物件的存取。每個儲存貯體與物件都會有一個與其連接的 ACL 作為子資源。它定義了哪些 AWS 帳戶 或組被授予訪問權限以及訪問的類型。收到針對資源的請求時，Amazon S3 會檢查對應的 ACL，以驗證請求者是否具有必要的存取許可。

S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來同時控制上傳至儲存貯體之物件的擁有權，以及停用或啟用 ACL。根據預設，物件擁有權設定為「儲存貯體擁有者強制執行」設定，而且所有 ACL 都會停用。停用 ACL 時，儲存貯體擁有者會擁有儲存貯體中的所有物件，並使用存取管理政策專門管理對這些物件的存取。

Amazon S3 中的大多數新式使用案例不再需要使用 ACL。建議您將 ACL 保持停用狀態，除非在異常情況下必須個別控制每個物件的存取。停用 ACL 後，您可以使用政策來控制對儲存貯體中所有物件的存取，無論是誰將物件上傳到您的儲存貯體。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

> **Important**
>
> 如果儲存貯體使用 S3 物件擁有權的儲存貯體擁有者強制執行設定，則您必須使用政策將存取權授予儲存貯體及其中的物件。在啟用儲存貯體擁有者強制執行設定的情況下，請求設定存取控制清單 (ACL) 或更新 ACL 失敗，並傳回 `AccessControlListNotSupported` 錯誤碼。仍支援讀取 ACL 的請求。

建立儲存貯體或物件時，Amazon S3 會建立預設 ACL 來授予資源擁有者完全控制資源。如下列範例儲存貯體 ACL 所示 (預設的物件 ACL 具有相同的結構)：

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
```

```
  <Owner>
    <ID>*** Owner-Canonical-User-ID ***</ID>
    <DisplayName>owner-display-name</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:type="Canonical User">
        <ID>*** Owner-Canonical-User-ID ***</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

此範本 ACL 包含 Owner 元素，可依 AWS 帳戶的正式使用者 ID 來識別擁有者。如需尋找您的正式使用者 ID 說明，請參閱 尋找 AWS 帳戶 規範的使用者 ID。Grant元素會識別受權者 (AWS 帳戶 或預先定義的群組) 以及授與的權限。此預設 ACL 包含擁有者的一個 Grant 元素。您可以新增 Grant 元素來授予許可，每個授予都會識別被授予者與許可。

> **Note**
>
> 一個 ACL 最多可以有 100 個授予。

主題

- 誰是被授予者？
- 我可以授予哪些許可？
- 常見 Amazon S3 請求的 aclRequired 值
- ACL 範例
- 固定的 ACL

誰是被授予者？

受權者可以是預先定義的 Amazon S3 群組 AWS 帳戶 或其中一個。您授予 AWS 帳戶 使用電子郵件地址或規範用戶 ID 的權限。不過，如果您在授予請求中提供電子郵件地址，Amazon S3 會尋找該帳戶的正式使用者 ID 並新增至 ACL。產生的 ACL 一律包含的標準使用者識別碼 AWS 帳戶，而不是的電子郵件地址。 AWS 帳戶

當您授與存取權利時，您可以將每個授與者指定為 *type*="*value*" 組，其中 *type* 為下列其中一項：

- id— 如果指定的值是 AWS 帳戶
- uri – 如果您將許可授與給預先定義的群組
- emailAddress – 如果指定的值是 AWS 帳戶的電子郵件地址

> **Important**
>
> 使用電子郵件地址指定僅在以下 AWS 區域中支援的被授與者：
>
> - 美國東部 (維吉尼亞北部)
> - 美國西部 (加利佛尼亞北部)
> - 美國西部 (奧勒岡)
> - 亞太區域 (新加坡)
> - 亞太區域 (雪梨)
> - 亞太區域 (東京)
> - 歐洲 (愛爾蘭)
> - 南美洲 (聖保羅)
>
> 如需 Amazon S3 支援的所有區域和端點的清單，請參閱《Amazon Web Services 一般參考》中的區域與端點。

Example 範例：電子郵件地址

例如，下列x-amz-grant-read標頭會授與「由電子郵件地址 AWS 帳戶 識別」讀取物件資料及其中繼資料的權限：

```
x-amz-grant-read: emailAddress="xyz@example.com", emailAddress="abc@example.com"
```

> **Warning**
>
> 當您授與資源的其他 AWS 帳戶 存取權時，請注意， AWS 帳戶 可以將其權限委派給其帳戶下的使用者。這稱為跨帳戶存取。如需有關使用跨帳戶存取的資訊，請參閱《IAM 使用者指南》中的建立角色將許可委派給 IAM 使用者。

尋找 AWS 帳戶 規範的使用者 ID

正式使用者 ID 與您的 AWS 帳戶相關聯。此 ID 是一長串字元，例如：

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

如需如何尋找帳戶的標準使用者 ID 的詳細資訊，請參閱《帳戶管理參考指南》 AWS 帳戶中的「尋找您的標準使用者 ID」AWS 。

您也可以讀取值區的 ACL 或 AWS 帳戶 具有存取 AWS 帳戶 權限的物件，來查詢的標準使用者 ID。當授與要求授予 AWS 帳戶 個人權限時，會使用帳戶的標準使用者 ID 將授權項目新增至 ACL。

> **Note**
>
> 如果您設定儲存貯體為公有 (不建議)，任何授權使用者皆可上傳物件至此儲存貯體中。這些匿名使用者沒有 AWS 帳戶。當任何匿名使用者上傳物件至您的儲存貯體時，Amazon S3 會新增特別的正式使用者 ID (65a011a29cdf8ec533ec3d1ccaae921c) 當做在 ACL 的物件擁有者。如需詳細資訊，請參閱 Amazon S3 儲存貯體和物件擁有權。

Amazon S3 預先定義的群組

Amazon S3 有一組預先定義的群組。將帳戶存取授予群組時，您會指定我們的其中一個 Amazon S3 URI，而不是正式使用者 ID。Amazon S3 提供下列預先定義的群組：

- 「已驗證的使用者」群組 – 以 `http://acs.amazonaws.com/groups/global/AuthenticatedUsers` 表示。

  這個群組代表全部 AWS 帳戶。此群組的存取權限允許任 AWS 帳戶 何人存取資源。但是，所有請求都必須經過簽署 (身分驗證)。

  > **Warning**
  >
  > 當您授予「已驗證的使用者」群組的存取權時，世界上任何 AWS 已驗證的使用者都可以存取您的資源。

- 「所有使用者」群組 – 以 `http://acs.amazonaws.com/groups/global/AllUsers` 表示。

  此群組的存取許可允許世界上任何人存取資源。此請求可以經過簽署 (通過身分驗證) 或未經過簽署 (匿名)。未簽署的要求會省略要求中的身分驗證標頭。

> **Warning**
>
> 強烈建議您絕不要將 WRITE、WRITE_ACP 或 FULL_CONTROL 許可授予所有使用者群組。例如，雖然 WRITE 許可不允許非擁有者覆寫或刪除現有物件，但 WRITE 許可仍會允許任何人在儲存貯體中儲存物件，而導致您必須支付費用。如需這些許可的詳細資訊，請參閱下節「我可以授予哪些許可？」。

- 「日誌交付」群組 – 以 http://acs.amazonaws.com/groups/s3/LogDelivery 表示。

  儲存貯體的 WRITE 許可允許此群組將伺服器存取日誌 (請參閱 使用伺服器存取記錄記錄要求) 寫入至儲存貯體。

> **Note**
>
> 使用 ACL 時，受權者可以是預先定義的 Amazon S3 群組 AWS 帳戶 或其中一個。不過，被授予者不可以是 IAM 使用者。如需 IAM 中 AWS 使用者與許可的詳細資訊，請參閱使用 AWS Identity and Access Management。

我可以授予哪些許可？

下表列出 Amazon S3 在 ACL 中支援的一組許可。物件 ACL 與儲存貯體 ACL 有相同的一組 ACL 許可。不過，視內容而定 (儲存貯體 ACL 或物件 ACL)，這些 ACL 許可會授予許可進行特定儲存貯體或物件操作。下表列出這些許可，並說明其在物件與儲存貯體內容中所代表的意義。

如需 Amazon S3 主控台中 ACL 許可的詳細資訊，請參閱 設定 ACL。

ACL 許可

許可	在儲存貯體上授予時	在物件上授予時
READ	允許被授予者列出儲存貯體中的物件	允許被授予者讀取物件資料及其中繼資料
WRITE	允許被授予者在儲存貯體中建立新物件。對於現有物件的儲存貯體和物件擁有者，還允許刪除和覆寫這些物件。	不適用
READ_ACP	允許被授予者讀取儲存貯體 ACL	允許被授予者讀取物件 ACL

許可	在儲存貯體上授予時	在物件上授予時
WRITE_ACP	允許被授予者寫入適用儲存貯體的 ACL	允許被授予者寫入適用物件的 ACL
FULL_CONTROL	允許承授者對儲存貯體的 READ、WRITE、READ_ACP 和 WRITE_ACP 許可	允許承授者對儲存貯體的 READ、READ_ACP 和 WRITE_ACP 許可

> **Warning**
>
> 授予 S3 儲存貯體與物件的存取許可時請小心。例如，將 WRITE 存取許可授予儲存貯體可允許被授予者在儲存貯體中建立物件。強烈建議您閱讀「存取控制清單 (ACL) 概觀」全節，再授予許可。

ACL 許可與存取政策許可的對應

如上表所示，相較於您可以在存取政策中設定的許可數目 (請參閱「Amazon S3 的政策動作」)，ACL 只允許一組有限的許可。其中每個許可都允許一或多個 Amazon S3 操作。

下表顯示每個 ACL 許可如何對應至對應的存取政策許可。如您所見，存取政策比 ACL 允許更多許可。您可以將 ACL 主要用來授予基本的讀取/寫入許可 (類似於檔案系統許可)。如需何時使用 ACL 的詳細資訊，請參閱 適用於 Amazon S3 的 Identity and Access Management。

如需 Amazon S3 主控台中 ACL 許可的詳細資訊，請參閱 設定 ACL。

ACL 許可	在儲存貯體上授予 ACL 許可時的對應存取政策許可	在物件上授予 ACL 許可時的對應存取政策許可
READ	s3:ListBucket 、s3:ListBucketVersions 與 s3:ListBucketMultipartUploads	s3:GetObject 和 s3:GetObjectVersion
WRITE	s3:PutObject	不適用

ACL 許可	在儲存貯體上授予 ACL 許可時的對應存取政策許可	在物件上授予 ACL 許可時的對應存取政策許可
	儲存貯體擁有者可以建立、覆寫和刪除儲存貯體中的任何物件，並且物件擁有者擁有其物件的 FULL_CONTROL 。 此外，當被授予者是儲存貯體擁有者時，授予儲存貯體 ACL 中的 WRITE 許可允許在該儲存貯體中的任何版本上執行 s3:DeleteObjectVersion 動作。	
READ_ACP	s3:GetBucketAcl	s3:GetObjectAcl 和 s3:GetObjectVersionAcl
WRITE_ACP	s3:PutBucketAcl	s3:PutObjectAcl 和 s3:PutObjectVersionAcl
FULL_CONTROL	相當於授予 READ、WRITE、READ_ACP 與 WRITE_ACP ACL 許可。因此，此 ACL 許可會對應至一組對應的存取政策許可。	相當於授予 READ、READ_ACP 與 WRITE_ACP ACL 許可。因此，此 ACL 許可會對應至一組對應的存取政策許可。

條件索引鍵

當您授與存取政策許可時，您可以使用條件索引鍵來限制物件上使用儲存貯體原則的 ACL 值。下面的內容金鑰與 ACL 對應。您可以使用這些內容金鑰來強制在請求中使用特定 ACL：

- s3:x-amz-grant-read - 需要讀取存取權。
- s3:x-amz-grant-write - 需要寫入存取權。
- s3:x-amz-grant-read-acp - 需要對儲存貯體 ACL 的讀取存取權。
- s3:x-amz-grant-write-acp - 需要對儲存貯體 ACL 的寫入存取權。
- s3:x-amz-grant-full-control - 需要完全控制權。
- s3:x-amz-acl - 需要固定的 ACL。

關於涉及 ACL 特定標頭的政策範例，請參閱 授予 s3：具有要求存儲桶所有者獲得完全控制PutObject權的條件的權限。如需 Amazon S3 特定條件金鑰的完整清單，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

常見 Amazon S3 請求的 **aclRequired** 值

若要識別需要 ACL 進行授權的 Amazon S3 請求，您可以使用 Amazon S3 伺服器存取日誌或 AWS CloudTrail中的 `aclRequired` 值。顯示在 CloudTrail 或 Amazon S3 伺服器存取日誌中的`aclRequired`值取決於呼叫的操作以及有關請求者、物件擁有者和儲存貯體擁有者的特定資訊。如果不需要 ACL，或者您要設定`bucket-owner-full-control`固定 ACL，或者儲存貯體政策允許這些請求，則 Amazon S3 伺服器存取日誌中的`aclRequired`值字串為 - ""，且在中不存在 CloudTrail。

下表列出各種 Amazon S3 API 操作的預期`aclRequired`值， CloudTrail 或 Amazon S3 伺服器存取日誌中的預期值。您可以使用此資訊來了解哪些 Amazon S3 操作依賴 ACL 進行授權。在下表中，A、B 和 C 代表與請求者、物件擁有者和儲存貯體擁有者相關聯的不同帳戶。帶星號 (*) 的項目表示任何帳戶 A、B 或 C。

> **Note**
>
> 除非另有指定，否則下表中的 `PutObject` 作業指示不設定 ACL 的請求，除非 ACL 是 `bucket-owner-full-control` ACL。的空值表`aclRequired`示記`aclRequired` AWS CloudTrail 錄檔中不存在。

aclRequired值 CloudTrail

操作名稱	要求者	物件擁有者	儲存貯體擁有者	儲存貯體政策授予存取權	**aclRequired** 值	原因
`GetObject`	A	A	A	是或否	null	相同帳戶的存取
	A	B	A	是或否	null	強制執行儲存貯體擁有者的相同帳戶存取

操作名稱	要求者	物件擁有者	儲存貯體擁有者	儲存貯體政策授予存取權	`aclRequired` 值	原因
	A	A	B	是	null	儲存貯體政策授予的跨帳戶存取權
	A	A	B	否	是	跨帳戶存取權依賴 ACL
	A	A	B	是	null	儲存貯體政策授予的跨帳戶存取權
	A	B	B	否	是	跨帳戶存取權依賴 ACL
	A	B	C	是	null	儲存貯體政策授予的跨帳戶存取權
	A	B	C	否	是	跨帳戶存取權依賴 ACL
`PutObject`	A	不適用	A	是或否	null	相同帳戶的存取
	A	不適用	B	是	null	儲存貯體政策授予的跨帳戶存取權
	A	不適用	B	否	是	跨帳户存取權依賴 ACL

操作名稱	要求者	物件擁有者	儲存貯體擁有者	儲存貯體政策授予存取權	**aclRequired** 值	原因
PutObject 具有 ACL (bucket-owner-full-control 除外)	*	不適用	*	是或否	是	請求授與 ACL
ListObjects	A	不適用	A	是或否	null	相同帳戶的存取
	A	不適用	B	是	null	儲存貯體政策授予的跨帳戶存取權
	A	不適用	B	否	是	跨帳戶存取權依賴 ACL
DeleteObject	A	不適用	A	是或否	null	相同帳戶的存取
	A	不適用	B	是	null	儲存貯體政策授予的跨帳戶存取權
	A	不適用	B	否	是	跨帳戶存取權依賴 ACL
PutObjectAcl	*	*	*	是或否	是	請求授與 ACL

操作名稱	要求者	物件擁有者	儲存貯體擁有者	儲存貯體政策授予存取權	**aclRequired** 值	原因
PutBucketAcl	*	不適用	*	是或否	是	請求授與 ACL

> **Note**
>
> 除非另有指定，否則下表中的 REST.PUT.OBJECT 操作指示不設定 ACL 的請求，除非 ACL 是 bucket-owner-full-control ACL。「-」字串的 aclRequired 值表示 Amazon S3 伺服器存取日誌中的空值。

Amazon S3 伺服器存取日誌的 **aclRequired** 值

操作名稱	要求者	物件擁有者	儲存貯體擁有者	儲存貯體政策授予存取權	**aclRequired** 值	原因
REST.GET.OBJECT	A	A	A	是或否	-	相同帳戶的存取
	A	B	A	是或否	-	強制執行儲存貯體擁有者的相同帳戶存取
	A	A	B	是	-	儲存貯體政策授予的跨帳戶存取權
	A	A	B	否	是	跨帳戶存取權依賴 ACL

操作名稱	要求者	物件擁有者	儲存貯體擁有者	儲存貯體政策授予存取權	`aclRequired` 值	原因
	A	B	B	是	-	儲存貯體政策授予的跨帳戶存取權
	A	B	B	否	是	跨帳戶存取權依賴 ACL
	A	B	C	是	-	儲存貯體政策授予的跨帳戶存取權
	A	B	C	否	是	跨帳戶存取權依賴 ACL
`REST.PUT.OBJECT`	A	不適用	A	是或否	-	相同帳戶的存取
	A	不適用	B	是	-	儲存貯體政策授予的跨帳戶存取權
	A	不適用	B	否	是	跨帳户存取權依賴 ACL

操作名稱	要求者	物件擁有者	儲存貯體擁有者	儲存貯體政策授予存取權	**aclRequired** 值	原因
REST.PUT.OBJECT 具有 ACL (bucket-owner-full-control 除外)	*	不適用	*	是或否	是	請求授與 ACL
REST.GET.BUCKET	A	不適用	A	是或否	-	相同帳戶的存取
	A	不適用	B	是	-	儲存貯體政策授予的跨帳戶存取權
	A	不適用	B	否	是	跨帳戶存取權依賴 ACL
REST.DELETE.OBJECT	A	不適用	A	是或否	-	相同帳戶的存取
	A	不適用	B	是	-	儲存貯體政策授予的跨帳戶存取權
	A	不適用	B	否	是	跨帳戶存取權依賴 ACL
REST.PUT.ACL	*	*	*	是或否	是	請求授與 ACL

ACL 範例

以下位於儲存貯體上的 ACL 範例會識別資源擁有者與一組授予。格式為 Amazon S3 REST API 中 ACL 的 XML 表示法。儲存貯體擁有者具備資源的 FULL_CONTROL。此外，ACL 還顯示如何將資源上的許可授與兩個 AWS 帳戶資源 (以標準使用者 ID 識別)，以及前一節討論的兩個預先定義的 Amazon S3 群組。

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>Owner-canonical-user-ID</ID>
    <DisplayName>display-name</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>Owner-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>

    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>user1-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>

    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>user2-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
```

```
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>

  </AccessControlList>
</AccessControlPolicy>
```

固定的 ACL

Amazon S3 支援一組預先定義的許可，稱為固定的 ACL。每個固定的 ACL 都有一組預先定義的被授與者和許可。下表列出一組固定的 ACL 及相關聯之預先定義的授予。

固定的 ACL	適用對象	新增至 ACL 的許可
`private`	儲存貯體與物件	擁有者取得 `FULL_CONTROL` 。其他人則沒有存取權利 (預設)。
`public-read`	儲存貯體與物件	擁有者取得 `FULL_CONTROL` 。`AllUsers` 群組 (請參閱「誰是被授予者？」) 取得 `READ` 存取。
`public-read-write`	儲存貯體與物件	擁有者取得 `FULL_CONTROL` 。`AllUsers` 群組取得 `READ` 與 `WRITE` 存取。通常不建議在儲存貯體上授予此存取許可。
`aws-exec-read`	儲存貯體與物件	擁有者取得 `FULL_CONTROL` 。Amazon EC2 取得 `READ` 存取，可從 Amazon S3 `GET` Amazon Machine Image (AMI) 套件。
`authenticated-read`	儲存貯體與物件	擁有者取得 `FULL_CONTROL` 。`AuthenticatedUsers` 群組取得 `READ` 存取許可。

固定的 ACL	適用對象	新增至 ACL 的許可
`bucket-owner-read`	物件	物件擁有者取得 `FULL_CONTROL` 。儲存貯體擁有者取得 `READ` 存取許可。如果您在建立儲存貯體時指定此固定的 ACL，Amazon S3 會予以忽略。
`bucket-owner-full-control`	物件	物件擁有者與儲存貯體擁有者都會取得物件的 `FULL_CONTROL` 。如果您在建立儲存貯體時指定此固定的 ACL，Amazon S3 會予以忽略。
`log-delivery-write`	儲存貯體	`LogDelivery` 群組取得儲存貯體的 `WRITE` 與 `READ_ACP` 許可。如需日誌的詳細資訊，請參閱「使用伺服器存取記錄記錄要求」。

> **Note**
>
> 您只能在要求中指定其中一個固定的 ACL。

您可以在請求中使用 `x-amz-acl` 請求標頭來指定固定 ACL。當 Amazon S3 收到含有固定 ACL 的請求時，就會將預先定義的授權新增至資源的 ACL。

設定 ACL

本節說明如何使用存取控制清單 (ACL) 來管理 S3 儲存貯體和物件的存取許可。您可以使用 AWS Management Console、 AWS Command Line Interface (CLI)、REST API 或 AWS SDK 將授權新增至資源 ACL。

儲存貯體許可與物件許可各自互相獨立。物件不會繼承其儲存貯體的許可。例如，若您建立儲存貯體，並將寫入存取授予使用者，除非使用者明確授予存取權給您，否則您無法存取該使用者的物件。

您可以將權限授與其他 AWS 帳戶 使用者或預先定義的群組。您要授予許可的使用者或群組稱為「被授予者」。根據預設，建立值區的擁有者擁有完整權限。 AWS 帳戶

您每授予使用者或群組一項許可，就會在與儲存貯體相關聯的 ACL 中新增一個項目。ACL 會列出授予，其中指出被授予者及獲授予之許可。

S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來同時控制上傳至儲存貯體之物件的擁有權，以及停用或啟用 ACL。根據預設，物件擁有權設定為「儲存貯體擁有者強制執行」設定，而

且所有 ACL 都會停用。停用 ACL 時，儲存貯體擁有者會擁有儲存貯體中的所有物件，並使用存取管理政策專門管理對這些物件的存取。

Amazon S3 中的大多數新式使用案例不再需要使用 ACL。建議您將 ACL 保持停用狀態，除非在異常情況下必須個別控制每個物件的存取。停用 ACL 後，您可以使用政策來控制對儲存貯體中所有物件的存取，無論是誰將物件上傳到您的儲存貯體。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

> **Important**
>
> 如果儲存貯體使用 S3 物件擁有權的儲存貯體擁有者強制執行設定，則您必須使用政策將存取權授予儲存貯體及其中的物件。在啟用儲存貯體擁有者強制執行設定的情況下，請求設定存取控制清單 (ACL) 或更新 ACL 失敗，並傳回 `AccessControlListNotSupported` 錯誤碼。仍支援讀取 ACL 的請求。

> **Warning**
>
> 我們強烈建議您避免授與寫入權限給 Everyone (公開存取) 或「已驗證的使用者」群組 (所有 AWS 已驗證的使用者) 群組。如需詳細資訊了解授予寫入存取權限對這些群組的影響，請參閱「Amazon S3 預先定義的群組」。

使用 S3 主控台來設定儲存貯體的 ACL 許可

主控台會顯示重複被授予者的合併存取授予。若要查看 ACL 的完整清單，請使用 Amazon S3 REST API 或 AWS 開發套件。 AWS CLI

下表顯示您可以在 Amazon S3 主控台中為儲存貯體設定的 ACL 許可。

儲存貯體的 Amazon S3 主控台 ACL 許可

主控台許可	ACL 許可	Access (存取)
Objects (物件) – List (列出)	`READ`	允許被授予者列出儲存貯體中的物件。

主控台許可	ACL 許可	Access (存取)
Objects (物件) - Write (寫入)	`WRITE`	允許被授予者在儲存貯體中建立新物件。對於現有物件的儲存貯體和物件擁有者，還允許刪除和覆寫這些物件。
Bucket ACL (儲存貯體 ACL) - Read (讀取)	`READ_ACP`	允許被授予者讀取儲存貯體 ACL。
Bucket ACL (儲存貯體 ACL) - Write (寫入)	`WRITE_ACP`	允許被授予者寫入適用儲存貯體的 ACL。
每個人（公開存取）：物件 - 列出	`READ`	對儲存貯體中的物件授予公開讀取存取權限。當您將列出存取權限授予每個人（公開存取權限）時，世界上的任何人都可以存取儲存貯體中的物件。
每個人 (公開存取權限)：儲存貯體 ACL - 讀取	`READ_ACP`	對儲存貯體 ACL 授予公開讀取存取權限。當您將讀取存取權限授予每個人（公開存取權限）時，世界上的任何人都可以存取儲存貯體 ACL。

如需 ACL 許可的詳細資訊，請參閱存取控制清單 (ACL) 概觀。

Important

如果儲存貯體使用 S3 物件擁有權的儲存貯體擁有者強制執行設定，則您必須使用政策將存取權授予儲存貯體及其中的物件。在啟用儲存貯體擁有者強制執行設定的情況下，請求設定存取控制清單 (ACL) 或更新 ACL 失敗，並傳回 `AccessControlListNotSupported` 錯誤碼。仍支援讀取 ACL 的請求。

設定儲存貯體的 ACL 許可

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇要設定許可的儲存貯體名稱。
3. 選擇 Permissions (許可)。
4. 在 Access control list (存取控制清單) 下，選擇 Edit (編輯)。

   您可以編輯儲存貯體的下列 ACL 許可：

   物件

   - List (清單) – 允許承授者列出儲存貯體中的物件。
   - Write (寫入) – 允許承授者在儲存貯體中建立新物件。對於現有物件的儲存貯體和物件擁有者，還允許刪除和覆寫這些物件。

     在 S3 主控台中，您只能授與寫入存取權給 S3 日誌傳遞群組和儲存貯體擁有者 (您的 AWS 帳戶)。強烈建議您不要授予其他承授者的寫入存取權。不過，如果您需要授與寫入存取權，您可以使用 AWS CLI、 AWS SDK 或 REST API。

   儲存貯體 ACL

   - Read (讀取) – 允許承授者讀取儲存貯體 ACL。
   - Write (寫入) – 允許承授者寫入適用儲存貯體的 ACL。
5. 若要變更值區擁有者的權限，請在值區擁有者 (您的 AWS 帳戶) 旁清除或選取下列 ACL 權限：

   - Objects (物件) – List (列出) 或 Write (寫入)
   - Bucket ACL (儲存貯體) – Read (讀取) 或 Write (寫入)

   擁有者指的是 IAM 使用者 AWS 帳戶根使用者，而不是 AWS Identity and Access Management IAM 使用者。如需根使用者的詳細資訊，請參閱《IAM 使用者指南》中的 AWS 帳戶根使用者。
6. 若要授予或復原一般公眾 (網際網路上的所有人) 的許可，請在 Everyone (public access) (所有人 (公開存取)) 旁邊，清除或從下列 ACL 許可中選取：

   - Objects (物件) – List (列出)
   - Bucket ACL (儲存貯體) – Read (讀取)

> **Warning**
>
> 將 S3 儲存貯體的公開存取授予 Everyone (每個人) 群組時請小心。當您將存取授予此群組時，全世界的所有人都能存取您的儲存貯體。強烈建議您絕不要授予任何種類的 S3 儲存貯體公用寫入存取。

7. 若要授與或復原任何擁有「已驗證使用者」群組 (擁有 AWS 帳戶) 旁的使用者的權限，請清除或從下列 ACL 權限中選取： AWS 帳戶

   - Objects (物件) – List (列出)
   - Bucket ACL (儲存貯體) – Read (讀取)

8. 若要授予或復原 Amazon S3 將伺服器存取日誌寫入儲存貯體的許可，請在 S3 log delivery group (S3 日誌交付群組) 下，清除或從下列 ACL 許可中選取：

   - Objects (物件) – List (列出) 或 Write (寫入)
   - Bucket ACL (儲存貯體) – Read (讀取) 或 Write (寫入)

   如果儲存貯體設定為要接收存取日誌的目標儲存貯體，儲存貯體許可必須將儲存貯體的寫入存取授予 Log Delivery (日誌交付) 群組。當您啟用儲存貯體上的伺服器存取記錄日誌時，Amazon S3 主控台會將寫入存取權限授予您選擇接收日誌之目標儲存貯體的 Log Delivery (日誌交付) 群組。如需伺服器存取記錄日誌的詳細資訊，請參閱「啟用 Amazon S3 伺服器存取記錄日誌」。

9. 若要授與存取權給另一個人 AWS 帳戶，請執行下列動作：

   a. 選擇 Add grantee (新增承授者)。

   b. 在 Grantee (被授予者) 方塊中，輸入其他 AWS 帳戶的正式 ID。

   c. 從下列 ACL 許可中選取：

      - Objects (物件) – List (列出) 或 Write (寫入)
      - Bucket ACL (儲存貯體) – Read (讀取) 或 Write (寫入)

> **Warning**
>
> 當您授與資源的其他 AWS 帳戶 存取權時，請注意， AWS 帳戶 可以將其權限委派給其帳戶下的使用者。這稱為跨帳戶存取。如需有關使用跨帳戶存取的資訊，請參閱《IAM 使用者指南》中的建立角色將許可委派給 IAM 使用者。

10. 若要移除對其他人的存取權 AWS 帳戶，請在 [其他 AWS 帳戶存取權] 下選擇 [移除]
11. 若要儲存您所做的變更，請選擇 Save changes (儲存變更)。

使用 S3 主控台來設定物件的 ACL 許可

主控台會顯示重複被授予者的合併存取授予。若要查看 ACL 的完整清單，請使用 Amazon S3 REST API 或 AWS 開發套件。 AWS CLI下表顯示您可以在 Amazon S3 主控台中為物件設定的 ACL 許可。

物件的 Amazon S3 主控台 ACL 許可

主控台許可	ACL 許可	Access (存取)
物件 - 讀取	`READ`	允許被授予者讀取物件資料及其中繼資料。
物件 ACL - 讀取	`READ_ACP`	允許被授予者讀取物件 ACL。
物件 ACL - 寫入	`WRITE_ACP`	允許被授予者寫入適用物件的 ACL

如需 ACL 許可的詳細資訊，請參閱存取控制清單 (ACL) 概觀。

> **Important**
>
> 如果儲存貯體使用 S3 物件擁有權的儲存貯體擁有者強制執行設定，則您必須使用政策將存取權授予儲存貯體及其中的物件。在啟用儲存貯體擁有者強制執行設定的情況下，請求設定存取控制清單 (ACL) 或更新 ACL 失敗，並傳回 `AccessControlListNotSupported` 錯誤碼。仍支援讀取 ACL 的請求。

設定物件的 ACL 許可

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇包含該物件的儲存貯體名稱。
3. 在 objects (物件) 清單中，選擇您要設定許可的物件名稱。
4. 選擇 Permissions (許可)。
5. 在 Access control list (ACL) (存取控制清單 (ACL)) 下，選擇 Edit (編輯)。

您可以編輯物件的下列 ACL 許可：

物件

- Read (讀取) – 允許承授者讀取物件資料及其中繼資料

物件 ACL

- Read (讀取) – 允許承授者讀取物件 ACL。
- Write (寫入) – 允許承授者寫入適用物件的 ACL。在 S3 主控台中，您只能授與儲存貯體擁有者 (您的 AWS 帳戶) 的寫入存取權。強烈建議您不要授予其他承授者的寫入存取權。不過，如果您需要授與寫入存取權，您可以使用 AWS CLI、 AWS SDK 或 REST API。

6. 您可以管理下列項目的物件存取許可：

   a. 其他擁有者的存取

      擁有者指的是 AWS 帳戶根使用者，而不是 AWS Identity and Access Management IAM 使用者。如需根使用者的詳細資訊，請參閱《IAM 使用者指南》中的 AWS 帳戶根使用者。

      若要變更擁有者的物件存取權限，請在物件擁有者的存取權下，選擇您的 AWS 帳戶 (擁有者)。

      選取您要變更之許可的核取方塊，然後選擇 Save (儲存)。

   b. 訪問其他 AWS 帳戶

      若要授與其他 AWS 使用者的權限 AWS 帳戶，請在 [其他使用者的存取權] 底下 AWS 帳戶，選擇 [新增帳戶]。在「輸入 ID」欄位中，輸入您要授與物件權限之 AWS 使用者的標準 ID。如需有關尋找規範 ID 的詳細資訊，請參閱中的 AWS 帳戶. Amazon Web Services 一般參考您最多可以新增 99 個使用者。

      選取您要授予使用者之許可的核取方塊，然後選擇 Save (儲存)。若要顯示許可的相關資訊，請選擇說明圖示。

   c. 公用存取

      若要將物件的存取權授予一般大眾 (全世界的所有人)，請在 Public access (公開存取) 下，選擇 Everyone (每個人)。授予公用存取許可表示全世界的所有人都能存取該物件。

      選取您要授予之許可的核取方塊，然後選擇 Save (儲存)。

> **Warning**
>
> - 將 Amazon S3 物件的匿名存取權授予每個人群組時請謹慎小心。當您將存取權授予此群組時，全世界任何人都能存取您的物件。若必須將存取權授予每個人，強烈建議您只授予 Read objects (讀取物件) 許可。
> - 強烈建議您「不要」將寫入物件許可授予 Everyone (每個人) 群組。否則所有人都可覆寫物件的 ACL 許可。

使用 AWS 軟體開發套件

此章節提供範例說明如何設定存取控制清單 (ACL) 授與給儲存貯體和物件。

> **Important**
>
> 如果儲存貯體使用 S3 物件擁有權的儲存貯體擁有者強制執行設定，則您必須使用政策將存取權授予儲存貯體及其中的物件。在啟用儲存貯體擁有者強制執行設定的情況下，請求設定存取控制清單 (ACL) 或更新 ACL 失敗，並傳回 `AccessControlListNotSupported` 錯誤碼。仍支援讀取 ACL 的請求。

Java

此章節提供範例說明如何設定存取控制清單 (ACL) 授與給儲存貯體和物件。第一個範例為以定義的 ACL (請參閱 固定的 ACL) 建立儲存貯體後，建立自訂許可授與名單，然後用含有自訂授與 ACL 取代定義的 ACL。第二個範例說明如何修改使用 `AccessControlList.grantPermission()` 法修改 ACL。

Example 建立儲存貯體並指定授予 S3 日誌交付群組許可的固定 ACL

此為建立儲存貯體的範例。在此要求中，該範例指定了定義的 ACL 授與日誌傳遞群組許可，以寫入儲存貯體日誌。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

```
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.ArrayList;

public class CreateBucketWithACL {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String userEmailForReadPermission = "*** user@example.com ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .build();

            // Create a bucket with a canned ACL. This ACL will be replaced by the
            // setBucketAcl()
            // calls below. It is included here for demonstration purposes.
            CreateBucketRequest createBucketRequest = new
 CreateBucketRequest(bucketName, clientRegion.getName())
                    .withCannedAcl(CannedAccessControlList.LogDeliveryWrite);
            s3Client.createBucket(createBucketRequest);

            // Create a collection of grants to add to the bucket.
            ArrayList<Grant> grantCollection = new ArrayList<Grant>();

            // Grant the account owner full control.
            Grant grant1 = new Grant(new
 CanonicalGrantee(s3Client.getS3AccountOwner().getId()),
                    Permission.FullControl);
            grantCollection.add(grant1);

            // Grant the LogDelivery group permission to write to the bucket.
            Grant grant2 = new Grant(GroupGrantee.LogDelivery, Permission.Write);
            grantCollection.add(grant2);

            // Save grants by replacing all current ACL grants with the two we just
 created.
            AccessControlList bucketAcl = new AccessControlList();
            bucketAcl.grantAllPermissions(grantCollection.toArray(new Grant[0]));
            s3Client.setBucketAcl(bucketName, bucketAcl);
```

```
            // Retrieve the bucket's ACL, add another grant, and then save the new
 ACL.
            AccessControlList newBucketAcl = s3Client.getBucketAcl(bucketName);
            Grant grant3 = new Grant(new
 EmailAddressGrantee(userEmailForReadPermission), Permission.Read);
            newBucketAcl.grantAllPermissions(grant3);
            s3Client.setBucketAcl(bucketName, newBucketAcl);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example 更新現有物件的 ACL

此範例示範更新物件的 ACL。範例會執行下列任務：

- 擷取物件的 ACL
- 藉由移除所有現存許可來清除 ACL
- 新增兩項許可：擁有者的完全存取許可和 WRITE_ACP (請參閱[我可以授予哪些許可？](#)) 利用電子郵件地址辨別使用者。
- 儲存 ACL 至物件中

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.CanonicalGrantee;
import com.amazonaws.services.s3.model.EmailAddressGrantee;
import com.amazonaws.services.s3.model.Permission;
```

```
import java.io.IOException;

public class ModifyACLExistingObject {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String emailGrantee = "*** user@example.com ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Get the existing object ACL that we want to modify.
            AccessControlList acl = s3Client.getObjectAcl(bucketName, keyName);

            // Clear the existing list of grants.
            acl.getGrantsAsList().clear();

            // Grant a sample set of permissions, using the existing ACL owner for
 Full
            // Control permissions.
            acl.grantPermission(new CanonicalGrantee(acl.getOwner().getId()),
 Permission.FullControl);
            acl.grantPermission(new EmailAddressGrantee(emailGrantee),
 Permission.WriteAcp);

            // Save the modified ACL back to the object.
            s3Client.setObjectAcl(bucketName, keyName, acl);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

Example 建立儲存貯體並指定授予 S3 日誌交付群組許可的固定 ACL

此為建立儲存貯體的 C# 範例。在此要求中，該程式碼也指定了定義的 ACL 授與日誌傳遞群組許可，以便在儲存貯體寫入日誌。

如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ManagingBucketACLTest
    {
        private const string newBucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            CreateBucketUseCannedACLAsync().Wait();
        }

        private static async Task CreateBucketUseCannedACLAsync()
        {
            try
            {
                // Add bucket (specify canned ACL).
                PutBucketRequest putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = S3Region.EUW1, // S3Region.US,
                                                  // Add canned ACL.
```

```
                    CannedACL = S3CannedACL.LogDeliveryWrite
                };
                PutBucketResponse putBucketResponse = await
 client.PutBucketAsync(putBucketRequest);

                // Retrieve bucket ACL.
                GetACLResponse getACLResponse = await client.GetACLAsync(new
 GetACLRequest
                {
                    BucketName = newBucketName
                });
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("S3 error occurred. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

Example 更新現有物件的 ACL

此 C# 範例示範更新現有物件的 ACL。範例會執行下列任務：

- 擷取物件的 ACL。
- 藉由移除所有現存許可來清除 ACL。
- 新增兩項許可：擁有者的完全存取許可和 WRITE_ACP 利用電子郵件地址辨別使用者。
- 藉傳送 PutAcl 請求儲存 ACL。

如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
```

```
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ManagingObjectACLTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** object key name ***";
        private const string emailAddress = "*** email address ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            TestObjectACLTestAsync().Wait();
        }
        private static async Task TestObjectACLTestAsync()
        {
            try
            {
                    // Retrieve the ACL for the object.
                    GetACLResponse aclResponse = await client.GetACLAsync(new
 GetACLRequest
                    {
                        BucketName = bucketName,
                        Key = keyName
                    });

                    S3AccessControlList acl = aclResponse.AccessControlList;

                    // Retrieve the owner (we use this to re-add permissions after
 we clear the ACL).
                    Owner owner = acl.Owner;

                    // Clear existing grants.
                    acl.Grants.Clear();

                    // Add a grant to reset the owner's full permission (the
 previous clear statement removed all permissions).
                    S3Grant fullControlGrant = new S3Grant
                    {
```

```
                        Grantee = new S3Grantee { CanonicalUser = owner.Id },
                        Permission = S3Permission.FULL_CONTROL

                    };

                    // Describe the grant for the permission using an email address.
                    S3Grant grantUsingEmail = new S3Grant
                    {
                        Grantee = new S3Grantee { EmailAddress = emailAddress },
                        Permission = S3Permission.WRITE_ACP
                    };
                    acl.Grants.AddRange(new List<S3Grant> { fullControlGrant,
 grantUsingEmail });

                    // Set a new ACL.
                    PutACLResponse response = await client.PutACLAsync(new
 PutACLRequest
                    {
                        BucketName = bucketName,
                        Key = keyName,
                        AccessControlList = acl
                    });
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

使用 REST API

Amazon S3 API 可讓您在建立儲存貯體或物件時設定 ACL。Amazon S3 也提供在現有儲存貯體或物件上設定 ACL 的 API。這些 API 提供下列方法來設定 ACL：

- 使用請求標頭設定 ACL – 當您傳送請求以建立資源 (儲存貯體或物件) 時，您可以使用請求標頭來設定 ACL。您可以使用這些標頭來指定固定的 ACL，或明確指定授予 (明確識別被授予者與許可)。
- 使用請求內文設定 ACL – 當您傳送請求以在現有的資源上設定 ACL 時，您可以在請求標頭或內文中設定 ACL。

如需有關 REST API 支援管理 ACL 的資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列章節：

- GET 儲存貯體 acl
- PUT 儲存貯體 acl
- GET 物件 acl
- PUT 物件 acl
- PUT 物件
- PUT 儲存貯體
- PUT 物件 - 複製
- 啟動分段上傳

> **Important**
>
> 如果儲存貯體使用 S3 物件擁有權的儲存貯體擁有者強制執行設定，則您必須使用政策將存取權授予儲存貯體及其中的物件。在啟用儲存貯體擁有者強制執行設定的情況下，請求設定存取控制清單 (ACL) 或更新 ACL 失敗，並傳回 `AccessControlListNotSupported` 錯誤碼。仍支援讀取 ACL 的請求。

存取控制清單 (ACL)-特定請求標頭

您可以使用標頭，授予以存取控制清單 (ACL) 為基礎的許可。所有物件預設皆為私有。只有擁有者有完整的存取控制權。新增物件時，您可以將許可授與個別 AWS 帳戶 或 Amazon S3 定義的預先定義群組。然後，這些許可會新增至物件上的存取控制清單 (ACL)。如需詳細資訊，請參閱 存取控制清單 (ACL) 概觀。

透過此操作，您可以使用下列兩種方法之一授與存取許可：

- 固定的 ACL (**`x-amz-acl`**) — Amazon S3 支援一組預先定義的 ACL，稱為固定的 ACL。每個固定的 ACL 都有一組預先定義的被授與者和許可。如需詳細資訊，請參閱 固定的 ACL。

- 存取權限 — 若要明確授與特定 AWS 帳戶 或群組的存取權限，請使用下列標頭。每個標頭映射到 Amazon S3 在 ACL 中支援的特定許可。如需詳細資訊，請參閱 存取控制清單 (ACL) 概觀。在標頭中，您指定取得特定許可的授與者清單。
  - x-amz-grant-read
  - x-amz-grant-write
  - x-amz-grant-read-ACP
  - x-amz-grant-write-ACP
  - x-amz-grant-full-控制

使用 AWS CLI

若要取得有關使用管理 ACL 的更多資訊 AWS CLI，請參閱《AWS CLI 指令參考》put-bucket-acl中的〈〉。

> **Important**
>
> 如果儲存貯體使用 S3 物件擁有權的儲存貯體擁有者強制執行設定，則您必須使用政策將存取權授予儲存貯體及其中的物件。在啟用儲存貯體擁有者強制執行設定的情況下，請求設定存取控制清單 (ACL) 或更新 ACL 失敗，並傳回 `AccessControlListNotSupported` 錯誤碼。仍支援讀取 ACL 的請求。

ACL 的政策範例

您可以使用儲存貯體政策中的條件金鑰來控制對 Amazon S3 的存取。

主題

- 授予 s3：具有要求存儲桶所有者獲得完全控制PutObject 權的條件的權限
- 授予 s3：具有 x-amz-acl 標題條件的PutObject 權限

授予 s3：具有要求存儲桶所有者獲得完全控制PutObject 權的條件的權限

PUT 物件操作允許存取控制清單 (ACL) 特定的標頭，可用來授予以 ACL 為基礎的許可。使用這些金鑰，儲存貯體擁有者可設定條件，在使用者上傳物件時要求特定的存取許可。

假設帳戶 A 擁有一個儲存貯體，而帳戶管理員希望將上傳物件的許可授予帳戶 B 的使用者 Dave。根據預設，Dave 上傳的物件是為帳戶 B 所擁有，而帳戶 A 沒有這些物件的許可。因為付費的是儲存貯

體擁有者，它希望有 Dave 上傳之物件的完整許可。帳戶 A 管理員只要將 s3:PutObject 許可授予 Dave，附帶要求包含 ACL 專屬標頭的條件，明確授予完整許可或指定使用固定的 ACL，即可完成此操作。如需詳細資訊，請參閱 PUT 物件。

需要標 x-amz-full-control 頭

您可以指定要求中的 x-amz-full-control 標頭需要有儲存貯體擁有者的完全控制許可。下列儲存貯體政策授予使用者 Dave s3:PutObject 許可，附使用 s3:x-amz-grant-full-control 條件索引鍵的條件，需要要求包含 x-amz-full-control 標頭。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountB-ID:user/Dave"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::awsexamplebucket1/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
        }
      }
    }
  ]
}
```

> **Note**
>
> 此範例有關跨帳戶許可。不過，如果 Dave (取得權限的人) 屬於擁 AWS 帳戶 有值區的人，則不需要此條件權限。這是因為 Dave 隸屬的父帳戶擁有使用者上傳的物件。

新增明確拒絕

前面的儲存貯體政策將條件式許可授予帳戶 B 的使用者 Dave。當此政策生效時，Dave 卻可能透過另一個政策無條件取得相同的許可。例如，Dave 可屬於某個群組，而您無條件授予該群組 s3:PutObject 許可。為避免此等許可漏洞，您可以新增明確拒絕，撰寫較嚴格的存取政策。在本例

中，如果 Dave 在授予儲存貯體擁有者的完整許可要求中未包含必要標頭，您會明確拒絕他的使用者上傳許可。明確拒絕會取代任何其他已授予的許可。以下是已新增明確拒絕的修訂後存取政策範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        },
        {
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        }
    ]
}
```

使用以下項目測試原則 AWS CLI

如果您有兩個 AWS 帳戶，則可以使用 AWS Command Line Interface (AWS CLI) 來測試原則。您可以附加原則，並使用 Dave 的認證，使用下列 AWS CLI `put-object`命令來測試權限。您可以新增 `--profile` 參數，來提供 Dave 的憑證。您可以透過新增 `--grant-full-control` 參數，授予儲存貯

體擁有者的完全控制許可。若要取得有關設定和使用的更多資訊 AWS CLI，請參閱使用 AWS CLI 來透過 Amazon S3 進行開發。

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg
 --grant-full-control id="AccountA-CanonicalUserID" --profile AccountBUserProfile
```

需要標 x-amz-acl 頭

您可以要求有固定 ACL 的 x-amz-acl 標頭將完全控制許可授予儲存貯體擁有者。要求中若需要 x-amz-acl 標頭，您可以取代 Condition 區塊中的金鑰/值對，並指定 s3:x-amz-acl 條件索引鍵，如以下範例所示。

```
"Condition": {
    "StringEquals": {
        "s3:x-amz-acl": "bucket-owner-full-control"
    }
}
```

若要使用測試權限 AWS CLI，請指定--acl參數。 AWS CLI 然後在發送請求時添加x-amz-acl標題。

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg
 --acl "bucket-owner-full-control" --profile AccountBadmin
```

授予 s3：具有 x-amz-acl 標題條件的PutObject 權限

AWS 帳戶 如果請求包含使物件可公開讀取的x-amz-acl標頭，則下列儲存貯體政策會授與兩s3:PutObject者的權限。Condition 區塊使用 StringEquals 條件，而且有用於評估的金鑰/值對 "s3:x-amz-acl":["public-read"]。在此金鑰/值對中，s3:x-amz-acl 是 Amazon S3 專用金鑰，如字首 s3: 所示。

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
         "Sid":"AddCannedAcl",
            "Effect":"Allow",
         "Principal": {
                "AWS": [
                "arn:aws:iam::Account1-ID:root",
    "arn:aws:iam::Account2-ID:root"
```

```
    ]
            },
         "Action":"s3:PutObject",
            "Resource": ["arn:aws:s3:::awsexamplebucket1/*"],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl":["public-read"]
                }
            }
        }
    ]
}
```

> **Important**
>
> 不是所有的條件都對所有動作有意義。例如，在授予 `s3:CreateBucket` Amazon S3 許可的政策中包含 `s3:LocationConstraint` 條件有意義。但在授予 `s3:GetObject` 許可的政策中包含此條件就沒意義。Amazon S3 可以測試這類涉及 Amazon S3 特定條件的語意錯誤。但是，如果您為 IAM 使用者或角色建立政策，但包含語意無效的 Amazon S3 條件，則不會回報任何錯誤，因為 IAM 無法驗證 Amazon S3 條件。

封鎖對 Amazon S3 儲存體的公開存取權

Amazon S3 封鎖公開存取功能可提供存取點、儲存貯體和帳戶的設定，以協助您管理對 Amazon S3 資源的公開存取。依預設，新的儲存貯體、存取點和物件不允許公開存取。不過，使用者可以修改儲存貯體政策、存取點政策或物件許可，以允許公開存取。S3 封鎖公開存取設定能覆寫這些政策和許可的設定，讓您可以限制這些資源的公開存取。

使用 S3 封鎖公開存取，帳戶管理員和儲存貯體擁有者即可輕鬆設定集中式控制項來限制 Amazon S3 的公開存取權限，無論資源的建立方式為何都會強制執行。

如需設定公有區塊存取的指示，請參閱 設定封鎖公開存取。

Amazon S3 在收到請求存取儲存貯體或物件時，將會判斷儲存貯體或儲存貯體擁有者的帳戶是否套用封鎖公開存取設定。如果請求是透過存取點提出，Amazon S3 也會檢查存取點的封鎖公開存取設定。若有禁止所請求存取的現有封鎖公開存取設定，Amazon S3 便會拒絕該請求。

Amazon S3 封鎖公開存取提供四個設定。這些是獨立的設定，且可以任意組合使用。每個設定都可套用至一個存取點、一個儲存貯體或整個 AWS 帳戶。如果存取點、儲存貯體或帳戶的封鎖公開存取設定不同，則 Amazon S3 會套用存取點、儲存貯體和帳戶設定的最嚴格組合。

Amazon S3 在評估封鎖公開存取設定是否禁止操作時，將會拒絕任何違反存取點、儲存貯體或帳戶設定的請求。

Important

透過存取控制清單 (ACL)、存取點政策、儲存貯體政策或全部來授與對儲存貯體和物件的公開存取許可。為協助確保封鎖所有 Amazon S3 存取點、儲存貯體和物件的公開存取，我們建議為帳戶開啟封鎖公開存取的所有四項設定。這些設定會封鎖所有現有和未來儲存貯體和存取點的公開存取。

套用這些設定前，請確認應用程式無須公用存取權限也可正常運作。如果儲存貯體或物件需要特定層級的公開存取權限 (例如使用 Amazon S3 託管靜態網站中所述的託管靜態網站)，則您可根據特定儲存使用案例自訂以下各設定。

啟用區塊公開存取可防止透過直接附加至 S3 資源的資源政策或存取控制清單 (ACL) 授與公開存取，以協助保護您的資源。除了啟用「封鎖公用存取」之外，請仔細檢查下列原則，以確認其未授予公用存取權：

- 附加至關聯 AWS 主體 (例如 IAM 角色) 的身分識別型政策
- 附加至關聯 AWS 資源 (例如 AWS Key Management Service (KMS) 金鑰) 的以資源為基礎的政策

Note

- 您只能針對存取點、儲存貯體和 AWS 帳戶啟用封鎖公開存取設定。Amazon S3 不支援個別物件的封鎖公開存取設定。
- 當您將封鎖公開存取設定套用至帳戶時，這些設定會套用至 AWS 區域 全域的所有設定。設定可能不會在區域內立即或同時生效，但它們最後會散佈到所有區域。

主題

- 設定封鎖公開存取
- 為您的帳戶設定封鎖公開存取
- 為您的 S3 儲存貯體設定封鎖公開存取

封鎖公開存取設定

S3 封鎖公開存取提供四種設定。您可以任意組合使用這些設定，並套用到個別存取點、儲存貯體或整個 AWS 帳戶。若您將設定套用到帳戶，它會套用到該帳戶擁有的所有儲存貯體和存取點。同樣地，如果您將設定套用至儲存貯體，它會套用至與該儲存貯體相關聯的所有存取點。

下表包含可用的設定。

名稱	描述
`BlockPublicAcls`	將此選項設為 TRUE 會導致以下行為： • 若指定的存取控制清單 (ACL) 為公有，則 PUT 儲存貯體 acl 和 PUT 物件 acl 呼叫會失敗。 • 若請求包含公有 ACL，則 PUT 物件呼叫會失敗。 • 若此設定套用到帳戶，則當請求包含公有 ACL 時，PUT 儲存貯體呼叫便會失敗。 當此設定設定為時TRUE，指定的作業會失敗 (無論是透過 REST API 或 AWS SDK 進行)。 AWS CLI但是，儲存貯體和物件現有的政策和 ACL 則不會修改。此設定可讓您防止公開存取，同時允許您稽核、縮小搜尋範圍，或是改變您儲存貯體和物件的現有政策和 ACL。 **Note** 存取點沒有與它們相關聯的 ACL。如果您將此設定套用至某個存取點，它會作為通往基礎儲存貯體的通道。如果存取點已啟用此設定，則透過存取點提出的請求，其行為就如同基礎儲存貯體已啟用此設定，無論儲存貯體是否已實際啟用此設定。

名稱	描述
IgnorePublicAcls	將此選項設為 TRUE 會導致 Amazon S3 在儲存貯體及其中任何物件上忽略所有公有 ACL。此設定可讓您安全地封鎖 ACL 授予的公開存取，同時仍允許包含公有 ACL 的 PUT 物件呼叫 (與拒絕包含公有 ACL PUT 物件的 BlockPublicAcls 相反)。起用此設定不會影響任何現有 ACL 的持久性，也不會阻止設定新的公有 ACL。 **Note** 存取點沒有與它們相關聯的 ACL。如果您將此設定套用至某個存取點，它會作為通往基礎儲存貯體的通道。如果存取點已啟用此設定，則透過存取點提出的請求，其行為就如同基礎儲存貯體已啟用此設定，無論儲存貯體是否已實際啟用此設定。

名稱	描述
BlockPublicPolicy	若針對儲存貯體將此選項設為 TRUE，則當指定儲存貯體政策允許公開存取時，Amazon S3 會拒絕對 PUT Bucket 政策發出的呼叫。如果指定的儲存貯體政策允許公開存取，針對儲存貯體將此選項設為 TRUE 時，也會導致 Amazon S3 拒絕針對所有儲存貯體的相同帳戶存取點呼叫 PUT 存取點政策。 如果指定的政策 (存取點或基礎儲存貯體) 允許公開存取，則針對存取點將此選項設為 TRUE 時，將會導致 Amazon S3 拒絕透過存取點呼叫 PUT 存取點政策和 PUT 儲存貯體政策。 您可以使用此設定，允許使用者管理存取點和儲存貯體政策，而無須讓他們公開共享儲存貯體或其包含的物件。啟用此設定不會影響現有的存取點或儲存貯體政策。 **Important** 若要有效使用此設定，建議您在帳戶層級套用它。儲存貯體政策可讓使用者改變儲存貯體的封鎖公開存取設定。因此，具有變更儲存貯體政策許可的使用者可插入政策，允許他們停用儲存貯體的封鎖公開存取設定。若為整個帳戶啟用此設定，而非僅針對特定儲存貯體，則即使使用者改變儲存貯體政策來停用此設定，Amazon S3 仍會封鎖公有政策。
RestrictPublicBuckets	設定此選項可TRUE限制存取具有公用政策的存取點或值區，只能存取值區擁有者帳戶和存取點擁有者帳戶內的 AWS 服務主體和授權使用者。此設定會封鎖存取點或值區的所有跨帳戶存取 (AWS 服務主體除外)，同時仍允許帳戶內的使用者管理存取點或值區。 啟用此設定不會影響現有的存取點或儲存貯體政策，但 Amazon S3 會封鎖從任何公用存取點或儲存貯體政策衍生的公用及跨帳戶存取，包括對特定帳戶的非公用委派。

Important

- 呼叫 GET 儲存貯體 acl 和 GET 物件 acl 一律會傳回對指定儲存貯體或物件最有效的許可。例如，假設儲存貯體具有授予公開存取的 ACL，同時也啟用了 `IgnorePublicAcls` 設定。在這種情況下，GET 儲存貯體 ACL 會傳回的 ACL 會反映 Amazon S3 正在強制執行的存取許可，而非傳回與儲存貯體相關聯的實際 ACL。
- 封鎖公開存取設定不會改變現有政策或 ACL。因此，移除封鎖公開存取設定會導致具備公有政策或 ACL 的儲存貯體或物件再次開放公開存取。

在存取點執行封鎖公開存取操作

若要在存取點上執行封鎖公用存取作業，請使用 AWS CLI 服務`s3control`。

Important

請注意，建立存取點之後，目前無法變更存取點的封鎖公開存取設定。因此，為存取點指定封鎖公開存取設定的唯一方法是在建立存取點時併入這些設定。

「公有」的意義

ACL

如果儲存貯體或物件 ACL 將任何許可授予預先定義的 `AllUsers` 或 `AuthenticatedUsers` 群組的成員，則 Amazon S3 會將此 ACL 視為公有。如需預先定義群組的詳細資訊，請參閱 Amazon S3 預先定義的群組。

儲存貯體政策

評估儲存貯體政策時，Amazon S3 首先假設政策為公用。然後，評估政策來判斷其是否符合非公有的資格。若要被視為是非公有，儲存貯體政策必須僅將存取授予以下一或多個項目的固定值 (不包含萬用字元或 AWS Identity and Access Management 政策變數 的值)：

- AWS 主體、使用者、角色或服務主體 (例如`aws:PrincipalOrgID`)
- 一組無類別網域間路由選擇 (CIDR)，使用 `aws:SourceIp`。如需 CIDR 的詳細資訊，請參閱 RFC Editor 網站上的 RFC 4632。

> **Note**
>
> 根據具有非常廣泛 IP 範圍 (例如，0.0.0.0/1) 的 `aws:SourceIp` 條件金鑰授予存取權的儲存貯體政策會評估為「公有」。這包括比 /8 (若為 IPv4) 和 /32 (若為 IPv6) 更寬泛的值，但 RFC1918 私有範圍排除在外。封鎖公開存取會拒絕這些「公有」政策，並防止跨帳戶存取已使用這些「公有」政策的儲存貯體。

- `aws:SourceArn`
- `aws:SourceVpc`
- `aws:SourceVpce`
- `aws:SourceOwner`
- `aws:SourceAccount`
- `s3:x-amz-server-side-encryption-aws-kms-key-id`
- `aws:userid`，位於 "AROLEID:*" 模式之外
- `s3:DataAccessPointArn`

> **Note**
>
> 在儲存貯體政策中使用時，只要帳戶 ID 為固定，此值就可以包含存取點名稱的萬用字元，而不會將該政策轉譯為公開。例如，允許存取 `arn:aws:s3:us-west-2:123456789012:accesspoint/*` 將允許存取與區域 123456789012 中帳戶 us-west-2 相關聯的任何存取點，而不會將儲存貯體政策轉譯為公開。請注意，存取點政策的此項行為是不同的。如需詳細資訊，請參閱 存取點。

- `s3:DataAccessPointAccount`

如需儲存貯體政策的詳細資訊，請參閱「Amazon S3 的存儲桶政策」。

Example ：公有儲存貯體政策

在這些規則之下，下列範例政策會被視為公用。

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
```

```
  "Effect": "Allow"
 }
```

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": { "StringLike": {"aws:SourceVpc": "vpc-*"}}
 }
```

您可以透過使用固定值來包含任何先前列出的條件索引鍵，使這些政策成為非公有。例如，您可以透過將 aws:SourceVpc 設為固定值來使上述最後一個政策成為非公用，如下所示。

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": {"StringEquals": {"aws:SourceVpc": "vpc-91237329"}}
 }
```

Amazon S3 如何評估同時包含公開及非公開存取授權的政策。

此範例示範 Amazon S3 如何評估同時包含公開及非公開存取授權的政策。

假設儲存貯體具有政策，會將存取授予一組固定的委託人。在先前說明的規則之下，此政策便不是公有。因此，若您啟用 RestrictPublicBuckets 設定，政策仍會以已寫入的狀態繼續生效，因為 RestrictPublicBuckets 只會套用到具有公有政策的儲存貯體。不過，如果您將公有陳述式新增至政策，則 RestrictPublicBuckets 會在儲存貯體上生效。它只允許值區擁有者帳戶的 AWS 服務主體和授權使用者存取值區。

例如，假設 "Account-1" 擁有的儲存貯體具有一個政策，該政策包含以下內容：

1. 授與存取權的陳述式 AWS CloudTrail (這是 AWS 服務主體)
2. 將存取授予 "Account-2" 帳戶的陳述式
3. 公開授予存取的陳述式 (例如透過使用無限制的 Condition 來指定 "Principal": "*")

此政策因為第三個陳述式，所以符合公有的資格。設定並RestrictPublicBuckets啟用此政策後，Amazon S3 只允許存取 CloudTrail。即使第 2 個陳述式並非公有，Amazon S3 也會

停用 "Account-2" 的存取。這次因為第 3 個陳述式會使整個政策都成為公有狀態，因此會套用 RestrictPublicBuckets。因此，即使政策將存取委派給指定的帳戶 "Account-2"，Amazon S3 還是會停用跨帳戶存取。但是，若您將第 3 個陳述式從政策中移除，則政策便不符合公有的資格，不再套用 RestrictPublicBuckets。因此，"Account-2" 會重新取得儲存貯體的存取，即使您將 RestrictPublicBuckets 維持在啟用狀態。

存取點

相較於儲存貯體，Amazon S3 評估存取點的封鎖公開存取設定時，方式略有不同。Amazon S3 套用以判斷存取點政策為公開的規則，在儲存貯體與存取點間通常是相同的，但下列情況除外：

- 無論其存取點政策的內容為何，具有 VPC 網路來源的存取點一律會被視為非公開。
- 使用 s3:DataAccessPointArn 授予存取給一組存取點的存取點政策會被視為公開。請注意，儲存貯體政策的此行為是不同的。例如，授予存取給符合 s3:DataAccessPointArn 的 arn:aws:s3:us-west-2:123456789012:accesspoint/* 的儲存貯體政策不會被視為公開。不過，存取點政策中的相同陳述式會將該存取點轉譯成公開。

使用 IAM Access Analyzer for S3 來檢閱公用儲存貯體

您可以使用 IAM Access Analyzer for S3，檢閱以儲存貯體 ACL、儲存貯體政策或存取點政策授予公開存取的儲存貯體。適用於 S3 的 IAM Access Analyzer 會向您發出設定為允許存取網際網路或其他 AWS 帳戶任何人 (包括組織 AWS 帳戶 外部) 的儲存貯體提醒您。針對每個公開或共用儲存貯體，您會收到報告公開或共用存取來源和層級的發現項目。

在 IAM Access Analyzer for S3 中，只要按一下滑鼠，就可以封鎖對儲存貯體的所有公開存取。您還可以深入檢視儲存貯體的層級許可設定，以設定精細的存取層級。對於需要公開或共用存取的特定和經驗證使用案例，您可以將對儲存貯體的發現項目存檔，以確認並記錄您要讓儲存貯體保持公開或共用。

在極少數情況下，對於 Amazon S3 封鎖公開存取評估回報為公開的儲存貯體，IAM Access Analyzer for S3 可能回報沒有問題。這是因為對於目前動作及日後可能新增的任何潛在動作，Amazon S3 封鎖公開存取會檢閱相關政策，導致儲存貯體變成公開。另一方面，IAM Access Analyzer for S3 只會在處於評估存取狀態時，分析為 Amazon S3 服務指定的目前動作。

如需 IAM Access Analyzer for S3 的詳細資訊，請參閱 使用 IAM Access Analyzer for S3 檢閱儲存貯體存取權。

許可

若要使用 Amazon S3 封鎖公開存取功能，您必須具備下列許可。

操作	必要許可
GET 儲存貯體政策狀態	`s3:GetBucketPolicyStatus`
GET 儲存貯體封鎖公開存取設定	`s3:GetBucketPublicAccessBlock`
PUT 儲存貯體封鎖公開存取設定	`s3:PutBucketPublicAccessBlock`
DELETE 儲存貯體封鎖公開存取設定	`s3:PutBucketPublicAccessBlock`
GET 帳戶封鎖公開存取設定	`s3:GetAccountPublicAccessBlock`
PUT 帳戶封鎖公開存取設定	`s3:PutAccountPublicAccessBlock`
DELETE 帳戶封鎖公開存取設定	`s3:PutAccountPublicAccessBlock`
PUT 存取點封鎖公開存取設定	`s3:CreateAccessPoint`

> **Note**
>
> DELETE 操作需要的許可與 PUT 操作相同。DELETE 操作沒有獨立的許可。

設定封鎖公開存取

如需為您 AWS 帳戶 和 Amazon S3 儲存貯體設定區塊公開存取的詳細資訊，請參閱下列主題。

- 為您的帳戶設定封鎖公開存取
- 為您的 S3 儲存貯體設定封鎖公開存取

為您的帳戶設定封鎖公開存取

Amazon S3 封鎖公開存取功能可提供存取點、儲存貯體和帳戶的設定，以協助您管理對 Amazon S3 資源的公開存取。依預設，新的儲存貯體、存取點和物件不允許公開存取。

如需詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

> **Note**
>
> 帳戶層級設定會覆寫個別物件上的設定。將您的帳戶設定為封鎖公開存取，將會覆寫對帳戶內個別物件所做的任何公開存取設定。

您可以使用 S3 主控台 AWS CLI、開 AWS 發套件和 REST API，為帳戶中的所有儲存貯體設定封鎖公用存取設定。如需詳細資訊，請參閱「以下各節」。

若要設定儲存貯體的封鎖公開存取設定，請參閱 為您的 S3 儲存貯體設定封鎖公開存取。如需存取點的詳細資訊，請參閱 在存取點執行封鎖公開存取操作。

使用 S3 主控台

Amazon S3 封鎖公開存取可防止套用任何設定而允許公開存取 S3 儲存貯體內的資料。本節說明如何編輯 AWS 帳戶中所有 S3 儲存貯體的封鎖公開存取設定。如需封鎖公開存取的詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

編輯中所有 S3 儲存貯體的封鎖公用存取設定 AWS 帳戶

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 選擇 Block Public Access settings for this account (此帳戶的封鎖公開存取設定)。
3. 選擇 Edit (編輯) 以變更 AWS 帳戶中所有儲存貯體的封鎖公開存取設定。
4. 選擇您要變更的設定，然後選擇 Save changes (儲存變更)。
5. 出現確認提示時，輸入 **`confirm`**。然後選擇 Confirm (確認) 以儲存變更。

使用 AWS CLI

您可以透過 AWS CLI使用 Amazon S3 封鎖公開存取。如需有關設定和使用的詳細資訊 AWS CLI，請參閱什麼是 AWS Command Line Interface?

帳戶

若要在帳戶上執行封鎖公開存取操作，請使用 AWS CLI 服務 `s3control`。使用此服務的帳戶層級操作如下：

- PUT PublicAccessBlock （對於一個帳戶）
- GET PublicAccessBlock （對於一個帳戶）

- 刪除 PublicAccessBlock （對於一個帳戶）

如需其他資訊和範例，請參閱〈AWS CLI 參考〉put-public-access-block中的〈〉。

使用 AWS 軟體開發套件

Java

下列範例說明如何使用 Amazon S3 區塊公開存取搭配使 AWS SDK for Java 用，將公用存取區塊組態放在 Amazon S3 帳戶上。

```
AWSS3ControlClientBuilder controlClientBuilder =
 AWSS3ControlClientBuilder.standard();
controlClientBuilder.setRegion(<region>);
controlClientBuilder.setCredentials(<credentials>);

AWSS3Control client = controlClientBuilder.build();
client.putPublicAccessBlock(new PutPublicAccessBlockRequest()
  .withAccountId(<account-id>)
  .withPublicAccessBlockConfiguration(new PublicAccessBlockConfiguration()
    .withIgnorePublicAcls(<value>)
    .withBlockPublicAcls(<value>)
    .withBlockPublicPolicy(<value>)
    .withRestrictPublicBuckets(<value>)));
```

Important

此範例僅適用於帳戶層級操作，這類操作會使用 `AWSS3Control` 用戶端類別。針對儲存貯體層級操作，請參閱先前的範例。

Other SDKs

如需使用其他 AWS SDK 的相關資訊，請參閱使用開發 AWS 套件使用 Amazon S3 進行開發。

使用 REST API

如需有關透過 REST API 使用 Amazon S3 封鎖公開存取的資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列主題。

- 帳戶層級操作
  - 放 PublicAccessBlock
  - 獲取 PublicAccessBlock
  - 刪除 PublicAccessBlock

為您的 S3 儲存貯體設定封鎖公開存取

Amazon S3 封鎖公開存取功能可提供存取點、儲存貯體和帳戶的設定，以協助您管理對 Amazon S3 資源的公開存取。依預設，新的儲存貯體、存取點和物件不允許公開存取。

如需詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

您可以使用 S3 主控台 AWS CLI、開 AWS 發套件和 REST API 授與一或多個儲存貯體的公用存取權。您也可以封鎖對已公開之儲存貯體的公開存取。如需詳細資訊，請參閱「以下各節」。

若要為您帳戶中的每個儲存貯體設定封鎖公開存取，請參閱 為您的帳戶設定封鎖公開存取。如需有關為存取點設定封鎖公開存取的資訊，請參閱 在存取點執行封鎖公開存取操作。

使用 S3 主控台

Amazon S3 封鎖公開存取可防止套用任何設定而允許公開存取 S3 儲存貯體內的資料。本節說明如何編輯一或多個 S3 儲存貯體的封鎖公開存取設定。如需使用 AWS CLI、開 AWS 發套件和 Amazon S3 REST API 封鎖公用存取的相關資訊，請參閱封鎖對 Amazon S3 儲存體的公開存取權。

您可以在 Buckets (儲存貯體)清單中查看您的儲存貯體是否可公開存取。在 Access (存取)欄中，Amazon S3 標示儲存貯體的許可，如下所示：

- 公開 - 每個人都擁有下列其中一或多項的存取：列出物件、寫入物件、讀取和寫入許可。
- 物件可為公開 - 儲存貯體非公開，但具備適當許可的人員都可授予物件公開存取。
- 儲存貯體和物件不可為公開 - 儲存貯體和物件不可公開存取。
- 只有此帳戶的授權使用者 — 存取權限會對此帳戶和 AWS 服務主體中的 IAM 使用者和角色隔離，因為存在授予公用存取權的政策。

您也可以依存取類型來篩選儲存貯體搜尋。從 Search for buckets (搜尋儲存貯體) 列旁的下拉式清單選擇存取類型。

如果您在列出儲存貯體及其公用存取設定時看到 `Error`，則您可能沒有所需的許可。檢查以確保您已將下列許可新增至您的使用者或角色政策中：

```
s3:GetAccountPublicAccessBlock
s3:GetBucketPublicAccessBlock
s3:GetBucketPolicyStatus
s3:GetBucketLocation
s3:GetBucketAcl
s3:ListAccessPoints
s3:ListAllMyBuckets
```

在極少數狀況下，請求也可能因 AWS 區域 中斷而失敗。

編輯單個 S3 儲存貯體的 Amazon S3 封鎖公開存取設定

如果您需要變更單一 S3 儲存貯體的公開存取設定，請遵循下列步驟。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Bucket name (儲存貯體名稱) 清單中，選擇所需的儲存貯體名稱。
3. 選擇 Permissions (許可)。
4. 選擇 Edit (編輯) 以變更儲存貯體的公開存取設定。如需四個 Amazon S3 封鎖公開存取設定的詳細資訊，請參閱「封鎖公開存取設定」。
5. 選擇您希望變更的設定，然後選擇 Save (儲存)。
6. 出現確認提示時，輸入 **`confirm`**。然後選擇 Confirm (確認) 以儲存變更。

建立儲存貯體時，您可以變更 Amazon S3 封鎖公開存取設定。如需詳細資訊，請參閱 建立儲存貯體。

使用 AWS CLI

若要封鎖值區上的公開存取，或刪除公用存取區塊，請使用 AWS CLI 服務`s3api`。使用此服務的儲存貯體層級操作如下：

- 放 PublicAccessBlock （ 用於桶 ）
- GET PublicAccessBlock （ 用於一個桶 ）
- 刪除 PublicAccessBlock （ 用於存儲桶 ）
- 獲取 BucketPolicyStatus

如需詳細資訊和範例，請參閱〈AWS CLI 參考〉put-public-access-block中的〈〉。

使用 AWS 軟體開發套件

Java

```
AmazonS3 client = AmazonS3ClientBuilder.standard()
   .withCredentials(<credentials>)
   .build();

client.setPublicAccessBlock(new SetPublicAccessBlockRequest()
  .withBucketName(<bucket-name>)
  .withPublicAccessBlockConfiguration(new PublicAccessBlockConfiguration()
    .withBlockPublicAcls(<value>)
    .withIgnorePublicAcls(<value>)
    .withBlockPublicPolicy(<value>)
    .withRestrictPublicBuckets(<value>)));
```

> **Important**
>
> 此範例僅適用於儲存貯體層級操作，這類操作會使用 AmazonS3 用戶端類別。針對帳戶層級操作，請參閱以下範例。

Other SDKs

如需使用其他 AWS SDK 的相關資訊，請參閱使用開發 AWS 套件使用 Amazon S3 進行開發。

使用 REST API

如需有關透過 REST API 使用 Amazon S3 封鎖公開存取的資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列主題。

- 儲存貯體層級操作
  - 放 PublicAccessBlock
  - 獲取 PublicAccessBlock
  - 刪除 PublicAccessBlock
  - 獲取 BucketPolicyStatus

使用 IAM Access Analyzer for S3 檢閱儲存貯體存取權

適用於 S3 的 IAM Access Analyzer 會向您發出設定為允許存取網際網路或其他 AWS 帳戶任何人 (包括組織 AWS 帳戶 外部) 的 S3 儲存貯體提醒您。針對每個公開或共用儲存貯體，您會收到公開或共用存取來源和層級的發現項目。例如，IAM Access Analyzer for S3 可能會顯示具有透過儲存貯體存取控制清單 (ACL)、儲存貯體政策、多區域存取點政策或存取點政策提供之讀取或寫入存取權限的儲存貯體。使用這些問題清單，您可以採取立即且精確的更正動作，將儲存貯體存取還原成您想要的內容。

在 IAM Access Analyzer for S3 中檢閱有風險的儲存貯體時，只要按一下就可以封鎖對儲存貯體的所有公開存取。除非您需要公開存取以支援特定使用案例，否則建議您封鎖對儲存貯體的所有存取。在封鎖所有公開存取之前，請確定您的應用程式在沒有公開存取的情況下可繼續正常運作。如需詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

您還可以深入檢視儲存貯體的層級權限設定，以設定精細的存取層級。對於需要公開存取的特定和經驗證使用案例 (例如靜態網站託管、公開下載或跨帳戶共用)，您可以將對儲存貯體的發現項目存檔，以確認並記錄您要讓儲存貯體保持公開或共用。您可以隨時再次瀏覽和修改這些儲存貯體組態。您也可以將發現項目下載為 CSV 報告，供稽核之用。

您無需額外付費，即可在 Amazon S3 主控台上使用 IAM Access Analyzer for S3。IAM Access Analyzer for S3 採用 AWS Identity and Access Management (IAM) IAM Access Analyzer 技術。若要在 Amazon S3 主控台中使用適用於 S3 的 IAM 存取分析器，您必須造訪 IAM 主控台，並針對每個區域啟用 IAM 存取分析器。

如需 IAM 存取分析器的詳細資訊，請參閱什麼是 IAM 存取分析器？ 在 IAM 使用者指南中。如需 IAM Access Analyzer for S3 的詳細資訊，請參閱下列各節。

Important

- IAM Access Analyzer for S3 需要具備帳戶層級的分析器。若要使用適用於 S3 的 IAM 存取分析器，您必須造訪 IAM 存取分析器，並建立以帳戶做為信任區域的分析器。如需詳細資訊，請參閱《IAM 使用者指南》中的啟用 IAM Access Analyzer。
- IAM Access Analyzer for S3 不會分析附加到跨帳戶存取點的存取點政策。發生此行為的原因是存取點及其政策位於信任區域 (亦即帳戶) 之外。如果您未將 `RestrictPublicBuckets` 封鎖公開存取設定套用至儲存貯體或帳戶，則將存取權委派給跨帳戶存取點的儲存貯體會列示在具有公開存取權的儲存貯體下。當您套用`RestrictPublicBuckets`封鎖公開存取設定時，值區會在具有其他 AWS 帳戶 存取權的值區 (包括第三方) 下報告 AWS 帳戶。

- 新增或修改儲存貯體政策或儲存貯體 ACL 後，IAM Access Analyzer 會在 30 分鐘內根據變更產生並更新調查結果。在您變更設定後，最多可能有 6 小時的時間，與帳戶層級封鎖公開存取設定相關的發現項目不會產生或更新。建立、刪除多區域存取點或變更其政策後，最多可能有六小時的時間，與多區域存取點相關的發現項目不會產生或更新。

主題

IAM Access Analyzer for S3 提供哪些資訊？

IAM Access Analyzer for S3 會提供可在 AWS 帳戶外部存取的儲存貯體調查結果。網際網路上的任何人都可以存取列在 Buckets with public access (具有公開存取的儲存貯體) 下的儲存貯體。如果 IAM Access Analyzer for S3 發現公有儲存貯體，則顯示區域中公有儲存貯體數量的頁面頂端也會出現警告。列在「可從其他存取權的值區」下的值區 AWS 帳戶 — 包括第三方 AWS 帳戶會有條件地與其他廠商共用 AWS 帳戶，包括組織外部的帳戶。

IAM Access Analyzer for S3 會針對每個儲存貯體提供下列資訊：

- 儲存貯體名稱
- Access Analyzer 發現的項目 - 當 IAM Access Analyzer for S3 發現公開或共用儲存貯體存取時，就會提供此資訊。
- 共用方式 - 指出儲存貯體是透過儲存貯體政策、儲存貯體 ACL、多區域存取點政策或存取點政策共用。多區域存取點和跨帳戶存取點會在存取點下顯現。儲存貯體可透過政策和 ACL 共用。如果您要尋找並檢閱儲存貯體存取的來源，可以使用此欄中的資訊作為採取立即且精確的更正動作的起點。
- 狀態 - 儲存貯體問題清單的狀態。IAM Access Analyzer for S3 會顯示所有公有和共用儲存貯體的調查結果。
  - 作用中 - 問題清單尚未經過檢閱。

  - 已封存 - 問題清單已如預期經過檢閱和確認。
  - 所有-公開或與其 AWS 帳戶他人共用的值區 (包括組織 AWS 帳戶 外部) 的所有發現項目。
- 存取層級 - 授予儲存貯體的存取許可：
  - 列出 - 列出相關資源。
  - 讀取 - 讀取但不編輯資源內容和屬性。
  - 寫入 - 建立、刪除或修改資源。
  - 許可 - 授予或修改資源許可。
  - 標記 - 更新與資源相關聯的標籤。

啟用 IAM Access Analyzer for S3

若要使用 IAM Access Analyzer for S3，您必須完成下列先決條件步驟。

1. 授予所需的許可。

   如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM Access Analyzer 的必要許可。
2. 前往 IAM 主控台，為要使用 IAM Access Analyzer 的每個區域建立帳戶層級分析器。

   IAM Access Analyzer for S3 需要具備帳戶層級的分析器。若要使用 IAM Access Analyzer for S3，您必須建立以帳戶作為信任區域的分析器。如需詳細資訊，請參閱《IAM 使用者指南》中的啟用 IAM Access Analyzer。

封鎖所有公開存取

如果您想要按一下就封鎖對儲存貯體的所有存取，則可使用 IAM Access Analyzer for S3 中的封鎖所有公開存取按鈕。當您封鎖對儲存貯體的所有公開存取時，不會授予任何公開存取。除非您需要公開存取以支援特定和經驗證使用案例，否則建議您封鎖對儲存貯體的所有公開存取。在封鎖所有公開存取之前，請確定您的應用程式在沒有公開存取的情況下可繼續正常運作。

如果您不想封鎖對儲存貯體的所有公開存取，可以在 Amazon S3 主控台上編輯封鎖公開存取設定，為儲存貯體設定精細的存取層級。如需詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

在極少數情況下，對於 Amazon S3 封鎖公開存取評估回報為公開的儲存貯體，IAM Access Analyzer for S3 可能回報沒有問題。這是因為對於目前動作及日後可能新增的任何潛在動作，Amazon S3 封鎖公開存取會檢閱相關政策，導致儲存貯體變成公開。另一方面，IAM Access Analyzer for S3 只會在處於評估存取狀態時，分析為 Amazon S3 服務指定的目前動作。

使用 IAM Access Analyzer for S3 來封鎖對儲存貯體的所有公開存取

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側的導覽窗格中，於 Dashboards (儀表板) 下，選擇 Access Analyzer for S3。
3. 在 IAM Access Analyzer for S3 中，選擇某個儲存貯體。
4. 選擇 Block all public access (封鎖所有公用存取)。
5. 若要確認您要封鎖對儲存貯體的所有公開存取，請在 Block all public access (bucket settings) (封鎖所有公開存取 (儲存貯體設定)) 中輸入 **`confirm`**。

   Amazon S3 隨即會封鎖對儲存貯體的所有公開存取。儲存貯體調查結果的狀態會更新為已解決，且該儲存貯體會從 IAM Access Analyzer for S3 清單中消失。如果您想要檢閱已解決的值區，請在 IAM 主控台上開啟 IAM 存取分析器。

檢閱和變更儲存貯體存取

如果您不打算授與公眾或其他人的存取權 AWS 帳戶，包括組織外的帳戶，您可以修改值區 ACL、儲存貯體政策、多區域存取點政策或存取點政策，以移除值區的存取權。Shared through (共用方式) 欄位會顯示儲存貯體存取的所有來源：儲存貯體政策、儲存貯體 ACL 及/或存取點政策。多區域存取點和跨帳户存取點會在存取點下顯現。

若要檢閱和變更儲存貯體政策、儲存貯體 ACL、多區域存取點或存取點政策

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在導覽窗格中，選擇 Access Analyzer for S3。
3. 若要查看公開存取或共用存取是否透過儲存貯體政策、儲存貯體 ACL、多區域存取點政策或存取點政策授予，請查看 Shared through (共用方式) 欄。
4. 在 Buckets (儲存貯體) 底下，選擇具有要變更或檢閱之儲存貯體政策、儲存貯體 ACL、多區域存取點政策或存取點政策的儲存貯體名稱。
5. 如果您要變更或檢視儲存貯體 ACL：

   a. 選擇 Permissions (許可)。

   b. 選擇 Access Control List (存取控制清單)。

   c. 檢閱您的儲存貯體 ACL，並視需要進行變更。

      如需詳細資訊，請參閱 設定 ACL。

6. 如果您要變更或檢閱儲存貯體政策：

   a. 選擇 Permissions (許可)。

   b. 選擇 Bucket Policy (儲存貯體政策)。

   c. 視需要檢閱或變更儲存貯體政策。

      如需詳細資訊，請參閱 使用 Amazon S3 主控台新增儲存貯體政策。

7. 如果您要變更或檢視多區域存取點政策：

   a. 選擇 Multi-Region Access Point (多區域存取點)。

   b. 選擇多區域存取點名稱。

   c. 視需要檢閱或變更您的多區域存取點政策。

      如需詳細資訊，請參閱 許可。

8. 如果您要檢閱或變更存取點政策：

   a. 選擇 Access points (存取點)。

   b. 選擇存取點名稱。

   c. 視需要檢閱或變更存取權。

      如需詳細資訊，請參閱 透過 Amazon S3 主控台使用 Amazon S3 存取點。

   如果您編輯或移除儲存貯體 ACL、儲存貯體或存取點政策以移除公開或共用存取，儲存貯體發現項目的狀態會更新為已解決。已解決的儲存貯體發現項目會從適用於 S3 清單的 IAM 存取分析器中消失，但您可以在 IAM 存取分析器中檢視它們。

存檔儲存貯體發現項目

如果值區授予公用或其他人的存取權 AWS 帳戶，包括組織外的帳戶，以支援特定使用案例 (例如靜態網站、公開下載或跨帳戶共用)，您可以封存值區的發現項目。當您將儲存貯體發現項目存檔時，表示您確認並記錄要讓儲存貯體保持公開或共用。已封存的儲存貯體調查結果會保留在 IAM Access Analyzer for S3 清單中，以便您隨時掌握公有或共用的儲存貯體。

在 IAM Access Analyzer for S3 中封存儲存貯體調查結果

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在導覽窗格中，選擇 Access Analyzer for S3。

3. 在 IAM Access Analyzer for S3 中，選擇作用中的儲存貯體。
4. 若要確認您希望公眾或其他人存取此值區 AWS 帳戶，包括組織外的帳戶，請選擇「封存」。
5. 輸入 **`confirm`**，然後選擇 Archive (存檔)。

啟用存檔的儲存貯體

將發現項目存檔之後，您一律可以重新瀏覽它們，並將其狀態變更回作用中，指出該儲存貯體需要另一次檢閱。

在 IAM Access Analyzer for S3 中啟用封存的儲存貯體調查結果

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在導覽窗格中，選擇 Access Analyzer for S3。
3. 選擇存檔的儲存貯體發現項目。
4. 選擇 Mark as active (標記為作用中)。

檢視發現項目詳細資料

如果您需要查看值區的詳細資訊，可以在 IAM 主控台的 IAM Access Analyzer 中開啟儲存貯體尋找詳細資料。

在 IAM Access Analyzer for S3 中檢視調查結果詳細資訊

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在導覽窗格中，選擇 Access Analyzer for S3。
3. 在 IAM Access Analyzer for S3 中，選擇某個儲存貯體。
4. 請選擇 View Details (查看詳細資訊)。

   發現項目詳細資料會在 IAM 主控台的 IAM 存取分析器中開啟。

下載 IAM Access Analyzer for S3 報告

您可以將儲存貯體發現項目下載為 CSV 報告，以供稽核之用。此報告包含的資訊與 Amazon S3 主控台的 IAM Access Analyzer for S3 所顯示的資訊相同。

下載報告

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側的導覽窗格中，選擇 Access Analyzer for S3。
3. 在地區篩選器中，選擇地區。

   IAM Access Analyzer for S3 會隨即更新，進而顯示所選區域的儲存貯體。
4. 選擇 Download report (下載報告)。

   CSV 報告會產生並儲存至您的電腦。

使用儲存貯體擁有者條件驗證儲存貯體擁有權

Amazon S3 儲存貯體擁有者條件可確保您在 S3 操作中使用的儲存貯體屬於您預期的 AWS 帳戶 儲存貯體。

大多數 S3 操作從特定 S3 儲存貯體讀取或寫入。這些操作包含上傳、複製和下載物件、擷取或修改儲存貯體組態，以及擷取或修改物件組態。當您執行這些操作時，您可以在請求中加入其名稱來指定要使用的儲存貯體。例如，若要從 S3 擷取物件，您可以提出請求，指定儲存貯體的名稱和要從該儲存貯體擷取的物件金鑰。

由於 Amazon S3 會根據儲存貯體的名稱識別儲存貯體，因此在請求中使用錯誤儲存貯體名稱的應用程式可能會無意中針對與預期不同的儲存貯體執行操作。為了避免在這種情況下意外的儲存貯體互動，您可以使用儲存貯體擁有者條件。儲存貯體擁有者條件可讓您驗證目標儲存貯體是否由預期的 AWS 帳戶所擁有，提供額外的保證，讓您的 S3 操作具有您想要的效果。

主題

- 何時使用儲存貯體擁有者條件
- 驗證儲存貯體擁有者
- 範例
- 法規與限制

何時使用儲存貯體擁有者條件

每當您執行支援的 S3 操作並知道預期儲存貯體擁有者的帳戶 ID時，建議您使用儲存貯體擁有者條件。儲存貯體擁有者條件適用於所有 S3 物件操作和大多數 S3 儲存貯體操作。如需不支援儲存貯體擁有者條件的 S3 操作清單，請參閱 法規與限制。

若要查看使用時段擁有者條件的好處，請考慮下列涉及 AWS 客戶 Bea 的案例：

1. Bea 開發使用 Amazon S3 的應用程式。在開發過程中，Bea 使用僅測試 AWS 帳戶 來創建一個名為的存儲桶`bea-data-test`，並配置她的應用程序以向其發出請求。`bea-data-test`
2. Bea 部署了她的應用程式，但忘記重新配置應用程式以在生產 AWS 帳戶中使用儲存貯體。
3. 在生產中，BEA 的應用程式提出請求至 `bea-data-test` 並且已成功。這會導致生產資料寫入 BEA 測試帳戶中的儲存貯體。

BEA 可以透過使用儲存貯體擁有者條件來幫助防止這種情況。在存儲桶所有者條件下，Bea 可以在其請求中包含預期存儲桶所有者的 AWS 帳戶 ID。然後，Amazon S3 會在處理每個請求之前檢查儲存貯體擁有者的帳戶 ID。如果實際的儲存貯體擁有者與預期的儲存貯體擁有者不符，請求就會失敗。

如果 BEA 使用儲存貯體擁有者條件，則先前描述的案例不會導致 BEA 的應用程式不小心寫入測試時段。相反地，她的應用程式在步驟 3 所做的請求將會失敗，並顯示 `Access Denied` 錯誤訊息。透過使用儲存貯體擁有者條件，BEA 有助於消除在錯誤 AWS 帳戶中意外與儲存貯體互動的風險。

驗證儲存貯體擁有者

若要使用儲存貯體擁有者條件，請求包含參數，指定預期的儲存貯體擁有者。大多數 S3 操作只涉及一個儲存貯體，並且只需要此單一參數才能使用儲存貯體擁有者條件。對於 `CopyObject` 操作，此第一個參數指定目的地儲存貯體的預期擁有者，而且您可以包含第二個參數來指定來源儲存貯體的預期擁有者。

當您提出包含儲存貯體擁有者條件參數的請求時，S3 會針對指定的參數檢查儲存貯體擁有者的帳戶 ID，然後再處理請求。如果參數符合儲存貯體擁有者的帳戶 ID，S3 會處理請求。如果參數不符合儲存貯體擁有者的帳戶 ID，請求會失敗並顯示 `Access Denied` 錯誤訊息。

您可以將儲存貯體擁有者條件與 AWS Command Line Interface (AWS CLI)、 AWS 開發套件和 Amazon S3 REST API 搭配使用。將儲存貯體擁有者條件與 AWS CLI 和 Amazon S3 REST API 搭配使用時，請使用下列參數名稱。

存取方法	非複製操作的參數	複製操作來源參數	複製操作目的地參數
AWS CLI	`--expected-bucket-owner`	`--expected-source-bucket-owner`	`--expected-bucket-owner`

存取方法	非複製操作的參數	複製操作來源參數	複製操作目的地參數
Amazon S3 REST API	x-amz-expected-bucket-owner 標頭	x-amz-source-expected-bucket-owner 標頭	x-amz-expected-bucket-owner 標頭

使用儲存貯體擁有者條件搭配 AWS SDK 所需的參數名稱會視語言而有所不同。若要判斷必要的參數，請參閱您所需語言的 SDK 說明文件。您可以在 AWS 上建置的工具中找到 SDK 說明文件。

範例

下列範例說明如何使用 AWS CLI 或在 Amazon S3 中實作儲存貯體擁有者條件 AWS SDK for Java 2.x。

Example

範例：上傳物件

下列範例會使用儲存貯體擁有者條件將物件上傳至 S3 儲存貯體 *example-s3-bucket1*，以確保 *example-s3-bucket1* 為 AWS 帳戶 111122223333 所擁有。

AWS CLI

```
aws s3api put-object \
                --bucket example-s3-bucket1 --key exampleobject --
body example_file.txt \
                --expected-bucket-owner 111122223333
```

AWS SDK for Java 2.x

```
public void putObjectExample() {
    S3Client s3Client = S3Client.create();;
    PutObjectRequest request = PutObjectRequest.builder()
            .bucket("example-s3-bucket1")
            .key("exampleobject")
            .expectedBucketOwner("111122223333")
            .build();
    Path path = Paths.get("example_file.txt");
    s3Client.putObject(request, path);
}
```

Example

範例：複製物件

下列範例會將物件 object1 從 S3 儲存貯體 *example-s3-bucket1* 複製到 S3 儲存貯體 *example-s3-bucket2*。它會使用儲存貯體擁有者條件，以確保儲存貯體是由預期帳戶根據下表所擁有。

儲存貯體	預期的擁有者
example-s3-bucket1	111122223333
example-s3-bucket2	444455556666

AWS CLI

```
aws s3api copy-object --copy-source example-s3-bucket1/object1 \
                            --bucket example-s3-bucket2 --key object1copy \
                            --expected-source-bucket-owner 111122223333 --expected-
bucket-owner 444455556666
```

AWS SDK for Java 2.x

```
public void copyObjectExample() {
        S3Client s3Client = S3Client.create();
        CopyObjectRequest request = CopyObjectRequest.builder()
                .copySource("example-s3-bucket1/object1")
                .destinationBucket("example-s3-bucket2")
                .destinationKey("object1copy")
                .expectedSourceBucketOwner("111122223333")
                .expectedBucketOwner("444455556666")
                .build();
        s3Client.copyObject(request);
    }
```

Example

範例：擷取儲存貯體策略

下列範例會使用儲存貯體擁有者條件擷取 S3 儲存貯體的存取政策 *example-s3-bucket1*，以確保 *example-s3-bucket1* 為 AWS 帳戶 111122223333 所擁有。

AWS CLI

```
aws s3api get-bucket-policy --bucket example-s3-bucket1 --expected-bucket-
owner 111122223333
```

AWS SDK for Java 2.x

```
public void getBucketPolicyExample() {
    S3Client s3Client = S3Client.create();
    GetBucketPolicyRequest request = GetBucketPolicyRequest.builder()
            .bucket("example-s3-bucket1")
            .expectedBucketOwner("111122223333")
            .build();
    try {
        GetBucketPolicyResponse response = s3Client.getBucketPolicy(request);
    }
    catch (S3Exception e) {
        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it, so it returned an error response.
        e.printStackTrace();
    }
}
```

法規與限制

Amazon S3 儲存貯體擁有者條件具有下列限制和限制：

- 值區擁有者條件參數的值必須是 AWS 帳戶 ID (12 位數的數值)。不支援服務委託人。
- 存儲桶擁有者條件不適用於 CreateBucketListBuckets，或 AWS S3 控制中包含的任何操作。Amazon S3 會忽略對這些操作請求所包含的任何儲存貯體擁有者條件參數。
- 儲存貯體擁有者條件只會驗證驗證參數中指定的帳戶是否擁有該時段。儲存貯體擁有者條件不會檢查儲存貯體的組態。它也不保證儲存貯體的組態符合任何特定條件或匹配任何過去的狀態。

控制物件的擁有權並停用儲存貯體的 ACL

S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來控制上傳至儲存貯體之物件的擁有權，以及停用或啟用存取控制清單 (ACL)。根據預設，物件擁有權設定為「儲存貯體擁有者強制執行」設定，而且所有 ACL 都會停用。停用 ACL 時，儲存貯體擁有者會擁有儲存貯體中的所有物件，並使用存取管理政策專門管理對資料的存取。

Amazon S3 中的大多數現代使用案例不再需要使用 ACL，建議您將 ACL 保持停用狀態，除非在異常情況下必須個別控制每個物件的存取權。停用 ACL 後，您可以使用政策，更輕鬆地控制對儲存貯體中每個物件的存取，無論是誰上傳了您儲存貯體中的物件。

「物件擁有權」有三項設定，可讓您用來控制對上傳至儲存貯體之物件的擁有權，以及停用或啟用 ACL：

已停用 ACL

- 儲存貯體擁有者強制執行 (預設) - 停用 ACL，儲存貯體擁有者會自動擁有並完全控制儲存貯體中的每個物件。ACL 不再影響 S3 儲存貯體中資料的許可。儲存貯體使用政策來定義存取控制。

已啟用 ACL

- 儲存貯體擁有者偏好 – 儲存貯體擁有者擁有並完全控制其他帳戶使用 `bucket-owner-full-control` 標準 ACL 寫入儲存貯體的新物件。
- Object writer (物件寫入者) – 上傳物件的 AWS 帳戶 ，上傳的物件可擁有物件、完全控制該物件，並且可以透過 ACL 授權其他使用者存取該物件。

對於 S3 中的大多數現代使用案例，建議您將 ACL 保持停用狀態，方法為套用儲存貯體擁有者強制執行設定，並視需要使用您的儲存貯體政策與帳戶外的使用者共用資料。這種方法可簡化許可管理。您可以在新建立和現有儲存貯體上停用 ACL。針對新建立的儲存貯體，預設會停用 ACL。在現有儲存貯體中已有物件的情況下，停用 ACL 之後，物件和儲存貯體 ACL 不再是存取評估的一部分，並根據政策授予或拒絕存取。對於現有儲存貯體，您可以在停用 ACL 之後隨時重新啟用，並還原之前存在的儲存貯體和物件 ACL。

在停用 ACL 之前，建議您檢閱儲存貯體政策，以確保其涵蓋您打算將存取權授予帳戶外儲存貯體的所有方式。停用 ACL 之後，儲存貯體只接受未指定 ACL 的 PUT 請求或儲存貯體擁有者完全控制 ACL 的 PUT 請求，例如 `bucket-owner-full-control` 標準 ACL 或以 XML 表達此 ACL 的同等形式。支援儲存貯體擁有者完全控制 ACL 的現有應用程式不會受到影響。 PUT包含其他 ACL (例如，特定的自訂授權 AWS 帳戶) 的要求會失敗，並傳回含有`400`錯誤碼`AccessControlListNotSupported`的錯誤。

相比之下，具有「儲存貯體擁有者偏好」設定的儲存貯體會繼續接受並遵循儲存貯體和物件 ACL。使用此設定時，儲存貯體擁有者會自動擁有使用 `bucket-owner-full-control` 標準 ACL 寫入的新物件，而不是物件寫入者。所有其他 ACL 行為都會保留在原位。若要要求所有 Amazon S3 PUT 操作包含 `bucket-owner-full-control` 標準 ACL，您可以新增儲存貯體政策，只允許使用此 ACL 上傳物件。

若要查看哪些物件擁有權設定套用至儲存貯體，您可以使用 Amazon S3 Storage Lens 指標。S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。如需詳細資訊，請參閱使用 S3 Storage Lens 尋找物件擁有權設定。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱什麼是 S3 Express One Zone？ 和 目錄值區。

「物件擁有權」設定

此資料表顯示每個「物件擁有權」設定對 ACL、物件、物件擁有權和物件上傳的影響。

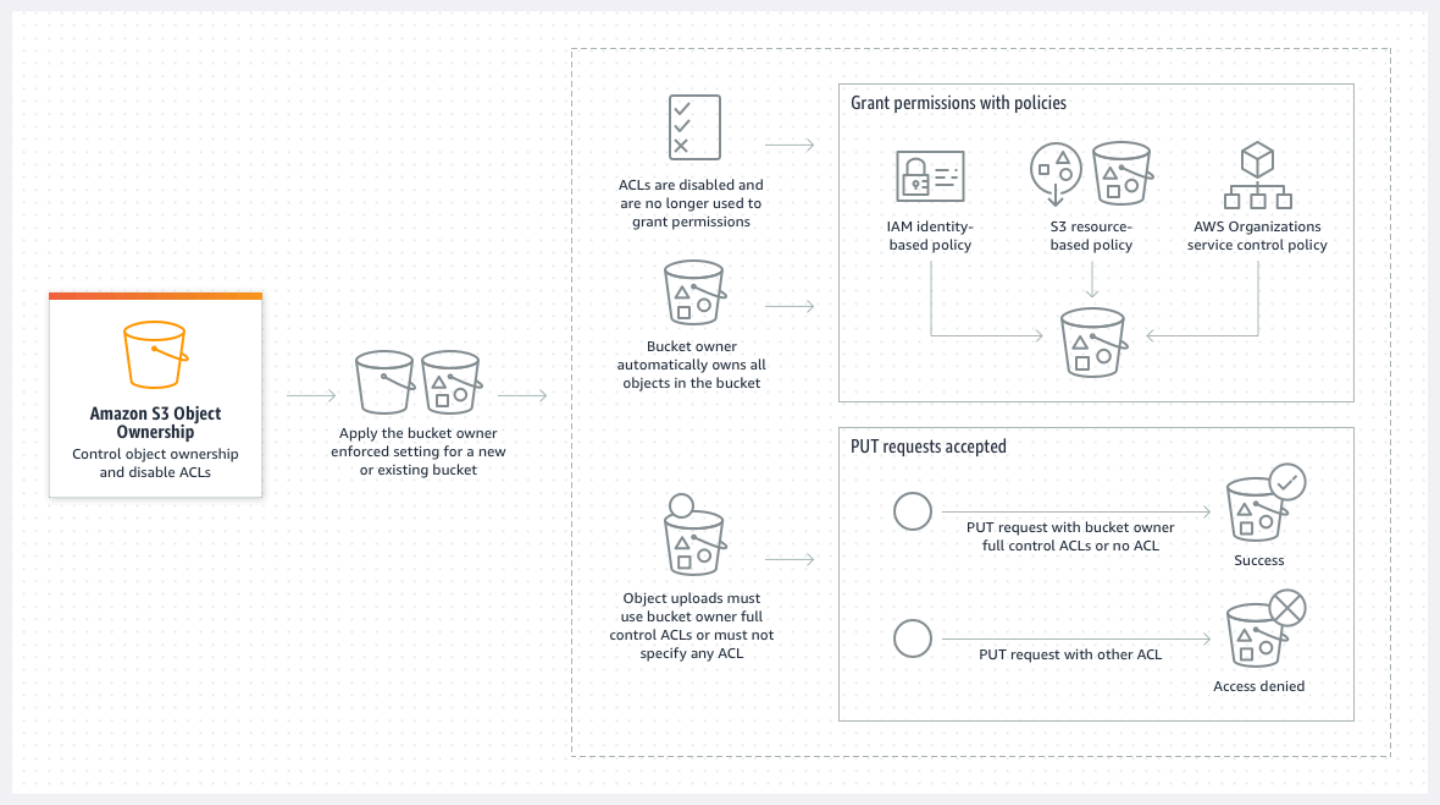
設定	適用對象	對物件所有權的影響	對 ACL 的影響	已接受上傳
儲存貯體擁有者強制執行 (預設)	全部新的和現有物件	儲存貯體擁有者擁有每個物件。	ACL 已停用，不再影響儲存貯體的存取許可。設定或更新 ACL 的請求失敗。不過，仍支援讀取 ACL 的請求。 儲存貯體擁有者擁有完整的擁有權和控制權。 物件寫入者不再擁有完整的擁有權和控制權。	儲存貯體擁有者完全控制 ACL 的上傳或未指定 ACL 的上傳
儲存貯體擁有者偏好	新物件	如果物件上傳包含 `bucket-owner-full-control` 標準 ACL，則儲存貯	ACL 可以更新，而且可以授予許可。	所有上傳

設定	適用對象	對物件所有權的影響	對 ACL 的影響	已接受上傳
		體擁有者會擁有物件。 與其他 ACL 一起上傳的物件是由寫入帳戶所擁有。	如果物件上傳包含 `bucket-owner-full-control` 標準 ACL，則儲存貯體擁有者擁有完整的存取控制權，而且物件寫入者不再擁有完整的存取控制權。	
物件寫入器	新物件	物件寫入者擁有物件。	ACL 可以更新，而且可以授予許可。 物件寫入者具有完整的存取控制權。	所有上傳

透過停用 ACL 引入的變更

套用物件擁有權的「儲存貯體擁有者強制執行」設定時，ACL 會停用，而且您會自動擁有並完全控制儲存貯體中的每個物件，而不會採取任何其他動作。儲存貯體擁有者強制執行是所有新建儲存貯體的預設設定。在套用儲存貯體擁有者強制執行設定之後，您會看到三個變更：

- 會停用所有儲存貯體 ACL 和物件 ACL，這可讓您身為儲存貯體擁有者擁有完整的存取權。在儲存貯體或物件上執行讀取 ACL 請求時，您會看到完整存取權僅提供給儲存貯體擁有者。
- 您身為儲存貯體擁有者會自動擁有並完全控制儲存貯體中的每個物件。
- ACL 不再影響儲存貯體的存取許可。因此，資料的存取控制是以政策為基礎，例如 IAM 政策、S3 儲存貯體政策、VPC 端點政策和組織 SCP。



如果您使用 S3 版本控制，則儲存貯體擁有者會擁有並且可以完全控制儲存貯體中的所有物件版本。套用「儲存貯體擁有者強制執行」設定不會新增物件的版本。

只有在使用儲存貯體擁有者完全控制 ACL 或未指定 ACL 時，新物件才能上傳到儲存貯體。如果物件指定任何其他 ACL，則物件上傳會失敗。如需詳細資訊，請參閱 故障診斷。

因為以下使用 AWS Command Line Interface (AWS CLI) 的範例 `PutObject` 操作包含 `bucket-owner-full-control` 標準 ACL，所以可將物件上傳至已停用 ACL 的儲存貯體。

```
aws s3api put-object --bucket DOC-EXAMPLE-BUCKET --key key-name --body path-to-file --
acl bucket-owner-full-control
```

因為以下 `PutObject` 操作未指定 ACL，所以對於已停用 ACL 的儲存貯體也會成功。

```
aws s3api put-object --bucket DOC-EXAMPLE-BUCKET --key key-name --body path-to-file
```

> **Note**
>
> 如果其他人在上傳後 AWS 帳戶 需要存取物件，您必須透過儲存貯體政策授予這些帳戶的其他權限。如需詳細資訊，請參閱 使用政策管理 Amazon S3 資源存取權的逐步解說。

重新啟用 ACL

您可以隨時從「儲存貯體擁有者強制執行」設定變更為另一個「物件擁有權」設定，以重新啟用 ACL。如果您在套用「儲存貯體擁有者強制執行」設定之前，先使用物件 ACL 進行許可管理，而且未將這些物件 ACL 許可遷移至儲存貯體政策，則重新啟用 ACL 之後，這些許可會還原。此外，在套用「儲存貯體擁有者強制執行」設定時寫入儲存貯體的物件仍然由儲存貯體擁有者擁有。

例如，如果您從「儲存貯體擁有者強制執行」設定變更回「物件寫入器」設定，身為儲存貯體擁有者，您就不再擁有之前由其他 AWS 帳戶擁有的物件，且不對其擁有完整控制權。相反，上傳帳戶再次擁有這些物件。其他帳戶擁有的物件會使用 ACL 來取得許可，因此您無法使用政策為這些物件授予許可。此外，您身為儲存貯體擁有者，仍然擁有在套用「儲存貯體擁有者強制執行」設定時寫入儲存貯體的任何物件。即使您重新啟用 ACL，這些物件也不屬於物件寫入者。

如需使用 AWS Management Console、 AWS Command Line Interface (CLI)、REST API 或 AWS SDK 啟用及管理 ACL 的指示，請參閱。設定 ACL

停用 ACL 的先決條件

停用現有儲存貯體的 ACL 之前，請先完成下列先決條件。

檢閱儲存貯體和物件 ACL，並遷移 ACL 許可

當您停用 ACL 時，儲存貯體和物件 ACL 所授予的許可不會再影響存取。停用 ACL 之前，請先檢閱儲存貯體和物件 ACL。

如果儲存貯體 ACL 將讀取或寫入許可授予帳戶外的其他人，則必須先將這些許可遷移至儲存貯體政策，才能套用「儲存貯體擁有者強制執行」設定。如果您未遷移授予帳戶外讀取或寫入存取權的儲存貯體 ACL，則套用「儲存貯體擁有者強制執行」設定的請求會失敗，並傳回 InvalidBucketAclWithObjectOwnership 錯誤代碼。

例如，如果您想要針對接收伺服器存取日誌的儲存貯體停用 ACL，則必須將 S3 日誌交付群組的儲存貯體 ACL 許可遷移至儲存貯體政策中的日誌記錄服務主體。如需詳細資訊，請參閱 授予 S3 日誌交付群組的存取權以進行伺服器存取日誌記錄。

如果您想要物件寫入者能夠完全控制其上傳的物件，則物件寫入者是您使用案例的最佳物件所有權設定。如果您想要控制個別物件層級的存取權，則儲存貯體擁有者偏好是最佳選擇。這些使用案例很少見。

若要檢閱 ACL 並將 ACL 許可遷移至儲存貯體政策，請參閱 停用 ACL 的先決條件。

識別需要 ACL 進行授權的所有請求

若要識別需要 ACL 進行授權的 Amazon S3 請求，您可以使用 Amazon S3 伺服器存取日誌或 AWS CloudTrail中的 `aclRequired` 值。如果請求需要 ACL 進行授權，或者您有指定 ACL 的 PUT 請求，則字串為 `Yes`。如果不需要 ACL，或者您要設定`bucket-owner-full-control`固定 ACL，或者儲存貯體政策允許這些請求，則 Amazon S3 伺服器存取日誌中的`aclRequired`值字串為 `-` ""，且在中不存在 CloudTrail。如需預期的 `aclRequired` 值詳細資訊，請參閱 常見 Amazon S3 請求的 `aclRequired` 值。

如果您有 `PutBucketAcl` 或 `PutObjectAcl` 請求具有授予 ACL 型許可的標頭，但 `bucket-owner-full-control` 固定 ACL 除外，則您必須先移除這些標頭，才能停用 ACL。否則，請求將會失敗。

對於需要 ACL 進行授權的所有其他請求，請將這些 ACL 許可移轉至儲存貯體政策。然後，在啟用儲存貯體擁有者強制執行設定之前，移除任何儲存貯體 ACL。

> **Note**
>
> 請勿移除物件 ACL。否則，依賴物件 ACL 取得許可的應用程式將失去存取權。

如果您發現沒有請求要求 ACL 進行授權，您可以繼續停用 ACL。如需識別請求的詳細資訊，請參閱 使用 Amazon S3 伺服器存取日誌來識別請求 和 使用識別 Amazon S3 請求 CloudTrail。

檢閱和更新使用 ACL 相關條件索引鍵的儲存貯體政策

套用「儲存貯體擁有者強制執行」設定以停用 ACL 之後，只有在請求使用儲存貯體擁有者完全控制 ACL 或未指定 ACL 時，才能將新物件上傳至儲存貯體。在停用 ACL 之前，請檢閱儲存貯體政策以取得與 ACL 相關的條件索引鍵。

如果您的儲存貯體政策使用與 ACL 相關的條件索引鍵來請求 `bucket-owner-full-control` 標準 ACL (例如 `s3:x-amz-acl`)，則您不需要更新儲存貯體政策。下列儲存貯體政策會使用 `s3:x-amz-acl` 以要求適用於 S3 `PutObject` 請求的 `bucket-owner-full-control` 標準 ACL。本政策仍然需要物件寫入者指定 `bucket-owner-full-control` 標準 ACL。不過，停用 ACL 的儲存貯體仍會接受此 ACL，因此請求會繼續成功，不需要用戶端進行變更。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Only allow writes to my bucket with bucket owner full control",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/ExampleUser"
                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

不過，如果儲存貯體政策使用需要不同 ACL 的 ACL 相關條件索引鍵，則必須移除此條件索引鍵。此範例儲存貯體政策存需要適用於 S3 PutObject 請求的 public-read ACL，因此必須先更新才能停用 ACL。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Only allow writes to my bucket with public read access",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/ExampleUser"                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "public-read"
                }
            }
        }
```

```
    ]
}
```

「物件擁有權」許可

若要套用、更新或刪除儲存貯體的「物件擁有權」設定，您需要 `s3:PutBucketOwnershipControls` 許可。若要退回儲存貯體的「物件擁有權」設定，您需要 `s3:GetBucketOwnershipControls` 許可。如需更多詳細資訊，請參閱 在建立儲存貯體時設定「物件擁有權」 及 檢視 S3 儲存貯體的「物件擁有權」設定。

停用所有新儲存貯體的 ACL

根據預設，所有新的儲存貯體是在套用儲存貯體擁有者強制執行設定，並停用 ACL 的情況下建立。建議將 ACL 保持停用狀態。一般而言，我們建議使用 S3 以資源為基礎的政策 (儲存貯體政策與存取點原則) 或 IAM 政策來獲取存取控制而非 ACL。政策是一種簡化且更具彈性的存取控制選項。使用儲存貯體政策和存取點政策，您可以定義若干規則，廣泛應用於 Amazon S3 資源的所有請求。

複寫與物件所有權

當您使用 S3 複寫且來源和目的地儲存貯體屬於不同時 AWS 帳戶，您可以停用 ACL (使用強制執行物件擁有權的儲存貯體擁有者設定)，將複本擁有權變更為擁有目的地儲存貯體 AWS 帳戶 的複本擁有權。此設定會模擬現有的擁有者覆寫行為，而無需 `s3:ObjectOwnerOverrideToBucketOwner` 許可。使用「儲存貯體擁有者強制執行」設定複寫至目的地儲存貯體的所有物件都由目的地儲存貯體擁有者所擁有。如需複寫組態之擁有者覆寫選項的詳細資訊，請參閱 變更複本擁有者。

設定「物件擁有權」

您可以使用 Amazon S3 主控台、 AWS CLI AWS 開發套件、Amazon S3 REST API 或 AWS CloudFormation。下列 REST API 和 AWS CLI 命令支援物件擁有權：

REST API	AWS CLI	描述
PutBucketOwnershipControls	put-bucket-ownership-controls	建立或修改現有 S3 儲存貯體的「物件擁有權」設定。
CreateBucket	create-bucket	使用 `x-amz-object-ownership` 請求標頭建立儲存貯體，以指定「物件所有權」設定。

REST API	AWS CLI	描述
GetBucketOwnershipControls	get-bucket-ownership-controls	抓取 Amazon S3 儲存貯體的「物件擁有權」設定。
DeleteBucketOwnershipControls	delete-bucket-ownership-controls	刪除 Amazon S3 儲存貯體的「物件擁有權」設定。

如需有關套用和使用「物件擁有權」設定的更多資訊，請參閱以下主題。

主題

- 停用 ACL 的先決條件
- 在建立儲存貯體時設定「物件擁有權」
- 設定現有儲存貯體的「物件擁有權」
- 檢視 S3 儲存貯體的「物件擁有權」設定
- 停用所有新儲存貯體的 ACL 並強制執行「物件擁有權」
- 故障診斷

停用 ACL 的先決條件

如果您的值區 ACL 授予您以外的存取權 AWS 帳戶，則在停用 ACL 之前，您必須將值區 ACL 權限移轉至值區政策，並將值區 ACL 重設為預設的私有 ACL。如果您未遷移這些儲存貯體 ACL，則套用「儲存貯體擁有者強制執行」設定以停用 ACL 的請求會失敗，並傳回 InvalidBucketAclWithObjectOwnership 錯誤代碼。也建議您檢閱物件 ACL 許可，並將其遷移至儲存貯體政策。如需有關建議先決條件的詳細資訊，請參閱 停用 ACL 的先決條件。

每個現有的儲存貯體和物件 ACL 在 IAM 政策中都有對應項。下列儲存貯體政策範例說明儲存貯體和物件 ACL 的 `READ` 和 `WRITE` 許可如何映射至 IAM 許可。如需每個 ACL 如何轉譯成 IAM 許可的詳細資訊，請參閱 ACL 許可與存取政策許可的對應。

若要檢閱 ACL 許可並將 ACL 許可遷移至儲存貯體政策，請參閱下列主題。

主題

- 儲存貯體政策範例
- 使用 S3 主控台來檢閱及遷移 ACL 許可
- 使用檢 AWS CLI 閱及移轉 ACL 權限

- 範例演練

儲存貯體政策範例

這些範例儲存貯體政策說明如何將 READ 和 WRITE 儲存貯體及第三方 AWS 帳戶 的物件 ACL 許可遷移至儲存貯體政策。READ_ACP 和 WRITE_ACP ACL 與政策較不相關，因為其會授予與 ACL 相關的許可 (s3:GetBucketAcl、s3:GetObjectAcl、s3:PutBucketAcl 和 s3:PutObjectAcl)。

Example — 儲存貯體的 **READ** ACL

如果您的值區擁有授與列出值區內容之 AWS 帳戶 *111122223333*權限的 READ ACL，您可以撰寫值區政策授s3:ListBucket與值區的s3:ListBucketMultipartUploads權限。s3:ListBucketVersions

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Permission to list the objects in a bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": [

                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:ListBucket",
                "s3:ListBucketVersions",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
        }
    ]
}
```

Example – **READ** 儲存貯體中每個物件的 ACL

如果值區中的每個物件都有授與存取權的 READ ACL AWS 帳戶 *111122223333*，您可以撰寫值區政策，為值區中的每個物件授s3:GetObject與此帳戶並授與s3:GetObjectVersion權限。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Read permission for every object in a bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

此範例資源元素會授予特定物件的存取權。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/OBJECT-KEY"
```

Example –**WRITE** ACL，授予將物件寫入儲存貯體的許可

如果您的值區具有授與將物件寫入值區之 AWS 帳戶 *111122223333*權限的 WRITE ACL，您可以撰寫值區政策來授予值區s3:PutObject權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Permission to write objects to a bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
```

```
        }
    ]
}
```

使用 S3 主控台來檢閱及遷移 ACL 許可

檢閱儲存貯體的 ACL 許可

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇您的儲存貯體名稱。
3. 選擇許可索引標籤標籤。
4. 在 Access control list (存取控制清單 (ACL)) 下，請檢閱您的儲存貯體 ACL 許可。

檢閱物件的 ACL 許可

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇包含您物件的儲存貯體名稱。
3. 在 Object (物件) 清單中，選擇您的物件名稱。
4. 選擇許可索引標籤標籤。
5. 在 Access control list (存取控制清單 (ACL)) 下，請檢閱您的物件 ACL 許可。

遷移 ACL 許可並更新您的儲存貯體 ACL

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇您的儲存貯體名稱。
3. 在 Permissions (許可) 索引標籤上，Bucket policy (儲存貯體政策) 下，選擇 Edit (編輯)。
4. 在 Policy (政策) 方塊中，新增或更新儲存貯體政策。

   如需儲存貯體政策，請參閱 儲存貯體政策範例 和 範例演練。
5. 選擇儲存變更。
6. 更新您的儲存貯體 ACL 以移除對其他群組或 AWS 帳戶的 ACL 授予。
7. 針對物件擁有權套用儲存貯體擁有者強制執行設定。

使用檢 AWS CLI 閱及移轉 ACL 權限

1. 若要傳回值區的值區 ACL，請使用以下get-bucket-acl AWS CLI 指令：

```
aws s3api get-bucket-acl --bucket DOC-EXAMPLE-BUCKET
```

例如，此儲存貯體 ACL 會向第三方帳戶授予 WRITE 和 READ 存取權。在此 ACL 中，第三方帳戶是由正式使用者 ID 識別。若要套用「儲存貯體擁有者強制執行」設定並停用 ACL，您必須將第三方帳戶的這些許可遷移至儲存貯體政策。

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "DisplayName": "THIRD-PARTY-EXAMPLE-ACCOUNT",
                "ID":
 "72806de9d1ae8b171cca9e2494a8d1335dfced4ThirdPartyAccountCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "READ"
        },
        {
            "Grantee": {
                "DisplayName": "THIRD-PARTY-EXAMPLE-ACCOUNT",
                "ID":
 "72806de9d1ae8b171cca9e2494a8d1335dfced4ThirdPartyAccountCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "WRITE"
```

```
        }
    ]
}
```

如需其他範例 ACL，請參閱 範例演練。

2. 將您的儲存貯體 ACL 許可遷移至儲存貯體政策：

此範例儲存貯體政策會為第三方帳戶授予 s3:PutObject 和 s3:ListBucket 許可。在儲存貯體策略中，第三方帳號由 AWS 帳戶 ID (*111122223333*) 識別。

```
aws s3api put-bucket-policy --bucket DOC-EXAMPLE-BUCKET --policy file://policy.json

policy.json:
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PolicyForCrossAccountAllowUpload",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": [
             "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
             "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

如需更多儲存貯體政策範例，請參閱 儲存貯體政策範例 和 範例演練。

3. 若要傳回特定物件的 ACL，請使用get-object-acl AWS CLI 指令。

```
aws s3api get-object-acl --bucket DOC-EXAMPLE-BUCKET --key EXAMPLE-OBJECT-KEY
```

4. 如有必要，請將物件 ACL 許可遷移至儲存貯體政策。

此範例資源元素會授予儲存貯體政策中特定物件的存取權。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/EXAMPLE-OBJECT-KEY"
```

5. 將儲存貯體的 ACL 重設為預設 ACL。

```
aws s3api put-bucket-acl --bucket DOC-EXAMPLE-BUCKET --acl private
```

6. 針對物件擁有權套用「儲存貯體擁有者強制執行」設定。

範例演練

下列範例說明如何針對特定使用案例，將 ACL 許可遷移至儲存貯體政策。

主題

- 授予 S3 日誌交付群組的存取權以進行伺服器存取日誌記錄
- 對儲存貯體中的物件授予公用讀取存取權。
- 授予 Amazon ElastiCache 對您 S3 存儲桶的 Redis 訪問權限

授予 S3 日誌交付群組的存取權以進行伺服器存取日誌記錄

如果想要套用「儲存貯體擁有者強制執行」設定以針對伺服器存取記錄目的地儲存貯體 (也稱為目標儲存貯體) 停用 ACL，則您必須將 S3 日誌交付群組的儲存貯體 ACL 許可遷移至儲存貯體政策中的記錄服務主體 (logging.s3.amazonaws.com)。如需有關日誌交付許可的詳細資訊，請參閱 日誌交付許可。

此儲存貯體 ACL 授予 S3 日誌交付群組的 WRITE 和 READ_ACP 存取權：

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "Type": "CanonicalUser",
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
```

```
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery"
            },
            "Permission": "WRITE"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery"
            },
            "Permission": "READ_ACP"
        }
    ]
}
```

將 S3 日誌交付群組的儲存貯體 ACL 許可遷移至儲存貯體政策中的日誌記錄服務主體

1. 將以下儲存貯體政策新增至您的目的地儲存貯體，取代範例值。

```
aws s3api put-bucket-policy --bucket DOC-EXAMPLE-BUCKET --policy file://policy.json

policy.json:      {
    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/EXAMPLE-LOGGING-PREFIX*",
            "Condition": {
```

```
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::SOURCE-BUCKET-NAME"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

2. 將目的地儲存貯體的 ACL 重設為預設 ACL。

```
aws s3api put-bucket-acl --bucket DOC-EXAMPLE-BUCKET --acl private
```

3. 針對物件擁有權套用「儲存貯體擁有者強制執行」設定至您的目的地儲存貯體。

對儲存貯體中的物件授予公用讀取存取權。

如果您的物件 ACL 授予儲存貯體中所有物件的公用讀取存取權，則您可以將這些 ACL 許可遷移至儲存貯體政策。

此物件 ACL 會授予儲存貯體中物件的公用讀取存取權：

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
```

```
            },
            "Permission": "READ"
        }
    ]
}
```

將公用讀取 ACL 許可遷移至儲存貯體政策

1. 若要授予儲存貯體中所有物件的公用讀取存取權，請新增下列儲存貯體政策，取代範例值。

```
aws s3api put-bucket-policy --bucket DOC-EXAMPLE-BUCKET --policy file://policy.json

policy.json:
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        }
    ]
}
```

若要授予儲存貯體政策中特定物件的公用存取權，請使用 Resource 元素的下列格式。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/OBJECT-KEY"
```

若要授予具有特定字首之所有物件的公用存取權，請使用 Resource 元素的下列格式。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/PREFIX/*"
```

2. 針對物件擁有權套用「儲存貯體擁有者強制執行」設定。

授予 Amazon ElastiCache 對您 S3 存儲桶的 Redis 訪問權限

您可以將ElastiCache 適用於 Redis 的備份匯出到 S3 儲存貯體，讓您可以從外部 ElastiCache存取備份。若要將備份匯出到 S3 儲存貯體，您必須授 ElastiCache予將快照複製到儲存貯體的許可。如果您已授與值區 ACL ElastiCache 中的權限，您必須先將這些權限移轉至值區政策，才能套用值區擁有者強制執行設定來停用 ACL。如需詳細資訊，請參閱 Amazon ElastiCache 使用者指南中的授予對 Amazon S3 儲 ElastiCache 存貯體的存取權限。

下列範例顯示授與權限的儲存貯體 ACL 權限 ElastiCache。

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "READ"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "WRITE"
        },
```

```
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "READ_ACP"
        }
    ]
}
```

若要將 Redis 的 ElastiCache 值區 ACL 權限移轉至值區原則

1. 將以下儲存貯體政策新增至您的儲存貯體，替換範例值。

```
aws s3api put-bucket-policy --bucket DOC-EXAMPLE-BUCKET --policy file://policy.json

policy.json:
"Id": "Policy15397346",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "Region.elasticache-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        }
    ]
}
```

2. 將儲存貯體的 ACL 重設為預設 ACL：

```
aws s3api put-bucket-acl --bucket DOC-EXAMPLE-BUCKET --acl private
```

3. 針對物件擁有權套用「儲存貯體擁有者強制執行」設定。

在建立儲存貯體時設定「物件擁有權」

建立儲存貯體時，您可以設定 S3 物件所有權。若要設定現有儲存貯體的「物件擁有權」，請參閱 設定現有儲存貯體的「物件擁有權」。

S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，可讓您停用存取控制清單 (ACL) 並取得儲存貯體中每個物件的擁有權，簡化存放在 Amazon S3 中資料的存取管理。根據預設，新儲存貯體的 S3 物件擁有權會設定為儲存貯體擁有者強制執行設定，而且會停用 ACL。停用 ACL 後，儲存貯體擁有者會擁有儲存貯體中的每個物件，並使用存取管理政策專門管理對資料的存取。建議您將 ACL 保時停用狀態，除非在異常情況下必須個別控制每個物件的存取。

「物件擁有權」有三項設定，可讓您用來控制對上傳至儲存貯體之物件的擁有權，以及停用或啟用 ACL：

已停用 ACL

- 儲存貯體擁有者強制執行 (預設) - 停用 ACL，儲存貯體擁有者會自動擁有並完全控制儲存貯體中的每個物件。ACL 不再影響 S3 儲存貯體中資料的許可。儲存貯體使用政策來定義存取控制。

已啟用 ACL

- 儲存貯體擁有者偏好 – 儲存貯體擁有者擁有並完全控制其他帳戶使用 `bucket-owner-full-control` 標準 ACL 寫入儲存貯體的新物件。
- 物件寫入器 — 上傳物件的物件擁有物件、擁有物件的完全控制權，並且可以透過 ACL 授與其他使用者存取 AWS 帳戶 該物件。

許可：若要套用 Bucket owner enforced (儲存貯體擁有者強制執行) 設定或 Bucket owner preferred (儲存貯體擁有者偏好) 設定，您必須擁有以下許可：`s3:CreateBucket` 和 `s3:PutBucketOwnershipControls`。在套用 Object writer (物件寫入器) 設定的情況下建立儲存貯體時，不需要其他許可。如需 Amazon S3 許可的詳細資訊，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

> **Important**
>
> Amazon S3 中的大多數現代使用案例不再需要使用 ACL，建議您停用 ACL，除非在異常情況下需要個別控制每個物件的存取權。使用「物件擁有權」，您可以停用 ACL，並依賴政策來進行存取控制。停用 ACL 時，您可以輕鬆維護值區，其中包含由不同 AWS 帳戶上傳的物件。身為儲存貯體擁有者，您擁有儲存貯體中的所有物件，並且可以使用政策管理對物件的存取。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在頁面頂端的導覽列中，選擇目前顯示的名稱 AWS 區域。接下來，選擇您要在其中建立值區的「區域」。

   > **Note**
   >
   > 請選擇接近您的區域，以充分降低延遲及成本，並因應法規要求。除非您明確地將存放在區域中的物件傳輸到其他區域，否則物件絕對不會離開該區域。如需 Amazon S3 的清單 AWS 區域，AWS 服務 請參閱 Amazon Web Services 一般參考.

3. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
4. 選擇 Create bucket (建立儲存貯體)。

   Create bucket (建立儲存貯體) 頁面隨即開啟。
5. 在 [一般設定] 下方，檢視 AWS 區域 儲存貯體的建立位置。
6. 在「鏟斗類型」下選擇「一般用途」。
7. 針對 Bucket name (儲存貯體名稱)，輸入儲存貯體的名稱。

   儲存貯體名稱必須；

   - 在分割區內不重複。分割區是區域的群組。 AWS 目前有三個分割區：aws(標準區域)、aws-cn(中國區域) 和 aws-us-gov (AWS GovCloud (US) Regions)。
   - 長度必須介於 3 與 63 個字元之間。
   - 只能由小寫字母、數字、句點 (.) 和連字號 (-) 組成。為了獲得最佳相容性，建議您避免在儲存貯體名稱中使用句點 (.)，但僅用於靜態網站託管的儲存貯體除外。
   - 開頭和結尾為字母或數字。

建立儲存貯體後，便無法變更其名稱。如需儲存貯體命名的詳細資訊，請參閱 [儲存貯體命名規則]。

> **Important**
>
> 避免在儲存貯體名稱中包含敏感資訊，例如帳戶號碼。在指向儲存貯體中之物件的 URL 中，會顯示儲存貯體名稱。

8. AWS Management Console 可讓您將現有值區的設定複製到新值區。如果您不想複製現有值區的設定，請跳至下一個步驟。

> **Note**
>
> 此選項：
>
> - 無法在中使用， AWS CLI 且只能在主控台中使用
> - 不適用於目錄值區
> - 不會將值區政策從現有值區複製到新值區

若要複製現有值區的設定，請在 [從現有值區複製設定] 底下，選取 [選擇值區]。「選擇時段」視窗即會開啟。找出包含您要複製之設定的值區，然後選取 [選擇值區]。「選擇時段」視窗即會關閉，且「建立時段」視窗會重新開啟。

在「從現有值區複製設定」 下方，您現在會看到所選值區的名稱。您也會看到 [還原預設值] 選項，可用來移除複製的值區設定。檢閱「建立值區」頁面上的剩餘值區設定。您將看到它們現在與您選擇的存儲桶的設置匹配。您可以跳到最後一步。

9. 在 Object Ownership (物件擁有權) 下，若要停用或啟用 ACL 並控制上傳在儲存貯體中物件的擁有權，請選擇下列其中一個設定：

已停用 ACL

- 儲存貯體擁有者強制執行 (預設) - 停用 ACL，儲存貯體擁有者會自動擁有並完全控制儲存貯體中的每個物件。ACL 不再影響 S3 儲存貯體中資料的存取許可。儲存貯體單獨使用政策來定義存取控制。

根據預設，會停用 ACL。Amazon S3 中的大多數新式使用案例不再需要使用 ACL。建議您將 ACL 保時停用狀態，除非在異常情況下必須個別控制每個物件的存取。如需詳細資訊，請參閱控制物件的擁有權並停用儲存貯體的 ACL。

已啟用 ACL

- 儲存貯體擁有者偏好 – 儲存貯體擁有者擁有並完全控制其他帳戶使用 `bucket-owner-full-control` 標準 ACL 寫入儲存貯體的新物件。

  如果您套用儲存貯體擁有者偏好設定，以要求所有 Amazon S3 上傳都包含 `bucket-owner-full-control` 固定 ACL 時，您可以新增儲存貯體政策，只允許使用此 ACL 的物件上傳。
- 物件寫入器 — 上傳物件的物件擁有物件、擁有物件的完全控制權，並且可以透過 ACL 授與其他使用者存取 AWS 帳戶 該物件。

> **Note**
>
> 預設設定為儲存貯體擁有者強制執行。若要套用預設設定並將 ACL 保持停用狀態，只需要 `s3:CreateBucket` 許可。若要啟用 ACL，您必須具有 `s3:PutBucketOwnershipControls` 許可。

10. 在封鎖此儲存貯體的公開存取設定之下，選擇要套用至儲存貯體的封鎖公開存取設定。

    根據預設，會啟用全部四個「封鎖公開存取」設定。建議您將所有設定保持啟用狀態，除非您知道需要針對特定使用案例關閉其中一或多個設定。如需封鎖公開存取的詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。

    > **Note**
    >
    > 若要啟用所有「封鎖公用存取」設定，只需要 `s3:CreateBucket` 許可。若要關閉任何「封鎖公開存取」設定，您必須具有 `s3:PutBucketPublicAccessBlock` 許可。

11. (選用) 在 Bucket Versioning (儲存貯體版本控制) 下，您可以選擇是否要在儲存貯體中保留物件的變體。如需版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。

    若要對儲存貯體停用或啟用版本控制，請選擇 Disable (停用) 或 Enable (啟用)。

12. (選用) 在 Tags (標籤) 下，您可以選擇新增標籤至儲存貯體。標籤是用來分類儲存的鍵值對。

若要新增儲存貯體標籤，請輸入 Key (金鑰) 並選擇性地輸入 Value (值)，然後選擇 Add tag (新增標籤)。

13. 在 Default encryption (預設加密) 底下，選擇 Edit (編輯)。
14. 若要設定預設加密，請在加密類型下，選擇下列其中一項：

    - Amazon S3 受管金鑰 (SSE-S3)
    - AWS Key Management Service 金鑰 (SSE-公里)

    > **Important**
    >
    > 如果您針對預設加密組態使用 SSE-KMS 選項，則受到 AWS KMS的每秒請求數目 (RPS) 限制。如需有關 AWS KMS 配額以及如何要求提高配額的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的配額。

    儲存貯體和新物件會以 Amazon S3 受管金鑰做為基本加密組態層級，使用伺服器端加密。如需預設加密的詳細資訊，請參閱對 Amazon S3 儲存貯體設定預設伺服器端加密行為。

    如需有關使用 Amazon S3 伺服器端加密來加密資料的詳細資訊，請參閱「使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密」。
15. 若您選擇 AWS Key Management Service 金鑰 (SSE-KMS)，請執行下列操作：

    a. 在 AWS KMS 金鑰下，使用下列其中一種方式指定 KMS 金鑰：

    - 若要從可用 KMS 金鑰清單中選擇，請選擇從您的金鑰中選擇 AWS KMS keys，然後從可用金鑰清單中選擇您的 KMS 金鑰。

      AWS 受管金鑰 (aws/s3) 和您的客戶管理金鑰都會出現在此清單中。如需詳細了解客戶受管金鑰，請參閱《AWS Key Management Service 開發人員指南》中的客戶金鑰和 AWS 金鑰。
    - 若要輸入 KMS 金鑰 ARN，請選擇輸入 AWS KMS key ARN，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。
    - 若要在 AWS KMS 主控台中建立新的客戶管理金鑰，請選擇 [建立 KMS 金鑰]。

      如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。

> **Important**
>
> 您只能使用與值區相同 AWS 區域 的 KMS 金鑰。Amazon S3 主控台僅會列出與儲存貯體位於相同區域的前 100 個 KMS 金鑰。若要使用未列出的 KMS 金鑰，必須輸入 KMS 金鑰 ARN。若您想要使用其他帳戶的 KMS 金鑰，您必須先具有該金鑰的使用權限，然後輸入 KMS 金鑰 ARN。如需詳細了解 KMS 金鑰跨帳戶權限，請參閱《AWS Key Management Service 開發人員指南》中的建立其他帳戶可使用的 KMS 金鑰。如需 SSE-KMS 的詳細資訊，請參閱 使用 AWS KMS (SSE-KMS) 指定伺服器端加密。
>
> 當您在 Amazon S3 中使 AWS KMS key 用伺服器端加密時，您必須選擇對稱加密 KMS 金鑰。Amazon S3 僅支援對稱加密 KMS 金鑰，而不支援非對稱 KMS 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的識別對稱和非對稱 KMS 金鑰。

如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。如需 AWS KMS 搭配 Amazon S3 搭配使用的詳細資訊，請參閱使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)。

b. 當您將儲存貯體設定為使用 SSE-KMS 的預設加密時，您還可以啟用 S3 儲存貯體金鑰。S3 儲存貯體金鑰透過將 Amazon S3 的請求流量減少到，從而降低加密成本 AWS KMS。如需詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本。

   若要使用 S3 儲存貯體金鑰，在 Bucket Key (儲存貯體金鑰) 下選擇 Enable (啟用)。

16. (選用) 如果您想要啟用 S3 物件鎖定，請執行下列動作：

    a. 選擇 Advanced settings (進階設定)。

    > **Important**
    >
    > 啟用物件鎖定也會啟用儲存貯體的版本控制。啟用後，您必須設定「物件鎖定」預設保留和合法保留設定，以防止新物件遭到刪除或覆寫。

    b. 如果想要啟用物件鎖定，請選擇 Enable (啟用)、讀取出現的警告並確認。

    如需詳細資訊，請參閱 使用 S3 物件鎖定。

> **Note**
>
> 若要建立已啟用物件鎖定的儲存貯體，您必須具備下列許可：s3:CreateBucket、s3:PutBucketVersioning 和 s3:PutBucketObjectLockConfiguration。

17. 選擇建立儲存貯體。

使用 AWS CLI

若要在建立新值區時設定物件擁有權，請使用create-bucket AWS CLI 指令搭配--object-ownership參數。

此範例會使用 AWS CLI為新的儲存貯體套用「儲存貯體擁有者強制執行」設定：

```
aws s3api create-bucket --bucket  DOC-EXAMPLE-BUCKET --region us-east-1 --object-ownership BucketOwnerEnforced
```

> **Important**
>
> 如果您在使用建立值區時未設定物件擁有權 AWS CLI，則預設設定為 ObjectWriter (啟用 ACL)。

使用適用於 Java 的 AWS 開發套件

此範例會使用 AWS SDK for Java為新的儲存貯體設定「儲存貯體擁有者強制執行」設定：

```
    // Build the ObjectOwnership for CreateBucket
    CreateBucketRequest createBucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectOwnership(ObjectOwnership.BucketOwnerEnforced)
            .build()

     // Send the request to Amazon S3
     s3client.createBucket(createBucketRequest);
```

使用 AWS CloudFormation

若要在建立新值區時使用`AWS::S3::Bucket` AWS CloudFormation 資源來設定物件擁有權，請參閱《使AWS CloudFormation 用指南》OwnershipControlsAWS::S3::Bucket中的〈〉。

使用 REST API

若要套用 S3 物件擁有權的「儲存貯體擁有者強制執行」設定，請使用 `CreateBucket` API 操作 (`x-amz-object-ownership` 請求標頭設定為 `BucketOwnerEnforced`)。如需資訊和範例，請參閱《Amazon Simple Storage Service API 參考》中的 CreateBucket。

後續步驟：套用「物件擁有權」的儲存貯體擁有者強制執行設定或儲存貯體擁有者偏好設定之後，您可以進一步採取下列步驟：

- 儲存貯體擁有者強制執行 – 需要使用 IAM 或 Organizations 政策，在停用 ACL 的情況下建立所有新儲存貯體。
- 儲存貯體擁有者偏好 – 新增 S3 儲存貯體政策，以要求所有物件上傳到您儲存貯體的 `bucket-owner-full-control` 標準 ACL。

設定現有儲存貯體的「物件擁有權」

您可以在現有 S3 儲存貯體上設定 S3 物件擁有權。若要在建立儲存貯體時套用「物件擁有權」，請參閱 在建立儲存貯體時設定「物件擁有權」。

S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，可讓您停用存取控制清單 (ACL) 並取得儲存貯體中每個物件的擁有權，簡化存放在 Amazon S3 中資料的存取管理。根據預設，新儲存貯體的 S3 物件擁有權會設定為儲存貯體擁有者強制執行設定，而且會停用 ACL。停用 ACL 後，儲存貯體擁有者會擁有儲存貯體中的每個物件，並使用存取管理政策專門管理對資料的存取。建議您將 ACL 保時停用狀態，除非在異常情況下必須個別控制每個物件的存取。

「物件擁有權」有三項設定，可讓您用來控制對上傳至儲存貯體之物件的擁有權，以及停用或啟用 ACL：

已停用 ACL

- 儲存貯體擁有者強制執行 (預設) - 停用 ACL，儲存貯體擁有者會自動擁有並完全控制儲存貯體中的每個物件。ACL 不再影響 S3 儲存貯體中資料的許可。儲存貯體使用政策來定義存取控制。

已啟用 ACL

- 儲存貯體擁有者偏好 – 儲存貯體擁有者擁有並完全控制其他帳戶使用 `bucket-owner-full-control` 標準 ACL 寫入儲存貯體的新物件。
- 物件寫入器 — 上傳物件的物件擁有物件、擁有物件的完全控制權，並且可以透過 ACL 授與其他使用者存取 AWS 帳戶 該物件。

先決條件：在套用「儲存貯體擁有者強制執行」設定以停用 ACL 之前，您必須將儲存貯體 ACL 許可遷移至儲存貯體政策，並將儲存貯體 ACL 重設為預設的私有 ACL。也建議您將物件 ACL 許可遷移至儲存貯體政策，並編輯需要 ACL (儲存貯體擁有者完全控制 ACL 除外) 的儲存貯體政策。如需詳細資訊，請參閱 停用 ACL 的先決條件。

許可：若要使用這項操作，您必須擁有 `s3:PutBucketOwnershipControls` 許可。如需 Amazon S3 許可的詳細資訊，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇您要將 S3 「物件擁有權」設定套用至其中的儲存貯體名稱。
3. 選擇許可索引標籤標籤。
4. 在 Object Ownership (物件擁有權) 下，選擇 Edit (編輯)。
5. 在 Object Ownership (物件擁有權) 下，若要停用或啟用 ACL 並控制上傳在儲存貯體中物件的擁有權，請選擇下列其中一個設定：

   已停用 ACL

   - Bucket owner enforced (儲存貯體擁有者強制執行) – 停用 ACL，儲存貯體擁有者會自動擁有並完全控制儲存貯體中的每個物件。ACL 不再影響 S3 儲存貯體中資料的許可。儲存貯體使用政策來定義存取控制。

     若要求在使用 IAM 或 AWS Organizations 政策停用 ACL 的情況下建立所有新值區，請參閱停用所有新儲存貯體的 ACL (儲存貯體擁有者強制執行)。

   已啟用 ACL

   - 儲存貯體擁有者偏好 – 儲存貯體擁有者擁有並完全控制其他帳戶使用 `bucket-owner-full-control` 標準 ACL 寫入儲存貯體的新物件。

如果您套用儲存貯體擁有者偏好設定，則要求所有 Amazon S3 上傳都包含 bucket-owner-full-control 罐裝 ACL 時，您可以新增儲存貯體政策，只允許使用此 ACL 的物件上傳。

- 物件寫入器 — 上傳物件的物件擁有物件、擁有物件的完全控制權，並且可以透過 ACL 授與其他使用者存取 AWS 帳戶 該物件。

6. 選擇儲存。

使用 AWS CLI

若要針對現有儲存貯體套用「物件擁有權」設定，請使用具有 --ownership-controls 參數的 put-bucket-ownership-controls 命令。擁有權的有效值為 BucketOwnerEnforced、BucketOwnerPreferred 或 ObjectWriter。

此範例會使用 AWS CLI為現有儲存貯體套用「儲存貯體擁有者強制執行」設定：

```
aws s3api put-bucket-ownership-controls --bucket DOC-EXAMPLE-BUCKET --ownership-
controls="Rules=[{ObjectOwnership=BucketOwnerEnforced}]"
```

如需 put-bucket-ownership-controls 的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的 put-bucket-ownership-controls。

使用適用於 Java 的 AWS 開發套件

此範例會使用 AWS SDK for Java為現有儲存貯體套用「物件擁有權」的 BucketOwnerEnforced 設定：

```
        // Build the ObjectOwnership for BucketOwnerEnforced
        OwnershipControlsRule rule = OwnershipControlsRule.builder()
                .objectOwnership(ObjectOwnership.BucketOwnerEnforced)
                .build();

        OwnershipControls ownershipControls = OwnershipControls.builder()
                  .rules(rule)
                  .build()

         // Build the PutBucketOwnershipControlsRequest
         PutBucketOwnershipControlsRequest putBucketOwnershipControlsRequest =
               PutBucketOwnershipControlsRequest.builder()
                       .bucket(BUCKET_NAME)
                       .ownershipControls(ownershipControls)
```

```
                        .build();

        // Send the request to Amazon S3
        s3client.putBucketOwnershipControls(putBucketOwnershipControlsRequest);
```

使用 AWS CloudFormation

若 AWS CloudFormation 要用於對現有值區套用「物件擁有權」設定，請參閱《使AWS CloudFormation 用指南》AWS::S3::Bucket OwnershipControls中的〈〉。

使用 REST API

若要使用 REST API 將「物件擁有權」設定套用至現有的 S3 儲存貯體，請使用 `PutBucketOwnershipControls`。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 PutBucketOwnershipControls。

後續步驟：套用「物件擁有權」的儲存貯體擁有者強制執行設定或儲存貯體擁有者偏好設定之後，您可以進一步採取下列步驟：

- 儲存貯體擁有者強制執行 – 需要使用 IAM 或 Organizations 政策，在停用 ACL 的情況下建立所有新儲存貯體。
- 儲存貯體擁有者偏好 – 新增 S3 儲存貯體政策，以要求所有物件上傳到您儲存貯體的 `bucket-owner-full-control` 標準 ACL。

檢視 S3 儲存貯體的「物件擁有權」設定

S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，可讓您停用存取控制清單 (ACL) 並取得儲存貯體中每個物件的擁有權，簡化存放在 Amazon S3 中資料的存取管理。根據預設，新儲存貯體的 S3 物件擁有權會設定為儲存貯體擁有者強制執行設定，而且會停用 ACL。停用 ACL 後，儲存貯體擁有者會擁有儲存貯體中的每個物件，並使用存取管理政策專門管理對資料的存取。建議您將 ACL 保時停用狀態，除非在異常情況下必須個別控制每個物件的存取。

「物件擁有權」有三項設定，可讓您用來控制對上傳至儲存貯體之物件的擁有權，以及停用或啟用 ACL：

已停用 ACL

- 儲存貯體擁有者強制執行 (預設) - 停用 ACL，儲存貯體擁有者會自動擁有並完全控制儲存貯體中的每個物件。ACL 不再影響 S3 儲存貯體中資料的許可。儲存貯體使用政策來定義存取控制。

已啟用 ACL

- 儲存貯體擁有者偏好 – 儲存貯體擁有者擁有並完全控制其他帳戶使用 `bucket-owner-full-control` 標準 ACL 寫入儲存貯體的新物件。
- 物件寫入器 — 上傳物件的物件擁有物件、擁有物件的完全控制權，並且可以透過 ACL 授與其他使用者存取 AWS 帳戶 該物件。

您可以檢視 Amazon S3 儲存貯體的 S3 物件擁有權設定。若要設定新儲存貯體的「物件擁有權」，請參閱 在建立儲存貯體時設定「物件擁有權」。若要設定現有儲存貯體的「物件擁有權」，請參閱 設定現有儲存貯體的「物件擁有權」。

許可：若要使用這項操作，您必須擁有 `s3:GetBucketOwnershipControls` 許可。如需 Amazon S3 許可的詳細資訊，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇您要將「物件擁有權」設定套用至其中的儲存貯體名稱。
3. 選擇許可索引標籤標籤。
4. 在 Object Ownership (物件擁有權) 下，您可以檢視儲存貯體的「物件擁有權」設定。

使用 AWS CLI

若要擷取 S3 儲存貯體的 S3 物件擁有權設定，請使用get-bucket-ownership-controls AWS CLI 命令。

```
aws s3api get-bucket-ownership-controls --bucket DOC-EXAMPLE-BUCKET
```

使用 REST API

若要擷取 S3 儲存貯體的「物件擁有權」設定，請使用 `GetBucketOwnershipControls` API 操作。如需詳細資訊，請參閱 GetBucketOwnershipControls。

停用所有新儲存貯體的 ACL 並強制執行「物件擁有權」

建議您停用 Amazon S3 儲存貯體上的 ACL。您可以套用 S3 物件擁有權的「儲存貯體擁有者強制執行」設定來執行此操作。當您套用此設定時，會停用 ACL，而且您會自動擁有並完全控制儲存貯體中的所有物件。若要求在停用 ACL 的情況下建立所有新值區，請使用 AWS Identity and Access Management (IAM) 政策或 AWS Organizations 服務控制政策 (SCP)，如下一節所述。

若要在不停用 ACL 的情況下強制執行新物件的「物件擁有權」，您可以套用儲存貯體擁有者偏好設定。套用此設定時，強烈建議您更新儲存貯體政策以要求對您儲存貯體的所有 PUT 請求使用 `bucket-owner-full-control` 固定 ACL。用戶端也應該更新，以從其他帳戶傳送 `bucket-owner-full-control` 固定 ACL 到您的儲存貯體。

主題

- 停用所有新儲存貯體的 ACL (儲存貯體擁有者強制執行)
- Amazon S3 PUT 操作需要 bucket-owner-full-control固定 ACL (建議使用儲存貯體擁有者)

停用所有新儲存貯體的 ACL (儲存貯體擁有者強制執行)

下列範例 IAM 政策會拒絕特定 IAM 使用者或角色的 `s3:CreateBucket` 許可，除非為物件擁有權套用「儲存貯體擁有者強制執行」設定。`Condition` 區塊中的鍵值對會指定 `s3:x-amz-object-ownership` 作為其鍵，並指定 `BucketOwnerEnforced` 設定作為其值。換句話說，只有在 IAM 使用者為物件擁有權設定「儲存貯體擁有者強制執行」設定並停用 ACL 時，IAM 使用者才能建立儲存貯體。您也可以使用此原則做為 AWS 組織的界限 SCP。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireBucketOwnerFullControl",
            "Action": "s3:CreateBucket",
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-object-ownership": "BucketOwnerEnforced"
                }
            }
        }
    ]
}
```

Amazon S3 **PUT** 操作需要 bucket-owner-full-control固定 ACL (建議使用儲存貯體擁有者)

使用「物件擁有權」的儲存貯體擁有者偏好設定，身為儲存貯體擁有者，您可以擁有並完全控制其他帳戶寫入您具有 `bucket-owner-full-control` 固定 ACL 之儲存貯體的物件。但是，如果其他帳戶將物件寫入沒有 `bucket-owner-full-control` 標準 ACL 的儲存貯體時，物件寫入者會維持完整

的控制存取權。身為儲存貯體擁有者，您可以實作只有在其指定 bucket-owner-full-control 標準 ACL 時才允許寫入的儲存貯體政策。

> **Note**
>
> 如果使用「儲存貯體擁有者強制執行」設定停用了 ACL，則您身為儲存貯體擁有者會自動擁有並完全控制儲存貯體中的所有物件。您不需要使用此區段來更新儲存貯體政策，以強制執行儲存貯體擁有者的物件擁有權。

下列儲存貯體政策指定只有在物件的 ACL 設定為 111122223333 時，帳戶 *DOC-EXAMPLE-BUCKET* 才能將物件上傳至 *bucket-owner-full-control*。請務必使用您的帳戶取代 *111122223333*，以及使用您儲存貯體的名稱取代 *DOC-EXAMPLE-BUCKET*。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Only allow writes to my bucket with bucket owner full control",
         "Effect": "Allow",
         "Principal": {
            "AWS": [
               "arn:aws:iam::111122223333:user/ExampleUser"
            ]
         },
         "Action": [
            "s3:PutObject"
         ],
         "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
         "Condition": {
            "StringEquals": {
               "s3:x-amz-acl": "bucket-owner-full-control"
            }
         }
      }
   ]
}
```

以下是範例複製操作，其中包含使用 AWS Command Line Interface (AWS CLI) 的 bucket-owner-full-control 固定 ACL。

```
aws s3 cp file.txt s3://DOC-EXAMPLE-BUCKET --acl bucket-owner-full-control
```

儲存貯體政策生效後，如果用戶端不包含 bucket-owner-full-control 固定 ACL，則操作會失敗，且上傳程式會收到下列錯誤：

調用 PutObject 操作時發生錯誤（AccessDenied）：訪問被拒絕。

> **Note**
>
> 如果用戶端在上傳後需要存取物件，您必須授予上傳帳戶的額外許可。如需有關授予帳戶對資源的存取權的資訊，請參閱 使用政策管理 Amazon S3 資源存取權的逐步解說。

故障診斷

套用 S3 物件擁有權的儲存貯體擁有者強制執行設定時，會停用存取控制清單 (ACL)，並且您身為儲存貯體擁有者會自動擁有儲存貯體中的所有物件。ACL 不再影響儲存貯體中物件的許可。您可以使用政策來授予許可。所有 S3 PUT 請求都必須指定 bucket-owner-full-control 固定 ACL 或不指定 ACL，否則這些請求將會失敗。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

如果指定了無效的 ACL，或是儲存貯體 ACL 許可授予 AWS 帳戶以外的存取權，則您可能會見到下列錯誤回應。

AccessControlListNotSupported

在您套用物件擁有權的「儲存貯體擁有者強制執行」設定之後，ACL 會停用。設定 ACL 或更新 ACL 的要求失敗，並顯示400錯誤，並傳回AccessControlListNotSupported錯誤碼。仍支援讀取 ACL 的請求。讀取 ACL 的請求一律會傳回回應，顯示儲存貯體擁有者的完整控制權。在 PUT 操作中，您必須指定儲存貯體擁有者完全控制 ACL 或不指定 ACL。否則，您的 PUT 操作將失敗。

下列範例put-object AWS CLI 指令包括public-read固定 ACL。

```
aws s3api put-object --bucket DOC-EXAMPLE-BUCKET --key object-key-name --body doc-example-body --acl public-read
```

如果儲存貯體使用「儲存貯體擁有者強制執行」設定來停用 ACL，則此操作會失敗，且上傳工具會收到下列錯誤訊息：

呼叫PutObject 作業時發生錯誤 (AccessControlListNotSupported)：值區不允許 ACL

InvalidBucketAclWithObjectOwnership

如果您想要套用「儲存貯體擁有者強制執行」設定來停用 ACL，則儲存貯體 ACL 必須將完全控制權僅提供給儲存貯體擁有者。值區 ACL 無法授予外部 AWS 帳戶 或任何其他群組的存取權。例如，如果您的`CreateBucket`請求設定值區擁有者強制執行，並指定提供外部存取權的值區 ACL AWS 帳戶，則您的請求會失敗並顯示`400`錯誤，並傳回InvalidBucketAclWithObjectOwnership錯誤碼。同樣地，如果您的 `PutBucketOwnershipControls` 請求設定在擁有儲存貯體 ACL (對其他人授予許可) 之儲存貯體上強制執行的儲存貯體擁有者，則請求會失敗。

Example ：現有儲存貯體 ACL 會授予公用讀取存取權

例如，如果現有儲存貯體 ACL 授予公用讀取存取權，則在將這些 ACL 許可遷移至儲存貯體政策，並將儲存貯體 ACL 重設為預設的私有 ACL 之前，您無法套用「儲存貯體擁有者強制執行」設定。如需詳細資訊，請參閱 停用 ACL 的先決條件。

此範例儲存貯體 ACL 會授予公用讀取存取權：

```
{
    "Owner": {
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        }
    ]
}
```

下列範例put-bucket-ownership-controls AWS CLI 命令會針對物件擁有權套用值區擁有者強制執行設定：

```
aws s3api put-bucket-ownership-controls --bucket DOC-EXAMPLE-BUCKET --ownership-controls Rules=[{ObjectOwnership=BucketOwnerEnforced}]
```

因為儲存貯體 ACL 會授予公用讀取存取權，所以請求會失敗，並傳回下列錯誤碼：

調用 PutBucketOwnershipControls 操作時發生錯誤（InvalidBucketAclWithObjectOwnership）：存儲桶不能使用 ObjectOwnership的 BucketOwnerEnforced 設置設置 ACL

使用跨來源資源分享 (CORS)

跨來源資源分享 (CORS) 會定義一種方式，讓載入單一個網域的用戶端 Web 應用程式，能與不同網域中的資源互動。透過 CORS 支援，您可以使用 Amazon S3 建立功能豐富的用戶端 Web 應用程式，而且也可以選擇性地允許跨來源存取您的 Amazon S3 資源。

本節提供 CORS 的概觀。這些副主題說明如何使用 Amazon S3 主控台啟用 CORS，或透過使用 Amazon S3 REST API 和開發套件以程式設計方式啟用 CORS。 AWS

跨來源資源分享：使用案例情境

以下為使用 CORS 的情境範例。

案例 1

假設您正在名為 website 的 Amazon S3 儲存貯體中託管網站，如 使用 Amazon S3 託管靜態網站 所述。您的使用者載入了網站端點：

```
http://website.s3-website.us-east-1.amazonaws.com
```

現在，您想要在此儲存貯體中存放的網頁 JavaScript 上使用，以便能夠使用儲存貯體的 Amazon S3 API 端點，對相同儲存貯體發出經驗證的 GET 和 PUT 請求website.s3.us-east-1.amazonaws.com。瀏覽器通常會阻 JavaScript 止允許這些請求，但是使用 CORS，您可以將存儲桶配置為顯式啟用跨源請求。website.s3-website.us-east-1.amazonaws.com

案例 2

假設您希望從 S3 儲存貯體託管一個 Web 字型。再者，瀏覽器需要 CORS 檢查(也稱為預先檢查)，來載入 Web 字體。您可以設定裝載 Web 字體的儲存貯體，允許任何來源發出這些要求。

Amazon S3 如何評估儲存貯體上的 CORS 組態？

當 Amazon S3 從瀏覽器接收到預檢要求時，會評估儲存貯體的 CORS 組態，並使用第一個符合來源瀏覽器要求的 `CORSRule` 規則來啟用跨來源要求。符合規則必須滿足下列條件：

- 要求的 `Origin` 標頭必須符合 `AllowedOrigin` 元素。
- 要求的方法 (例如 GET 或 PUT) 或預檢 `Access-Control-Request-Method` 要求的 `OPTIONS` 標頭，必須是 `AllowedMethod` 元素中的其中之一。
- 預檢要求上要求的 `Access-Control-Request-Headers` 標頭中所列出之每一個標頭，都必須符合 `AllowedHeader` 元素。

> **Note**
>
> 在儲存貯體上啟用 CORS 時，仍繼續適用 ACL 及政策。

Object Lambda 存取點如何支援 CORS

當 S3 Object Lambda 從瀏覽器接收到請求，或請求包含 `Origin` 標頭時，S3 Object Lambda 一律會新增 `“AllowedOrigins":"*"` 標頭欄位。

如需使用 CORS 的詳細資訊，請參閱下列主題。

主題

- CORS 組態
- 設定跨來源資源分享 (CORS)

CORS 組態

若要設定儲存貯體允許跨來源要求，您可以建立 CORS 組態。CORS 組態是具有規則的文件，這些規則可以識別允許您存取您儲存貯體的來源、每個來源支援的操作 (HTTP 方法)，以及其他操作特定資訊。最多可以將 100 條規則新增至組態。您可以將 CORS 組態以 `cors` 子資源的形式新增至儲存貯體。

如果您在 S3 主控台中設定 CORS，則必須使用 JSON 建立 CORS 組態。新的 S3 主控台只支援 JSON CORS 組態。

如需有關 CORS 組態及其中元素的詳細資訊，請參閱下列主題。如需如何新增 CORS 組態的說明，請參閱 設定跨來源資源分享 (CORS)。

> **Important**
>
> 在 S3 主控台中，CORS 組態必須為 JSON 格式。

主題

範例 1

除了使用 Amazon S3 網站端點存取網站之外，您也可以使用自己的網域 (例如 example1.com) 提供內容。如需使用自有網域的資訊，請參閱「教學課程：使用向 Route 53 註冊的自訂網域設定靜態網站」。

下列範例 cors 組態中有三條規則，以 CORSRule 元素的方式指定：

- 第一項規則允許來自 http://www.example1.com 來源的跨來源 PUT、POST 及 DELETE 要求。該規則也允許透過 Access-Control-Request-Headers 標頭傳送之預檢 OPTIONS 要求中的所有標頭。為回應預檢 OPTIONS 要求，Amazon S3 會傳回所有要求的標頭。
- 第二項規則允許與第一項規則相同的跨來源要求，但該規則適用於另一個來源 (http://www.example2.com)。
- 第三項規則則允許來自所有來源的跨來源 GET 要求。* 萬用字元表示所有來源。

JSON

```
[
    {
```

```
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example1.com"
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example2.com"
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": []
    }
]
```

XML

```
<CORSConfiguration>
 <CORSRule>
```

```
    <AllowedOrigin>http://www.example1.com</AllowedOrigin>

    <AllowedMethod>PUT</AllowedMethod>
    <AllowedMethod>POST</AllowedMethod>
    <AllowedMethod>DELETE</AllowedMethod>

    <AllowedHeader>*</AllowedHeader>
  </CORSRule>
  <CORSRule>
    <AllowedOrigin>http://www.example2.com</AllowedOrigin>

    <AllowedMethod>PUT</AllowedMethod>
    <AllowedMethod>POST</AllowedMethod>
    <AllowedMethod>DELETE</AllowedMethod>

    <AllowedHeader>*</AllowedHeader>
  </CORSRule>
  <CORSRule>
    <AllowedOrigin>*</AllowedOrigin>
    <AllowedMethod>GET</AllowedMethod>
  </CORSRule>
</CORSConfiguration>
```

範例 2

CORS 組態也允許選用的組態參數，如下列 CORS 組態中所示。在此範例中，以下 CORS 組態會允許來自 `http://www.example.com` 來源的跨來源 PUT、POST 及 DELETE 要求。

JSON

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example.com"
```

```
        ],
        "ExposeHeaders": [
            "x-amz-server-side-encryption",
            "x-amz-request-id",
            "x-amz-id-2"
        ],
        "MaxAgeSeconds": 3000
    }
]
```

XML

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example.com</AllowedOrigin>
   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>
   <AllowedHeader>*</AllowedHeader>
  <MaxAgeSeconds>3000</MaxAgeSeconds>
  <ExposeHeader>x-amz-server-side-encryption</
ExposeHeader>
  <ExposeHeader>x-amz-request-id</
ExposeHeader>
  <ExposeHeader>x-amz-id-2</ExposeHeader>
 </CORSRule>
</CORSConfiguration>
```

先前組態中的 CORSRule 元素，包含下列選用元素：

- MaxAgeSeconds — 指定時間秒數 (在此範例中為 3000)，瀏覽器會為指定的資源對預檢 OPTIONS 要求快取 Amazon S3 回應的時間。藉由快取回應的方式，瀏覽器便不需要在原始要求重複時，將再次向 Amazon S3 傳送預檢要求。
- ExposeHeader識別客戶能夠從其應用程式存取的回應標頭 (例如 x-amz-server-side-encryptionx-amz-request-id，從 JavaScript XMLHttpRequest物件x-amz-id-2) 存取的回應標頭 (在此範例中為、和)。

AllowedMethod 元素

在 CORS 組態中，您可以為 AllowedMethod 元素指定下列數值。

- GET
- PUT
- POST
- DELETE
- HEAD

AllowedOrigin 元素

在 AllowedOrigin 元素中，可指定希望允許跨網域要求的來源，例如 http://www.example.com。原始字串只可包含一個 '*' 萬用字元，例如 http://*.example.com。您可以選擇性地指定 '*' 作為來源，允許所有來源傳送跨來源要求。您也可以指定 https 而只允許安全的來源。

AllowedHeader 元素

AllowedHeader 元素透過 Access-Control-Request-Headers 標頭，可指定在預檢要求中允許的標頭。在 Access-Control-Request-Headers 標頭中的每個標頭名稱，都必須符合規則中相對應的項目。Amazon S3 在要求的回應中只會傳送允許的標頭。如需可用於 Amazon S3 請求的標頭範例清單，請移至 Amazon 簡易儲存服務 API 參考指南中的常見請求標頭。

規則中的每個 AllowedHeader 字串最多可包含一個 * 萬用字元。例如，<AllowedHeader>x-amz-*</AllowedHeader> 會啟用所有 Amazon 專屬的標頭。

ExposeHeader 元素

每個ExposeHeader元素會在回應中識別您希望客戶能夠從其應用程式存取的標頭 (例如，從 JavaScriptXMLHttpRequest物件)。如需常見的 Amazon S3 回應標頭清單，請移至 Amazon 簡易儲存服務 API 參考指南中的常用回應標頭。

MaxAgeSeconds 元素

MaxAgeSeconds 元素會指定秒數，即您的瀏覽器為根據資源、HTTP 方法及來源所識別之預檢，快取要求的時間。

設定跨來源資源分享 (CORS)

跨來源資源分享 (CORS) 會定義一種方式，讓載入單一個網域的用戶端 Web 應用程式，能與不同網域中的資源互動。透過 CORS 支援，您可以使用 Amazon S3 建立功能豐富的用戶端 Web 應用程式，而且也可以選擇性地允許跨來源存取您的 Amazon S3 資源。

本節說明如何使用 Amazon S3 主控台、Amazon S3 REST API 和 AWS 開發套件啟用 CORS。若要設定儲存貯體允許跨來源要求，您可以將 CORS 組態新增至儲存貯體。CORS 組態是具有規則的文件，這些規則可識別允許存取儲存貯體的來源、每個來源支援的操作 (HTTP 方法)，以及其他操作特定資訊。在 S3 主控台中，CORS 組態必須為 JSON 文件。

如需有關 JSON 和 XML 中的 CORS 組態範例，請參閱 CORS 組態。

使用 S3 主控台

本節說明如何使用 Amazon S3 主控台將跨來源資源分享 (CORS) 組態新增至 S3 儲存貯體。

在儲存貯體上啟用 CORS 時，仍繼續適用存取控制清單 (ACL) 與其他存取許可政策。

> **Important**
>
> 在新的 S3 主控台中，CORS 組態必須為 JSON 格式。如需有關 JSON 和 XML 中的 CORS 組態範例，請參閱CORS 組態。

將 CORS 組態新增至 S3 儲存貯體

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇要建立儲存貯體政策的儲存貯體名稱。
3. 選擇 Permissions (許可)。
4. 在 Cross-origin resource sharing (CORS) (跨來源資源分享 (CORS)) 區段中，選擇 Edit (編輯)。
5. 在 CORS configuration editor (CORS 組態編輯器) 文字方塊中，輸入或複製並貼上新的 CORS 組態，或編輯現有組態。

   CORS 組態是 JSON 檔案。您在編輯器中輸入的文字必須是有效的 JSON。如需詳細資訊，請參閱 CORS 組態。
6. 選擇 Save changes (儲存變更)。

> **Note**
>
> Amazon S3 會顯示 CORS configuration editor (CORS 組態編輯器) 標題旁儲存貯體的 Amazon Resource Name (ARN)。如需 ARN 的詳細資訊，請參閱中的 Amazon 資源名稱 (ARN) 和 AWS 服務命名空間。Amazon Web Services 一般參考

使用 AWS 軟體開發套件

您可以使用 AWS SDK 管理值區的跨來源資源共用 (CORS)。如需 CORS 的詳細資訊，請參閱「使用跨來源資源分享 (CORS)」。

請參閱以下範例：

- 為建立 CORS 組態和設定儲存貯體組態
- 擷取組態與增加規則修改組態。
- 將經過修改的組態設定加入到儲存貯體中。
- 刪除組態

Java

Example

Example

如需如何建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketCrossOriginConfiguration;
import com.amazonaws.services.s3.model.CORSRule;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CORS {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        // Create two CORS rules.
```

```
        List<CORSRule.AllowedMethods> rule1AM = new
ArrayList<CORSRule.AllowedMethods>();
        rule1AM.add(CORSRule.AllowedMethods.PUT);
        rule1AM.add(CORSRule.AllowedMethods.POST);
        rule1AM.add(CORSRule.AllowedMethods.DELETE);
        CORSRule rule1 = new
CORSRule().withId("CORSRule1").withAllowedMethods(rule1AM)
                .withAllowedOrigins(Arrays.asList("http://*.example.com"));

        List<CORSRule.AllowedMethods> rule2AM = new
ArrayList<CORSRule.AllowedMethods>();
        rule2AM.add(CORSRule.AllowedMethods.GET);
        CORSRule rule2 = new
CORSRule().withId("CORSRule2").withAllowedMethods(rule2AM)
                .withAllowedOrigins(Arrays.asList("*")).withMaxAgeSeconds(3000)
                .withExposedHeaders(Arrays.asList("x-amz-server-side-encryption"));

        List<CORSRule> rules = new ArrayList<CORSRule>();
        rules.add(rule1);
        rules.add(rule2);

        // Add the rules to a new CORS configuration.
        BucketCrossOriginConfiguration configuration = new
BucketCrossOriginConfiguration();
        configuration.setRules(rules);

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Add the configuration to the bucket.
            s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

            // Retrieve and display the configuration.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            printCORSConfiguration(configuration);

            // Add another new rule.
            List<CORSRule.AllowedMethods> rule3AM = new
ArrayList<CORSRule.AllowedMethods>();
            rule3AM.add(CORSRule.AllowedMethods.HEAD);
```

```
            CORSRule rule3 = new
CORSRule().withId("CORSRule3").withAllowedMethods(rule3AM)
                    .withAllowedOrigins(Arrays.asList("http://www.example.com"));

            rules = configuration.getRules();
            rules.add(rule3);
            configuration.setRules(rules);
            s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

            // Verify that the new rule was added by checking the number of rules in
the
            // configuration.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            System.out.println("Expected # of rules = 3, found " +
configuration.getRules().size());

            // Delete the configuration.
            s3Client.deleteBucketCrossOriginConfiguration(bucketName);
            System.out.println("Removed CORS configuration.");

            // Retrieve and display the configuration to verify that it was
            // successfully deleted.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            printCORSConfiguration(configuration);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    private static void printCORSConfiguration(BucketCrossOriginConfiguration
configuration) {
        if (configuration == null) {
            System.out.println("Configuration is null.");
        } else {
            System.out.println("Configuration has " +
configuration.getRules().size() + " rules\n");

            for (CORSRule rule : configuration.getRules()) {
```

```
                System.out.println("Rule ID: " + rule.getId());
                System.out.println("MaxAgeSeconds: " + rule.getMaxAgeSeconds());
                System.out.println("AllowedMethod: " + rule.getAllowedMethods());
                System.out.println("AllowedOrigins: " + rule.getAllowedOrigins());
                System.out.println("AllowedHeaders: " + rule.getAllowedHeaders());
                System.out.println("ExposeHeader: " + rule.getExposedHeaders());
                System.out.println();
            }
        }
    }
}
```

.NET

Example

如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CORSTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CORSConfigTestAsync().Wait();
        }
        private static async Task CORSConfigTestAsync()
        {
```

```
            try
            {
                // Create a new configuration request and add two rules
                CORSConfiguration configuration = new CORSConfiguration
                {
                    Rules = new System.Collections.Generic.List<CORSRule>
                        {
                          new CORSRule
                          {
                            Id = "CORSRule1",
                            AllowedMethods = new List<string> {"PUT", "POST",
 "DELETE"},
                            AllowedOrigins = new List<string> {"http://
*.example.com"}
                          },
                          new CORSRule
                          {
                            Id = "CORSRule2",
                            AllowedMethods = new List<string> {"GET"},
                            AllowedOrigins = new List<string> {"*"},
                            MaxAgeSeconds = 3000,
                            ExposeHeaders = new List<string> {"x-amz-server-side-
encryption"}
                          }
                        }
                };

                // Add the configuration to the bucket.
                await PutCORSConfigurationAsync(configuration);

                // Retrieve an existing configuration.
                configuration = await RetrieveCORSConfigurationAsync();

                // Add a new rule.
                configuration.Rules.Add(new CORSRule
                {
                    Id = "CORSRule3",
                    AllowedMethods = new List<string> { "HEAD" },
                    AllowedOrigins = new List<string> { "http://www.example.com" }
                });

                // Add the configuration to the bucket.
                await PutCORSConfigurationAsync(configuration);
```

```
                // Verify that there are now three rules.
                configuration = await RetrieveCORSConfigurationAsync();
                Console.WriteLine();
                Console.WriteLine("Expected # of rulest=3; found:{0}",
configuration.Rules.Count);
                Console.WriteLine();
                Console.WriteLine("Pause before configuration delete. To continue,
click Enter...");
                Console.ReadKey();

                // Delete the configuration.
                await DeleteCORSConfigurationAsync();

                // Retrieve a nonexistent configuration.
                configuration = await RetrieveCORSConfigurationAsync();
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        static async Task PutCORSConfigurationAsync(CORSConfiguration configuration)
        {

            PutCORSConfigurationRequest request = new PutCORSConfigurationRequest
            {
                BucketName = bucketName,
                Configuration = configuration
            };

            var response = await s3Client.PutCORSConfigurationAsync(request);
        }

        static async Task<CORSConfiguration> RetrieveCORSConfigurationAsync()
        {
            GetCORSConfigurationRequest request = new GetCORSConfigurationRequest
            {
```

```
                BucketName = bucketName

            };
            var response = await s3Client.GetCORSConfigurationAsync(request);
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
            return configuration;
        }

        static async Task DeleteCORSConfigurationAsync()
        {
            DeleteCORSConfigurationRequest request = new
DeleteCORSConfigurationRequest
            {
                BucketName = bucketName
            };
            await s3Client.DeleteCORSConfigurationAsync(request);
        }

        static void PrintCORSRules(CORSConfiguration configuration)
        {
            Console.WriteLine();

            if (configuration == null)
            {
                Console.WriteLine("\nConfiguration is null");
                return;
            }

            Console.WriteLine("Configuration has {0} rules:",
configuration.Rules.Count);
            foreach (CORSRule rule in configuration.Rules)
            {
                Console.WriteLine("Rule ID: {0}", rule.Id);
                Console.WriteLine("MaxAgeSeconds: {0}", rule.MaxAgeSeconds);
                Console.WriteLine("AllowedMethod: {0}", string.Join(", ",
rule.AllowedMethods.ToArray()));
                Console.WriteLine("AllowedOrigins: {0}", string.Join(", ",
rule.AllowedOrigins.ToArray()));
                Console.WriteLine("AllowedHeaders: {0}", string.Join(", ",
rule.AllowedHeaders.ToArray()));
                Console.WriteLine("ExposeHeader: {0}", string.Join(", ",
rule.ExposeHeaders.ToArray()));
            }
```

```
        }
    }
}
```

使用 REST API

您可以使用 AWS Management Console，在儲存貯體上設定 CORS 組態。您也可以視應用程式之需要，直接傳送 REST 要求。Amazon 簡易儲存服務 API 參考 中的下列章節說明與 CORS 組態相關的 REST API 動作：

- PutBucketCors
- GetBucketCors
- DeleteBucketCors
- OPTIONS 物件

在 Amazon S3 中記錄和監控

監控是維護 Amazon S3 和 AWS 解決方案的可靠性、可用性和效能的重要組成部分。您應該從 AWS 解決方案的所有部分收集監視資料，以便在發生多點失敗時更輕鬆地偵錯。 AWS 提供數種工具來監控 Amazon S3 資源並回應潛在事件。

如需詳細資訊，請參閱 監控 Amazon S3。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱 什麼是 S3 Express One Zone？ 和 目錄值區。

Amazon CloudWatch 警報

使用 Amazon CloudWatch 警示，您可以在指定的時間段內觀看單一指標。如果指標超過指定臨界值，則會向 Amazon SNS 主題或 AWS Auto Scaling 政策傳送通知。 CloudWatch 警示不會叫用動作，因為它們處於特定狀態。必須是狀態已變更並維持了所指定的時間長度，才會呼叫動作。如需詳細資訊，請參閱 使用 Amazon 監控指標 CloudWatch。

AWS CloudTrail 日誌

CloudTrail 提供 Amazon S3 中使用者、角色或 AWS 服務所採取的動作記錄。使用收集的資訊 CloudTrail，您可以判斷向 Amazon S3 發出的請求、提出請求的來源 IP 地址、提出請求的人員、提出請求的時間以及其他詳細資訊。如需詳細資訊，請參閱 使用記錄 Amazon S3 API 呼叫 AWS CloudTrail。

Amazon GuardDuty

Amazon GuardDuty 是一種威脅偵測服務，可持續監控您的帳戶、容器、工作負載和 AWS 環境中的資料，以識別 S3 儲存貯體的潛在威脅或安全風險。 GuardDuty 也提供有關其偵測到之威脅的豐富內容。 GuardDuty 監控 AWS CloudTrail 管理記錄檔中的威脅，並顯示安全性相關資訊。例如， GuardDuty 將包括 API 請求的因素，例如發出請求的用戶，發出請求的來源位置以及請求的特定 API，這在您的環境中可能是不尋常的。GuardDuty S3 防護會監控所收集的 S3 資料事件，CloudTrail 並識別環境中所有 S3 儲存貯體中的潛在異常和惡意行為。

Amazon S3 存取日誌

伺服器存取日誌提供了對儲存貯體進行請求的詳細記錄。伺服器存取日誌對許多應用程式來說，都是個很有用的資料。舉例來說，存取記錄資訊在安全與存取稽核中相當實用。如需詳細資訊，請參閱 使用伺服器存取記錄記錄要求。

AWS Trusted Advisor

Trusted Advisor 利用為數十萬名 AWS 客戶提供服務所學到的最佳實踐。 Trusted Advisor 檢查您的 AWS 環境，然後在存在機會時提出建議，以節省資金、改善系統可用性和效能，或協助縮小安全性漏洞。所有 AWS 客戶都可以使用五張 Trusted Advisor 支票。擁有商業或企業支援方案的客戶可以檢視所有 Trusted Advisor 檢查。

Trusted Advisor 具有以下與亞馬遜 S3 相關的檢查：

- Amazon S3 儲存貯體的記錄組態。
- 對於有開放式存取許可的 Amazon S3 儲存貯體，進行安全檢查。
- 對於未啟用版本控制或已暫停版本控制的 Amazon S3 儲存貯體，進行容錯能力檢查。

如需詳細資訊，請參閱《AWS Support 使用者指南》中的 AWS Trusted Advisor。

以下安全最佳實務也可用來解決記錄和監控：

- Identify and audit all your Amazon S3 buckets
- Implement monitoring using Amazon Web Services monitoring tools
- 啟用 AWS Config
- Enable Amazon S3 server access logging
- Use CloudTrail
- Monitor Amazon Web Services security advisories

Amazon S3 的合規驗證

Amazon S3 的安全性和合規是由第三方稽核人員評估為多個 AWS 合規計劃的一部分，包括下列各項：

- 系統和組織控制 (SOC)
- 支付卡產業資料安全標準 (PCI DSS)
- 聯邦風險與授權管理計劃 (FedRAMP)
- 美國健康保險流通與責任法案 (HIPAA)

AWS 在符合性計劃的 AWS 服務範圍中，提供特定合規方案範圍內經常更新的AWS 服務清單。

第三方稽核報告可供您使用下載 AWS Artifact。如需詳細資訊，請參閱在 AWS Artifact 中下載報表。

如需 AWS 規範遵循方案的詳細資訊，請參閱AWS 合規性方案。

使用 Amazon S3 時的合規責任取決於資料的敏感度、組織的合規目標，以及適用法律和法規。若您使用的 Amazon S3 必須遵循特定標準 (如 HIPAA、PCI 或 FedRAMP)， AWS 會提供資源予以協助：

- 安全性與合規性快速入門指南，討論在上部署以安全性和法規遵循為重點的基準環境的架構考量和步驟。 AWS
- HIPAA 安全性與法規遵循架構概述公司如何使用 AWS 來協助他們符合 HIPAA 要求。
- AWS 合規性資源提供數種不同的工作簿和指南，可能適用於您的產業和位置。
- AWS Config 可用來評定資源組態與內部實務、業界準則和法規的合規狀態。
- AWS Security Hub 提供您內部安全性狀態的全面檢視， AWS 並協助您檢查您是否符合安全性產業標準和最佳做法。
- 使用 S3 物件鎖定 可協助您滿足金融服務領域監管機構 (如 SEC、FINRA 和 CFTC) 的技術需求，這些機構需要單寫多讀 (WORM) 資料儲存體來處理特定類型的書籍和記錄資訊。
- Amazon S3 清查可協助您稽核與回報物件的複寫和加密狀態，以滿足業務、合規及法規需求。

Amazon S3 的恢復能力

AWS 全球基礎架構是圍繞區域和可用區域建立的。 AWS 區域 提供多個實體分離且隔離的可用區域，以低延遲、高輸送量和高度備援的網路連線。這些可用區域可讓您有效設計和操作應用程式與資料庫，它們的可用性、容錯能力和擴展性都比單一或多個資料中心的傳統基礎設施還高。如果您特別需要在更大的地理距離上複製資料，您可以使用複製物件概觀，這可以在不同儲存貯體之間自動、非同步複製物件 AWS 區域。

每個都 AWS 區域 有多個可用區域。您可以在相同區域中的多個可用區域間部署應用程式，以提供容錯能力與低延遲效能。可用區域會透過快速的私有光纖聯網彼此連接，讓您能夠輕鬆地建構能在可用區域間自動容錯移轉的應用程式，而不會發生中斷。

如需 AWS 區域 和可用區域的詳細資訊，請參閱AWS 全域基礎結構。

除了 AWS 全球基礎設施之外，Amazon S3 還提供多種功能來協助支援您的資料彈性和備份需求。

生命週期組態

生命週期組態是一組規則，可定義 Amazon S3 套用至一組物件的動作。透過生命週期組態規則，您可以指示 Amazon S3 將物件轉換為較便宜的儲存體方案、進行封存或刪除。如需詳細資訊，請參閱 管理儲存生命週期。

版本控制

版本控制是在相同儲存貯體中保留多個物件版本的方式。您可以使用版本控制功能來保留、擷取和恢復在 Amazon S3 儲存貯體中存放的每個物件的每個版本。透過版本控制，您就可以輕鬆地復原失誤的使用者動作和故障的應用程式。如需詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。

S3 物件鎖定

您可以使用 S3 物件鎖定功能搭配「單寫多讀」 (WORM) 模式來存放物件。透過 S3 物件鎖定功能，您可以在固定時間內或永遠避免刪除或覆寫物件。S3 物件鎖定功能可讓您符合必須使用 WORM 儲存體的法規要求，或單純多加一道保護，以免物件遭到變更或刪除。如需詳細資訊，請參閱 使用 S3 物件鎖定。

儲存類別

Amazon S3 根據您的工作負載需求，提供各種儲存類別供選擇。S3 標準 – IA 和 S3 單區域 – IA 儲存類別是針對您每月存取約一次且需要毫秒存取的資料所設計。S3 Glacier Instant Retrieval 儲存類別專為長期存在且可以毫秒存取的封存資料而設計，您可以每季度存取一次。對於不需要立即存取的封存資料，例如備份，您可以使用 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別。如需詳細資訊，請參閱 使用 Amazon S3 儲存體方案。

以下安全最佳實務也可用來解決復原問題：

- Enable versioning
- Consider Amazon S3 cross-region replication
- Identify and audit all your Amazon S3 buckets

Amazon S3 備份的加密

如果您使用 Amazon S3 來儲存備份，備份的加密取決於這些儲存貯體的組態。Amazon S3 提供為 S3 儲存貯體設定預設加密行為的方式。您可以在儲存貯體上設定預設加密，讓所有物件在存放於儲存貯體中時維持加密狀態。預設加密支援儲存在 AWS KMS (SSE-KMS) 中的金鑰。如需詳細資訊，請參閱 對 Amazon S3 儲存貯體設定預設伺服器端加密行為。

如需版本控制和物件鎖定的詳細資訊，請參閱下列主題：在 S3 儲存貯體中使用版本控制 使用 S3 物件鎖定

Amazon S3 的基礎設施安全性

作為受管服務，Amazon S3 受到 AWS Well-Architected 架構安 AWS 全支柱中所述的全球網路安全程序的保護。

透過網路存取 Amazon S3 是透過 AWS 已發佈的 API。用戶端必須支援 Transport Layer Security (TLS) 1.2。我們也建議同時支援 TLS 1.3。如需有關此建議的詳細資訊，請參閱AWS 安全性部落格上的 TLS 1.3 加快 AWS 雲端連線速度)。用戶端還必須支援具備完全正向加密 (PFS) 功能的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。此外，請求必須使用簽 AWS 章 V4 或簽 AWS 章 V2 簽署，而且需要提供有效的認證。

您可以從任何網路位置來呼叫這些 API，但 Amazon S3 所支援的資源類型存取政策可能包含與來源 IP 地址相關的限制。您也可以使用 Amazon S3 儲存貯體政策來控制從特定 Virtual Private Cloud (VPC) 端點或特定 VPC 存取儲存貯體。實際上，這只能將對指定 Amazon S3 儲存貯體的網路存取與網路內的特定 VPC 隔離。 AWS 如需詳細資訊，請參閱 使用儲存貯體政策控制來自 VPC 端點的存取。

以下安全最佳實務也可用來解決 Amazon S3 中的基礎設施安全問題：

- Consider VPC endpoints for Amazon S3 access
- Identify and audit all your Amazon S3 buckets

Amazon S3 中的組態與漏洞分析

AWS 處理基本安全性工作，例如客體作業系統 (OS) 和資料庫修補、防火牆組態和嚴重損壞修復。這些程序已由適當的第三方進行檢閱並認證。如需詳細資訊，請參閱以下 資源：

- Amazon S3 的合規驗證
- 共同的責任模型
- Amazon Web Services：安全程序概觀

以下安全最佳實務也可處理 Amazon S3 中的組態和漏洞分析：

- Identify and audit all your Amazon S3 buckets
- 啟用 AWS Config

Amazon S3 的安全最佳實務

在您開發和實作自己的安全政策時，可考慮使用 Amazon S3 提供的多種安全功能。以下最佳實務為一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

主題

Amazon S3 安全最佳實務

下列 Amazon S3 最佳實務有助於預防安全事件的發生。

停用存取控制清單 (ACL)

S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來控制上傳至儲存貯體之物件的擁有權，以及停用或啟用 ACL。根據預設，物件擁有權設定為「儲存貯體擁有者強制執行」設定，而且所有 ACL 都會停用。停用 ACL 時，儲存貯體擁有者會擁有儲存貯體中的所有物件，並使用存取管理政策專門管理對資料的存取。

Amazon S3 中的大多數現代使用案例不再需要使用存取控制清單 (ACL)。建議您停用 ACL，除非在異常情況下必須個別控制每個物件的存取。若要停用 ACL 並取得儲存貯體中每個物件的擁有權，請針對 S3 物件擁有權套用儲存貯體擁有者強制執行設定。停用 ACL 時，您可以輕鬆地維護具有由不同 AWS 帳戶上傳之物件的儲存貯體。

停用 ACL 時，您資料的存取控制是以政策為基礎，如下所示：

- AWS Identity and Access Management (IAM) 使用者政策
- S3 儲存貯體政策
- 虛擬私有雲端 (VPC) 端點政策
- AWS Organizations 服務控制政策 (SCP)

停用 ACL 可簡化許可管理和稽核。新的儲存貯體預設會停用 ACL。您也可以停用現有儲存貯體的 ACL。如果您具有的現有儲存貯體中已有物件，則在您停用 ACL 之後，物件和儲存貯體 ACL 不再是存取評估程序的一部分。相反地，會根據原則授予或拒絕存取權。

在您可以停用 ACL 之前，請務必執行下列動作：

- 檢閱您的儲存貯體政策，以確保其涵蓋您打算將存取權授予帳戶外儲存貯體的所有方式。
- 將儲存貯體 ACL 重設為預設值 (對儲存貯體擁有者的完全控制)。

停用 ACL 之後，會發生下列行為：

- 您的儲存貯體只接受未指定 ACL 的 PUT 請求，或具有儲存貯體擁有者完全控制 ACL 的 PUT 請求。這些 ACL 包括 `bucket-owner-full-control` 固定 ACL，或以 XML 表示此 ACL 的對等形式。
- 支援儲存貯體擁有者完整控制 ACL 的現有應用程式沒有任何影響。
- PUT包含其他 ACL (例如，特定的自訂授權 AWS 帳戶) 的要求會失敗，並傳回含有錯誤碼`400 (Bad Request)`的 HTTP 狀態碼`AccessControlListNotSupported`。

如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

確保 Amazon S3 儲存貯體使用正確的政策且不可公開存取

除非您有明確要求網際網路上的任何人都能讀取或寫入您的 S3 儲存貯體，否則請確保您的 S3 儲存貯體不是公用儲存貯體。以下是您可以採取以封鎖公開存取的一些步驟：

- 使用 S3 封鎖公開存取。使用 S3 封鎖公開存取，您可以輕鬆地設定集中控制，來限制對 Amazon S3 的公開存取。無論資源的建立方式為何，都會強制執行這些集中控制。如需詳細資訊，請參閱 封鎖對 Amazon S3 儲存體的公開存取權。
- 識別允許萬用字元身分 (例如 `"Principal": "*"`，這實際上意味著「任何人」) 的 Amazon S3 儲存貯體政策。此外，請尋找允許萬用字元動作 `"*"` (實際上允許使用者在 Amazon S3 儲存貯體中執行任何動作) 的政策。
- 同樣地，尋找 Amazon S3 儲存貯體存取控制清單 (ACL)，這些清單提供對「所有人」或「任何經過驗證的 AWS 使用者」的讀取、寫入或完整存取權。
- 使用 `ListBuckets` API 操作掃描所有的 Amazon S3 儲存貯體。然後，使用 `GetBucketAcl`、`GetBucketWebsite` 和 `GetBucketPolicy` 來判斷每個儲存貯體的存取控制和組態是否符合規範。
- 使用 AWS Trusted Advisor 檢查您的 Amazon S3 實作。
- 考慮藉由使用 s3-bucket-public-read-prohibited 和 s3-bucket-public-write-prohibited 受管 AWS Config 規則實作不間斷的偵測性控制。

如需詳細資訊，請參閱 適用於 Amazon S3 的 Identity and Access Management。

使用 Amazon 識別對您的 Amazon S3 儲存貯體的潛在威脅 GuardDuty

Amazon GuardDuty 是一種威脅偵測服務，可識別您的帳戶、容器、工作負載和 AWS 環境中資料的潛在威脅。透過使用機器學習 (ML) 模型以及異常和威脅偵測功能，Amazon 會 GuardDuty 持續監控不同的資料來源，以識別環境中潛在的安全風險和惡意活動並排定優先順序。啟用時 GuardDuty，它會針對包含AWS CloudTrail 管理事件、VPC 流程記錄和 DNS 記錄的基礎資料來源提供威脅偵測。若要將威脅偵測延伸至 S3 儲存貯體中的資料平面事件，您可以啟用 GuardDuty S3 保護功能。此功能可偵測威脅，例如資料洩漏和透過 Tor 節點存取 S3 儲存貯體的可疑資料。GuardDuty也會在您的環境中建立一個正常的基準模式，當它識別出可能的異常行為時，它會提供內容相關資訊，協助您修復可能受損的 S3 儲存貯體或 AWS 登入資料。如需詳細資訊，請參閱GuardDuty。

實作最低權限存取

當您授予許可時，需決定哪些使用者會取得哪些 Amazon S3 資源的許可。您還需針對這些資源啟用允許執行的動作，因此，建議您只授予執行任務所需的許可。對降低錯誤或惡意意圖所引起的安全風險和影響而言，實作最低權限存取是相當重要的一環。

下列工具可用來實作最低權限存取：

- Amazon S3 的政策動作 和 IAM 實體的許可界限
- Amazon S3 如何與 IAM 配合使用
- 存取控制清單 (ACL) 概觀
- 服務控制政策

如需選擇上述一或多個機制時應考量事項的指導方針，請參閱 適用於 Amazon S3 的 Identity and Access Management。

針對需要 Amazon S3 存取的應 AWS 服務 用程式使用 IAM 角色

為了讓在 Amazon EC2 或其他地方執行的應用程式能 AWS 服務 夠存取 Amazon S3 資源，這些應用程式必須在其 AWS API 請求中包含有效的 AWS 登入資料。建議您不要將 AWS 登入資料直接存放在應用程式或 Amazon EC2 執行個體中。這些是不會自動輪換的長期憑證，如果遭到盜用，可能會對業務造成嚴重的影響。

反之，請使用 IAM 角色，為需要存取 Amazon S3 的應用程式和服務管理臨時憑證。使用角色時，您不需要將長期登入資料 (例如使用者名稱和密碼或存取金鑰) 分發到 Amazon EC2 執行個體或 AWS 服務例如 AWS Lambda. 該角色提供應用程式在呼叫其他 AWS 資源時可以使用的臨時權限。

如需詳細資訊，請參閱《IAM 使用者指南》中的以下主題：

- IAM 角色

- 常見的角色方案：使用者、應用程式和服務

考慮靜態資料加密

下列選項皆可讓您保護 Amazon S3 中的靜態資料：

- 伺服器端加密 — 所有 Amazon S3 儲存貯體預設都已設定加密，所有上傳到 S3 儲存貯體的新物件都會在靜態時自動加密。伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 是 Amazon S3 中每個儲存貯體的預設加密組態。若要使用不同類型的加密，您可以指定 S3 PUT 請求中要使用的伺服器端加密類型，也可以在目的地儲存貯體中設定預設加密組態。

  Amazon S3 也提供下列伺服器端加密選項：

  - 使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 進行伺服器端加密
  - 使用 AWS Key Management Service (AWS KMS) 金鑰 (DSSE-KMS) 進行雙層伺服器端加密
  - 使用客戶提供金鑰 (SSE-C) 的伺服器端加密

  如需詳細資訊，請參閱 使用伺服器端加密保護資料。
- 用戶端加密 - 加密用戶端資料，並將加密的資料上傳至 Amazon S3。在這種情況下，您可以管理加密程序、加密金鑰和相關工具。和伺服器端加密的運作方式一樣，用戶端加密不會使用資料本身所在機制中存放的金鑰來加密資料，而是使用存放在另一套機制中的金鑰，藉此幫助您減少風險。

  Amazon S3 提供多個用戶端加密選項。如需詳細資訊，請參閱 使用用戶端加密保護資料。

強制加密傳輸中的資料

您可以使用 HTTPS (TLS) 來協助防止潛在攻擊者透過使 person-in-the-middle 用或類似攻擊竊取網路流量或操控網路流量。建議在您的 Amazon S3 儲存貯體政策中使用 aws:SecureTransport，僅允許透過 HTTPS (TLS) 進行加密連線。

> **Important**
>
> 我們建議您的應用程式不要釘選 Amazon S3 TLS 憑證，因為 AWS 不支援釘選公開信任的憑證。S3 會自動更新憑證，並且可以在憑證到期前的任何時間進行續訂。續訂憑證會產生新的公開-私 key pair。如果您釘選了最近使用新公開金鑰更新的 S3 憑證，則在應用程式使用新憑證之前，您將無法連線到 S3。

此外，考慮藉由使用 s3-bucket-ssl-requests-only 受管 AWS Config 規則實作不間斷的偵測性控制。

考慮使用 S3 物件鎖定

搭配 S3 物件鎖定，您可以使用「單寫多讀」(WORM) 模型來存放物件。S3 物件鎖定功能有助於避免資料不慎遭刪除或遭到不當刪除的現象。例如，您可以使用 S3 物件鎖定來協助保護您的 AWS CloudTrail 日誌。

如需詳細資訊，請參閱 使用 S3 物件鎖定。

啟用 S3 版本控制

S3 版本控制是在相同儲存貯體中保留多個物件版本的方式。您可以使用版本控制功能來保留、擷取和恢復在 儲存貯體中所存放每個物件的各個版本。透過版本控制，您就可以輕鬆地復原失誤的使用者動作和故障的應用程式。

此外，考慮藉由使用 s3-bucket-versioning-enabled 受管 AWS Config 規則實作不間斷的偵測性控制。

如需詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。

考慮使用 S3 跨區域複寫

雖然 Amazon S3 預設會跨多個地理位置不同的可用區域存放資料，但合規要求可能會需要您在更遠的距離下存放資料。使用 S3 跨區域複寫 (CRR)，您可以在遙遠之間複寫資料，以協 AWS 區域助滿足這些需求。CRR 可讓不同值區中的物件自動、非同 AWS 區域步複製。如需詳細資訊，請參閱 複製物件概觀。

> **Note**
>
> CRR 要求來源和目的地 S3 儲存貯體都必須啟用版本控制。

此外，考慮藉由使用 s3-bucket-replication-enabled 受管 AWS Config 規則實作不間斷的偵測性控制。

考慮使用 VPC 端點進行 Amazon S3 存取

適用於 Amazon S3 的 Virtual Private Cloud (VPC) 端點是 VPC 中的邏輯實體，僅允許連線到 Amazon S3。VPC 端點可協助防止流量周遊開放的網際網路。

適用於 Amazon S3 的 VPC 端點提供多種方式來控制對 Amazon S3 資料的存取：

- 您可以使用 S3 儲存貯體政策，控制經由特定 VPC 端點允許的請求、使用者或群組。

- 您可以使用 S3 儲存貯體政策來控制能存取 S3 儲存貯體的 VPC 或 VPC 端點。
- 您可以使用不具備網際網路閘道的 VPC 來防止資料外洩。

如需詳細資訊，請參閱 使用儲存貯體政策控制來自 VPC 端點的存取。

使用託管 AWS 安全服務監控數據安全

多種受管 AWS 安全服務可協助您識別、評估和監控 Amazon S3 資料的安全與合規風險。這些服務也可以協助您保護資料免於這些風險。這些服務包括自動偵測、監控和保護功能，這些功能旨在從 Amazon S3 資源擴展 AWS 帳戶 到適用於跨數千個帳戶的組織的資源。

如需詳細資訊，請參閱 使用託管安全服務監控數據 AWS 安全。

Amazon S3 監控和稽核最佳實務

下列 Amazon S3 最佳實務有助於偵測潛在安全弱點與事件。

識別並稽核所有 Amazon S3 儲存貯體

識別 IT 資產是控管和保障安全的重要環節。您必須掌握所有 Amazon S3 資源，才能存取其安全狀態並對潛在弱點採取行動。若要稽核您的資源，建議您執行下列動作：

- 使用標籤編輯器來識別並標記重視安全性或重視稽核的資源，接著在需要搜尋這些資源時使用這些標籤。若要取得更多資訊，請參閱〈標記資源使用指南〉中的〈搜尋要標記的 AWS 資源〉。
- 使用 S3 清查來稽核與回報物件的複寫和加密狀態，以滿足業務、合規及法規需求。如需詳細資訊，請參閱 Amazon S3 清查。
- 為 Amazon S3 資源建立資源群組。如需詳細資訊，請參閱《AWS Resource Groups 使用者指南》中的什麼是 Resource Groups？。

使用監視工具實作 AWS 監控

監控是維護 Amazon S3 和 AWS 解決方案的可靠性、安全性、可用性和效能的重要組成部分。 AWS 提供多種工具和服務，協助您監控 Amazon S3 和您的其他工具 AWS 服務。例如，您可以監控 Amazon S3 的 Amazon CloudWatch 指標，尤其是`PutRequests`、`GetRequests`、`4xxErrors`、和`DeleteRequests`指標。如需詳細資訊，請參閱 使用 Amazon 監控指標 CloudWatch 及 監控 Amazon S3。

如需第二個範例，請參閱範例：Amazon S3 儲存貯體活動。此範例說明如何建立在 Amazon S3 API 呼叫或儲存貯體政策、儲存貯體生命週期`PUT`或`DELETE`儲存貯體複寫組態或儲存貯體 ACL 時觸發`PUT`的 CloudWatch 警示。

啟用 Amazon S3 伺服器存取記錄

伺服器存取記錄會根據向儲存貯體發出的請求來提供詳細記錄。伺服器存取日誌可幫助您進行安全與存取稽核，讓您了解自己的客戶群並掌握 Amazon S3 帳單。如需啟用伺服器存取日誌的操作說明，請參閱 使用伺服器存取記錄記錄要求。

也可以考慮使用s3-bucket-logging-enabled AWS Config 受管規則來實作持續的偵測控制項。

使用 AWS CloudTrail

AWS CloudTrail 提供使用者、角色或 Amazon S3 AWS 服務 中所採取的動作記錄。您可以使用收集的信息 CloudTrail 來確定以下內容：

- 對 Amazon S3 提出的請求
- 提出請求的 IP 地址
- 提出要求的人員
- 提出請求的時間
- 有關請求的其他詳細資訊

例如，您可以識別影響資料存取的PUT動作 CloudTrail 項目`PutBucketAcl`，特別是`PutObjectAcl`、`PutBucketPolicy`、、和`PutBucketWebsite`。

設置時 AWS 帳戶，默認情況下 CloudTrail 處於啟用狀態。您可以在 CloudTrail 主控台中檢視最近的事件。若要為 Amazon S3 儲存貯體建立持續的活動和事件記錄，您可以在 CloudTrail 主控台中建立追蹤。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的記錄資料事件。

建立追蹤時，您可以設定 CloudTrail 為記錄資料事件。資料事件是在資源上或在資源內執行的資源操作記錄。在 Amazon S3 中，資料事件會記錄個別儲存貯體的物件層級 API 活動。CloudTrail 支援 Amazon S3 物件層級 API 操作的一個子集，例如`GetObject`、`DeleteObject`、和`PutObject`。如需有關如何使 CloudTrail 用 Amazon S3 的詳細資訊，請參閱使用記錄 Amazon S3 API 呼叫 AWS CloudTrail。在 Amazon S3 主控台中，您也可以將 S3 儲存貯體設為 啟用 S3 儲存貯體和物件的 CloudTrail 事件記錄。

AWS Config 提供受管規則 (`cloudtrail-s3-dataevents-enabled`)，您可以用來確認至少有一個 CloudTrail 追蹤正在記錄 S3 儲存貯體的資料事件。如需詳細資訊，請參閱《AWS Config 開發人員指南》中的 cloudtrail-s3-dataevents-enabled。

啟用 AWS Config

本主題中列出的數個最佳作法建議您建立 AWS Config 規則。 AWS Config 協助您評估、稽核和評估資 AWS 源的組態。 AWS Config 監控資源配置，以便您可以根據所需的安全配置評估記錄的配置。使用 AWS Config，您可以執行下列動作：

- 檢閱 AWS 資源之間組態和關係的變更。
- 調查詳細的資源組態歷史記錄
- 判斷您對內部指導方針中所指定組態的整體合規情形。

使用 AWS Config 可協助您簡化法規遵循稽核、安全性分析、變更管理及作業疑難排解。如需詳細資訊，請參閱AWS Config 開發人員指南中的 AWS Config 使用主控台進行設定。當您指定要記錄的資源類型時，請確定其中包含 Amazon S3 資源。

> **Important**
>
> AWS Config 受管規則僅在評估 Amazon S3 資源時支援一般用途儲存貯體。 AWS Config 不會記錄目錄值區的組態變更。如需詳細資訊，請參閱AWS ConfigAWS Config 開發人員指南中的 AWS Config 受管規則和受管規則清單。

如需如何使用的範例 AWS Config，請參閱AWS 安全部落格上的如 AWS Config 何使用監控和回應允許公用存取的 Amazon S3 儲存貯體。

使用 Amazon Macie 探索敏感資料

Amazon Macie 是一種安全服務，透過使用機器學習和模式比對來探索敏感資料。Macie 提供資料安全風險的可見性，並啟用自動防護來防範這些風險。使用 Macie，您可以自動探索和報告 Amazon S3 資料資產中的敏感資料，以更加了解貴組織存放在 S3 中的資料。

若要使用 Macie 偵測敏感資料，您可以使用內建準則和技術，其旨在偵測許多國家和地區的龐大和不斷增長的敏感資料類型。這些敏感資料類型包括多種類型的個人身分識別資訊 (PII)、財務資料和憑證資料。您也可以使用您定義的自訂準則，即定義要比對之文字模式的規則表達式，以及可選擇的字元序列和精簡結果的鄰近規則。

如果 Macie 在 S3 物件中偵測到敏感資料，Macie 會產生安全調查結果以通知您。此調查結果提供受影響物件的相關資訊、Macie 所找到敏感資料的類型和出現次數，以及協助您調查受影響 S3 儲存貯體和物件的其他詳細資訊。如需詳細資訊，請參閱 Amazon Macie 使用者指南。

使用 S3 Storage Lens

S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。S3 Storage Lens 也會分析指標，以提供內容相關建議，您可以用來最佳化儲存成本，並套用最佳實務保護您的資料。

透過 S3 Storage Len，您可以使用指標產生摘要洞見，例如了解您在整個組織中擁有多少儲存體，或是成長速度最快的儲存貯體和字首有哪些。您也可以使用 S3 Storage Lens 指標，識別成本最佳化機會、實作資料保護和存取管理最佳實務，以及改善應用程式工作負載的效能。

例如，您可以識別沒有 S3 生命週期規則的儲存貯體，這些規則可中止超過 7 天未完成的分段上傳。您也可以識別未遵循資料保護最佳實務的儲存貯體，例如使用 S3 複寫或 S3 版本控制。如需詳細資訊，請參閱瞭解 Amazon S3 Storage Lens。

監控 AWS 安全建議

建議您定期檢查針對 AWS 帳戶張貼在 Trusted Advisor 中的安全建議。尤其，尋找具有「開放式存取許可」之 Amazon S3 儲存貯體的相關警告。您可以使用 describe-trusted-advisor-checks，以程式設計方式執行此動作。

此外，主動監控您每個人註冊的主要電子郵件地址 AWS 帳戶。 AWS 使用此電子郵件地址與您聯繫，以了解可能會影響您的新興安全問題。

AWS 具有廣泛影響的操作問題張貼在 AWS Health Dashboard -服務健康狀態上。也會透過 AWS Health Dashboard將操作問題張貼至個別帳戶。如需詳細資訊，請參閱 AWS Health 文件。

使用託管安全服務監控數據 AWS 安全

多種受管 AWS 安全服務可協助您識別、評估和監控 Amazon S3 資料的安全與合規風險。它們還可以協助您保護資料免於這些風險。這些服務包括自動偵測、監控和保護功能，這些功能旨在從 Amazon S3 資源擴展 AWS 帳戶 到適用於數千個組織的資源 AWS 帳戶。

AWS 偵測與回應服務可協助您識別潛在的安全性錯誤設定、威脅或未預期的行為，以便您快速回應環境中潛在未經授權或惡意的活動。 AWS 資料保護服務可協助您監控並保護您的資料、帳戶和工作負載，防止未經授權的存取。它們也可以協助您在 Amazon S3 資料資產中探索敏感資料，例如個人身分識別資訊 (PII)。

為了協助您識別並評估資料安全和合規風險，受管 AWS 安全服務會產生調查結果，以將 Amazon S3 資料的潛在安全事件或問題通知您。這些調查結果會提供相關詳細資訊，您可以用來根據事件回應工作流程和政策調查、評估這些風險並採取行動。您可以使用每個服務直接存取調查結果資料。您也可以將資料傳送至其他應用程式、服務和系統，例如安全事件和事件管理系統 (SIEM)。

若要監控 Amazon S3 資料的安全性，請考慮使用這些受管 AWS 安全服務。

Amazon GuardDuty

Amazon GuardDuty 是一種威脅偵測服務，可持續監控您 AWS 帳戶 和工作負載中是否有惡意活動，並提供詳細的安全發現結果以提供可見性和修復。

使用中的 S3 保護功能 GuardDuty，您可以設定 GuardDuty 為分析 Amazon S3 資源的 AWS CloudTrail 管理和資料事件。 GuardDuty 然後監視這些事件是否存在惡意和可疑活動。為了通知分析並識別潛在的安全風險，請 GuardDuty 使用威脅情報饋送和機器學習。

GuardDuty 可以為您的 Amazon S3 資源監控不同類型的活動。例如，Amazon S3 的 CloudTrail 管理事件包括儲存貯體層級的操作，例如`ListBuckets`、`DeleteBucket`和。`PutBucketReplication` CloudTrail Amazon S3 的資料事件包括物件層級操作，例如`GetObject`、`ListObjects`、和`PutObject`。如果 GuardDuty 偵測到異常或潛在惡意活動，它會產生一個發現通知您。

有關更多信息，請參閱 Amazon GuardDuty 用戶指南 GuardDuty中的 Amazon S3 保護。

Amazon Detective

Amazon Detective 可簡化調查程序，並協助您進行更快、更有效的安全調查。Detective 提供預先建置的資料彙總、摘要和內容，可協助您分析並評估潛在安全問題的本質和範圍。

Detective 會自動擷取以時間為基礎的事件，例如來自資源的 API 呼叫 AWS CloudTrail 和 Amazon VPC 流程日誌。 AWS 它還會擷取 Amazon GuardDuty 產生的發現。Detective 接著會使用機器學習、統計分析和圖論來產生視覺化，協助您更快地進行有效的安全調查。

這些視覺效果提供了資源行為的統一互動式檢視，以及它們之間一段時間後的互動。您可以探索此行為圖表，以檢查潛在的惡意動作，例如失敗的登入嘗試或可疑的 API 呼叫。您也可以查看這些動作如何影響資源，例如 S3 儲存貯體和物件。

如需詳細資訊，請參閱 Amazon Detective 管理指南。

IAM Access Analyzer

AWS Identity and Access Management Access Analyzer (IAM 存取分析器) 可協助您識別與外部實體共用的資源。您也可以使用 IAM 存取分析器根據政策文法和最佳實務來驗證 IAM 政策，並根據日 AWS CloudTrail 誌中的存取活動產生 IAM 政策。

IAM Access Analyzer 使用邏輯推理來分析 AWS 環境中的資源政策，例如儲存貯體政策。使用適用於 S3 的 IAM Access Analyzer，當 S3 儲存貯體設定為允許存取網際網路上的任何人 (包括組織外部帳戶) 時 AWS 帳戶，系統就會收到警示。例如，IAM Access Analyzer for S3 可以報告儲存貯體具有透過儲存貯體存取控制清單 (ACL)、儲存貯體政策、多區域存取點政策或存取點政策提供的讀取或寫入存取權。針對每個公開或共用儲存貯體，您會收到調查結果，指出公開或共用存取的來源和層級。使用這些調查結果，您可以採取立即且精確的更正動作，將儲存貯體存取還原成您想要的內容。

如需詳細資訊，請參閱 使用 IAM Access Analyzer for S3 檢閱儲存貯體存取權。

Amazon Macie

Amazon Macie 是一種資料安全服務，透過使用機器學習和模式比對來探索敏感資料、提供資料安全風險的可見性，以及啟用自動保護以防範這些風險。

使用 Macie，您可以自動探索和報告 S3 儲存貯體中的敏感資料，以更加了解貴組織存放在 Amazon S3 中的資料。若要偵測敏感資料，您可以使用 Macie 提供的內建條件和技術，您定義的自訂條件，或兩者的組合。如果 Macie 在 S3 物件中偵測到敏感資料，Macie 會產生調查結果以通知您。此調查結果提供受影響儲存貯體和物件的相關資訊、Macie 所找到敏感資料的類型和出現次數，以及協助您調查的其他詳細資訊。

Macie 還提供統計資訊和其他資料，讓您洞察 Amazon S3 資料的安全狀態，而且它會針對安全和存取控制自動評估和監控 S3 儲存貯體。如果 Macie 偵測到您資料的安全性或隱私權存在潛在問題，例如變成可公開存取的儲存貯體，Macie 會視需要產生調查結果來檢閱並修補此問題。

如需詳細資訊，請參閱 Amazon Macie 使用者指南。

AWS Security Hub

AWS Security Hub 是一種安全性狀態管理服務，可執行安全性最佳作法檢查、將來自多個來源的警示和發現項目彙總為單一格式，並啟用自動補救。

Security Hub 會從整合式安全解決方案收集並提供 AWS Partner Network 安全發現資料 AWS 服務，包括 Amazon Detective GuardDuty、亞馬遜、IAM 存取分析器和 Amazon Macie。它還會根據 AWS 最佳實務和支援的業界標準執行連續的自動化安全檢查，產生自己的發現結果。

安全中樞接著會將提供者的調查結果相互關聯和合併，協助您優先處理最重要的調查結果。它也提供自訂動作的支援，您可以使用這些動作，來叫用特定調查結果類別的回應或修補動作。

使用 Security Hub，您可以評估 Amazon S3 資源的安全和合規狀態，並且可以進行更廣泛分析組織在個別 AWS 區域 和多個區域的安全狀態的一部分。這包括分析安全趨勢，以及識別優先順序最高的安全問題。您也可以彙總來自多個 AWS 區域的調查結果，以及監控並處理來自單一區域的彙總調查結果資料。

如需詳細資訊，請參閱《AWS Security Hub 使用者指南》中的 Amazon Simple Storage Service 控制。

管理您的 Amazon S3 儲存體

在 Amazon S3 中建立儲存貯體並上傳物件之後，您可以使用版本控制、儲存類別、物件鎖定、批次作業、複寫、標籤等功能來管理物件儲存。下列各節提供有關 Amazon S3 中可用儲存管理功能的詳細資訊。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱 什麼是 S3 Express One Zone？ 和 目錄值區。

主題

在 S3 儲存貯體中使用版本控制

在 Amazon S3 中使用版本控制是在相同儲存貯體中保留多個物件版本的一種方式。您可以使用 S3 版本控制功能來保留、擷取和恢復在儲存貯體中所存放每個物件的各個版本。透過版本控制，您就可以更

輕鬆地復原失誤的使用者動作和故障的應用程式。啟用儲存貯體的版本控制後，如果 Amazon S3 同時接收對同一物件的多個寫入要求，則會存放所有物件。

啟用版本控制的儲存貯體可讓您復原意外刪除或覆寫的物件。例如，如果您刪除某個物件，Amazon S3 會將刪除標記插入到該物件，而不是永久移除它。刪除標記會成為物件的目前版本。如果您覆寫物件，則會在儲存貯體中產生新的物件版本。您一律可以還原舊版本。如需詳細資訊，請參閱「刪除啟用版本控制功能之儲存貯體中的物件」。

儲存貯體預設停用 S3 版本控制，您必須明確啟用它。如需詳細資訊，請參閱「在儲存貯體上啟用版本控制」。

> **Note**
>
> - SOAP API 不支援 S3 版本控制。HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得。SOAP 不支援新的 Amazon S3 功能。
> - 標準 Amazon S3 費率適用於所存放和傳送物件的每個版本。物件的每個版本都是整個物件，而不只是與舊版本的差異。因此，如果您的所存放物件有三個版本，則會向您收取三個物件的費用。

未使用版本控制、已啟用版本控制和暫停版本控制的儲存貯體

儲存貯體可以是以下三種狀態之一：

- 未進行版本控制 (預設)
- 已啟用版本控制
- 已暫停版本控制

您可以在儲存貯體層級啟用和暫停版本控制。儲存貯體的版本控制一旦啟用，就無法再回復為未使用版本控制狀態。不過，您可以暫停儲存貯體的版本控制。

版本控制狀態會套用至該儲存貯體中的所有 (絕不會是一些) 物件。當您在儲存貯體中啟用版本控制時，所有新物件都會進行版本控制，並提供唯一的版本 ID。在啟用版本控制時已存在於儲存貯體中的物件，之後將始終進行版本控制，並在未來要求修改時提供唯一的版本 ID。注意下列事項：

- 在您設定版本控制狀態擁有版本 ID `null` 前，先儲存物件於儲存貯體中。當您啟用版本控制時，儲存貯體中的現有物件不會變更。Amazon S3 在未來請求中如何處理物件的方式會改變。如需詳細資訊，請參閱「使用已啟用版本控制之儲存貯體中的物件」。

- 儲存貯體擁有者 (或任何具有適當許可的使用者) 可以暫停版本控制，以停止產生物件版本。當您暫停版本控制時，儲存貯體中的現有物件不會變更。Amazon S3 在未來請求中如何處理物件的方式會改變。如需詳細資訊，請參閱「使用暫停版本控制之儲存貯體中的物件」。

搭配使用 S3 版本控制與 S3 生命週期

若要自訂您的資料保留方法及控制儲存體成本，請將物件版本控制搭配 S3 生命週期使用。如需詳細資訊，請參閱 管理儲存生命週期。如需使用 AWS Management Console、 AWS CLI、 AWS 開發套件或 REST API 建立 S3 生命週期組態的相關資訊，請參閱在值區上設定生命週期組態。

> **Important**
>
> 如果您在未使用版本控制的儲存貯體中有物件過期生命週期組態，而且想要在啟用版本控制時保持相同的永久刪除行為，您必須新增非目前的過期組態。非目前的過期生命週期組態，會管理啟用版本控制之儲存貯體中非目前物件版本的刪除。(已啟用版本控制的儲存貯體會維持一個目前的物件版本，以及零或多個非目前的物件版本。) 如需詳細資訊，請參閱「在值區上設定生命週期組態」。

如需使用 S3 版本控制的相關資訊，請參閱下列主題。

主題

- S3 版本控制的運作方式
- 在儲存貯體上啟用版本控制
- 設定 MFA Delete
- 使用已啟用版本控制之儲存貯體中的物件
- 使用暫停版本控制之儲存貯體中的物件

S3 版本控制的運作方式

您可以使用 S3 版本控制將物件的多個版本保留在一個儲存貯體中，以便可以還原意外刪除或覆寫的物件。例如，如果您將 S3 版本控制套用至儲存貯體，則會發生下列變更：

- 如果您刪除物件，而不是永久移除物件，Amazon S3 會插入刪除標記，這會變成目前物件版本。然後，您可以還原先前的版本。如需詳細資訊，請參閱 刪除啟用版本控制功能之儲存貯體中的物件。

- 如果您覆寫物件，則 Amazon S3 會在儲存貯體中新增物件版本。先前的版本會保留在儲存貯體中，並成為非目前版本。您可以還原先前的版本。

> **Note**
>
> 標準 Amazon S3 費率適用於所存放和傳送物件的每個版本。物件的每個版本都是整個物件，而不是與舊版本的差異。因此，如果您的所存放物件有三個版本，則會向您收取三個物件的費用。

您建立的每個 S3 儲存貯體都會有相關聯的 versioning 子資源。(如需詳細資訊，請參閱「儲存貯體組態選項」。) 儲存貯體預設為未使用版本控制，versioning 子資源會存放空的版本控制組態，如下所示。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</VersioningConfiguration>
```

若要啟用版本控制，您可以使用包含 `Enabled` 狀態的版本控制組態，將要求傳送給 Amazon S3。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

若要暫停版本控制，您必須將狀態值設為 `Suspended`。

> **Note**
>
> 您第一次在儲存貯體上啟用版本控制時，可能需要很短的時間來完全傳播變更。我們建議您在啟用版本控制之後等待 15 分鐘，然後再對儲存貯體中的物件發出寫入作業 (PUT 或 DELETE)。

值區擁有者和所有授權 AWS Identity and Access Management (IAM) 使用者都可以啟用版本控制。值區擁有者是建立值區的擁有者。 AWS 帳戶 如需許可的詳細資訊，請參閱「適用於 Amazon S3 的 Identity and Access Management」。

如需使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 REST API 啟用和停用 S3 版本控制的詳細資訊，請參閱the section called “在儲存貯體上啟用版本控制”。

主題

版本 ID

如果您啟用儲存貯體的版本控制，Amazon S3 會自動為要存放的物件產生唯一的版本 ID。例如，在一個儲存貯體中，兩個物件可以有相同的金鑰 (物件名稱)，但版本 ID 不同 (例如 `photo.gif` (111111 版) 和 `photo.gif` (121212 版))。
說明啟用版本控制的值區的圖表，該值區具有兩個具有相同索引鍵但版本 ID 不同的物件。

無論是否啟用 S3 版本控制，每個物件都有版本 ID。若尚未啟用 S3 版本控制，則 Amazon S3 會將版本 ID 的值設定為 `null`。如果啟用了 S3 版本控制，Amazon S3 會為物件指派一個版本 ID 值。此值會將該物件與相同金鑰的其他版本區分開來。

當您啟用現有儲存貯體的 S3 版本控制時，已存放在儲存貯體上的物件會保持不變。其版本 ID (`null`)、內容和許可都會保持不變。在啟用 S3 版本控制之後，新增至儲存貯體的每個物件都會取得版本 ID，這可將其與相同金鑰的其他版本區分開來。

只有 Amazon S3 會產生版本 ID，而且無法編輯它們。版本 ID 是 Unicode、UTF-8 編碼、可直接用為 URL，以及難解的字串，最長可達 1,024 個位元組。以下是範例：

```
3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo
```

> **Note**
>
> 為了簡單起見，本主題中的其他範例會使用較短的 ID。

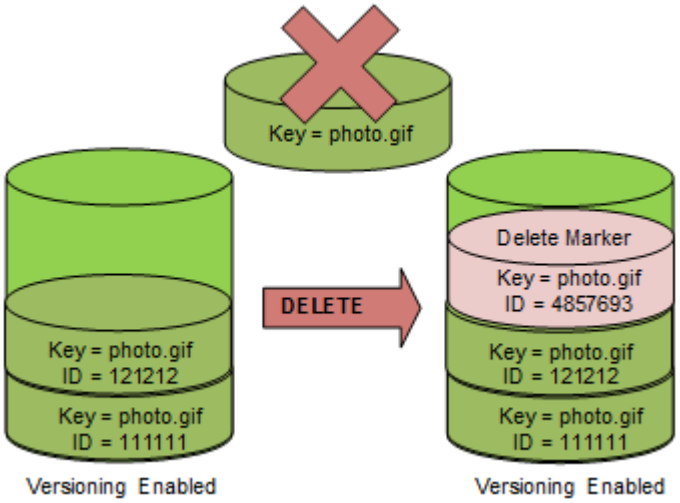
版本控制工作流程

當您將物件 PUT 到已啟用版本控制的儲存貯體時，並不會覆寫非目前的版本。如下圖所示，當新版的 `photo.gif` PUT 到已包含同名物件的儲存貯體時，會發生下列行為：

- 原始物件 (ID = 111111) 仍會留在儲存貯體中。
- Amazon S3 會產生新的版本 ID (121212)，並將此較新版本的物件新增至儲存貯體。

使用此功能時，如果物件遭到意外覆寫或刪除，您可以擷取物件的先前版本。

當您 DELETE 物件時，所有版本都會保留在儲存貯體中，且 Amazon S3 會插入刪除標記，如下圖所示。



刪除標記會成為物件的目前版本。根據預設，GET 要求會擷取最近存放的版本。當目前版本為刪除標記時執行 GET Object 請求，會傳回 404 Not Found 錯誤，如下圖所示。

但您可以藉由指定物件的版本 ID，來 GET 非目前版本的物件。在下圖中，您將 GET 特定的物件版本：111111。即使該物件版本並非目前的版本，Amazon S3 也會傳回該物件版本。

如需詳細資訊，請參閱 從啟用版本控制的儲存貯體擷取物件版本。

您可以指定要刪除的版本，來永久刪除物件。只有 Amazon S3 儲存貯體的擁有者或授權 IAM 使用者可以永久刪除版本。如果您的 DELETE 操作指定 versionId，則會永久刪除該物件版本，且 Amazon S3 不會插入刪除標記。

您可以將儲存貯體設定為啟用多重要素驗證 (MFA) 刪除，來提升安全性。當您針對儲存貯體啟用 MFA 刪除時，儲存貯體擁有者必須在任一請求中包含兩種身分驗證形式，才能刪除版本或變更儲存貯體的版本控制狀態。如需詳細資訊，請參閱 設定 MFA Delete。

何時建立物件的新版本？

只有在 PUT 新物件時才會建立物件的新版本。請注意，某些動作 (例如 CopyObject) 透過實作 PUT 操作來運作。

執行修改目前物件的某些動作不會建立新版本，因為它們不會 PUT 新物件。這包含變更物件上的標籤等動作。

> **Important**
>
> 當發現 Amazon S3 對啟用了 S3 版本控制之儲存貯體發出 PUT 或 DELETE 物件請求之後，接收 HTTP 503 (無法使用服務) 回應的數目明顯增加時，表示該儲存貯體中有一或多個物件可能擁有數百萬個版本。如需詳細資訊，請參閱 S3 版本控制區段中的 疑難排解。

在儲存貯體上啟用版本控制

您可以使用 Amazon S3 版本控制在單一儲存貯體中保留物件的多個版本。本節提供如何使用主控台、REST API、 AWS SDK 和 AWS Command Line Interface (AWS CLI) 在值區上啟用版本控制的範例。

> **Note**
>
> 如果您是第一次在值區上啟用版本控制，變更可能需要 15 分鐘才能完全傳播。我們建議您在啟用版本控制之後等待 15 分鐘，然後再對儲存貯體中的物件發出寫入作業 (PUT 或 DELETE)。在此轉換完成之前發出的寫入作業可能會套用至未建立版本化的物件。

如需 S3 版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。如需有關使用已啟用版本控制儲存貯體中物件的資訊，請參閱使用已啟用版本控制之儲存貯體中的物件。

若要進一步了解如何使用 S3 版本控制來保護資料，請參閱教學課程：使用 S3 版本控制、S3 物件鎖定和 S3 複寫，保護 Amazon S3 上的資料，以防意外刪除或發生應用程式錯誤。

您建立的每個 S3 儲存貯體都會有相關聯的 versioning 子資源。(如需詳細資訊，請參閱「儲存貯體組態選項」。) 儲存貯體預設為未使用版本控制，versioning 子資源會存放空的版本控制組態，如下所示。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</VersioningConfiguration>
```

若要啟用版本控制，您可以使用包含狀態的版本控制組態，將要求傳送給 Amazon S3。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

若要暫停版本控制，您必須將狀態值設為 Suspended。

儲存貯體擁有者和所有已獲授權的使用者都可以啟用版本控制。值區擁有者 AWS 帳戶 是建立儲存貯體 (根帳戶) 的擁有者。如需許可的詳細資訊，請參閱「適用於 Amazon S3 的 Identity and Access Management」。

以下各節提供有關使用主控台和 AWS SDK 啟用 S3 版本控制的詳細資訊。 AWS CLI

使用 S3 主控台

請按照下列步驟使用在 AWS Management Console S3 儲存貯體上啟用版本控制。

啟用或停用 S3 儲存貯體的版本控制

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇要啟用版本控制的儲存貯體名稱。
3. 選擇 Properties (屬性)。
4. 在 Bucket Versioning (儲存貯體版本控制) 底下，選擇 Edit (編輯)。
5. 選擇 Suspend (暫停) 或 Enable (啟用)，然後選擇 Save changes (儲存變更)。

> **Note**
>
> 您可以將 AWS 多因素驗證 (MFA) 與版本控制搭配使用。當您將 MFA 與版本控制搭配使用時，您 AWS 帳戶必須提供帳戶 MFA 裝置的存取金鑰和有效代碼，才能永久刪除物件版本，或暫停或重新啟用版本控制。
>
> 若要使用 MFA 與版本控制搭配，請啟用 `MFA Delete`。但是，您無法使用 AWS Management Console來啟用 `MFA Delete`。您必須使用 AWS Command Line Interface (AWS CLI) 或 API。如需詳細資訊，請參閱 設定 MFA Delete。

使用 AWS CLI

下列範例啟用 S3 儲存貯體上的版本控制。

```
aws s3api put-bucket-versioning --bucket example-s3-bucket1 --versioning-configuration
 Status=Enabled
```

下列範例會啟用儲存貯體上的 S3 版本控制和 多重要素驗證 (MFA) 刪除。

```
aws s3api put-bucket-versioning --bucket example-s3-bucket1 --versioning-configuration
 Status=Enabled,MFADelete=Enabled --mfa "SERIAL 123456"
```

> **Note**
>
> 使用 MFA Delete 需要已核准的實體或虛擬身分驗證裝置。如需在 Amazon S3 中使用 MFA Delete 的詳細資訊，請參閱「設定 MFA Delete」。

若要取得有關使用啟用版本化的更多資訊 AWS CLI，請參閱《指AWS CLI 令參考》put-bucket-versioning中的〈

使用 AWS 軟體開發套件

下列範例會啟用值區的版本控制，然後使用 AWS SDK for Java 和擷取版本控制狀態 AWS SDK for .NET。如需有關使用其他 AWS SDK 的詳細資訊，請參閱 AWS 開發人員中心。

.NET

如需有關設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using System;
using Amazon.S3;
using Amazon.S3.Model;

namespace s3.amazon.com.docsamples
{
    class BucketVersioningConfiguration
    {
        static string bucketName = "*** bucket name ***";

        public static void Main(string[] args)
        {
            using (var client = new AmazonS3Client(Amazon.RegionEndpoint.USEast1))
            {
                try
                {
```

```
                    EnableVersioningOnBucket(client);
                    string bucketVersioningStatus =
RetrieveBucketVersioningConfiguration(client);
                }
                catch (AmazonS3Exception amazonS3Exception)
                {
                    if (amazonS3Exception.ErrorCode != null &&
                        (amazonS3Exception.ErrorCode.Equals("InvalidAccessKeyId")
                        ||
                        amazonS3Exception.ErrorCode.Equals("InvalidSecurity")))
                    {
                        Console.WriteLine("Check the provided AWS Credentials.");
                        Console.WriteLine(
                        "To sign up for service, go to http://aws.amazon.com/s3");
                    }
                    else
                    {
                        Console.WriteLine(
                         "Error occurred. Message:'{0}' when listing objects",
                         amazonS3Exception.Message);
                    }
                }
            }

            Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }

        static void EnableVersioningOnBucket(IAmazonS3 client)
        {

                PutBucketVersioningRequest request = new PutBucketVersioningRequest
                {
                    BucketName = bucketName,
                    VersioningConfig = new S3BucketVersioningConfig
                    {
                        Status = VersionStatus.Enabled
                    }
                };

                PutBucketVersioningResponse response =
client.PutBucketVersioning(request);
        }
```

```
        static string RetrieveBucketVersioningConfiguration(IAmazonS3 client)
        {
                GetBucketVersioningRequest request = new GetBucketVersioningRequest
                {
                    BucketName = bucketName
                };

                GetBucketVersioningResponse response =
 client.GetBucketVersioning(request);
                return response.VersioningConfig.Status;
            }
    }
}
```

Java

如需如何建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.SetBucketVersioningConfigurationRequest;

public class BucketVersioningConfigurationExample {
    public static String bucketName = "*** bucket name ***";
    public static AmazonS3Client s3Client;

    public static void main(String[] args) throws IOException {
        s3Client = new AmazonS3Client(new ProfileCredentialsProvider());
        s3Client.setRegion(Region.getRegion(Regions.US_EAST_1));
        try {

            // 1. Enable versioning on the bucket.
         BucketVersioningConfiguration configuration =
           new BucketVersioningConfiguration().withStatus("Enabled");

   SetBucketVersioningConfigurationRequest setBucketVersioningConfigurationRequest
 =
```

```
     new SetBucketVersioningConfigurationRequest(bucketName,configuration);


 s3Client.setBucketVersioningConfiguration(setBucketVersioningConfigurationRequest);

   // 2. Get bucket versioning configuration information.
   BucketVersioningConfiguration conf =
 s3Client.getBucketVersioningConfiguration(bucketName);
    System.out.println("bucket versioning configuration status:    " +
 conf.getStatus());

        } catch (AmazonS3Exception amazonS3Exception) {
            System.out.format("An Amazon S3 error occurred. Exception: %s",
 amazonS3Exception.toString());
        } catch (Exception ex) {
            System.out.format("Exception: %s", ex.toString());
        }
    }
}
```

Python

以下 Python 程式碼範例會建立 Amazon S3 儲存貯體，使其能夠進行版本控制，並設定一個在 7 天後讓最新物件到期的生命週期。

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
```

```
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
```

```
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

設定 MFA Delete

在 Amazon S3 儲存貯體中使用 S3 版本控制時，您可以選擇將儲存貯體設定為啟用 MFA (多重因素認證) Delete，來增加額外的安全性。當您這樣做時，儲存貯體擁有者必須在任一要求中包含兩種身分驗證形式，才能刪除版本或變更儲存貯體的版本控制狀態。

MFA Delete 會要求額外的身分驗證，才能進行下列任一操作：

- 變更儲存貯體的版本控制狀態
- 永久刪除物件版本

MFA Delete 要求同時使用兩種形式的身分驗證：

- 安全憑證
- 核准的身分驗證設備上顯示的有效序號、空格和六位代碼組合。

因此，MFA Delete 可提供額外的安全性，例如在安全憑證洩露時。在某個使用者執行刪除動作時，MFA Delete 會要求啟動刪除動作的使用者證明其擁有實體 MFA 裝置和 MFA 代碼，從而為刪除動作新增另一層阻力和安全性，協助防止意外刪除儲存貯體。

若要識別已啟用 MFA 刪除的儲存貯體，您可以使用 Amazon S3 Storage Lens 指標。S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。如需詳細資訊，請參閱使用 S3 Storage Lens 評估儲存活動和用量。如需完整的指標清單，請參閱 S3 Storage Lens 指標詞彙表。

值區擁有者、建立值區 (root 帳戶) 的擁有者，以及所有授權的使用者都可以啟用版本控制。 AWS 帳戶 但只有儲存貯體擁有者 (根帳戶) 才能啟用 MFA Delete。如需詳細資訊，請參閱 AWS 安全性部落格上的保護 AWS 使用 MFA 的存取安全。

> **Note**
>
> 若要使用 MFA Delete 搭配額版本控制，請啟用 `MFA Delete`。但是，您無法使用 AWS Management Console啟用 `MFA Delete`。您必須使用 AWS Command Line Interface (AWS CLI) 或 API。
>
> 如需使用 MFA Delete 搭配版本控制的範例，請參閱 在儲存貯體上啟用版本控制 主題中的範例一節。
>
> 您無法搭配使用 MFA 刪除與生命週期組態。如需生命週期組態以及它們如何與其他組態互動的詳細資訊，請參閱 生命週期及其他儲存貯體組態。

若要啟用或停用 MFA Delete，您可以使用用來設定儲存貯體版本控制的相同 API。Amazon S3 會在存放儲存貯體版本控制狀態的相同 versioning 子資源中存放 MFA Delete 組態。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>VersioningState</Status>
  <MfaDelete>MfaDeleteState</MfaDelete>
</VersioningConfiguration>
```

若要使用 MFA Delete，您可以使用硬體或虛擬 MFA 裝置來產生身分驗證碼。下列範例顯示硬體裝置上所顯示的已產生身分驗證碼。

MFA Delete 與 MFA 保護的 API 存取功能針對不同情境提供保護。您可以在儲存貯體上設定 MFA Delete，確保無法意外刪除儲存貯體中的資料。存取機密 Amazon S3 資源時，可以使用 MFA 保護的 API 存取來強制另一個身分驗證因素 (MFA 碼)。您可能需要透過使用 MFA 所建立的暫時性憑證，才能完成任何對這些 Amazon S3 資源的操作。如需範例，請參閱 需要 MFA。

如需購買及啟用身分驗證裝置的詳細資訊，請參閱 Multi-Factor Authentication。

啟用 S3 版本控制和設定 MFA Delete

使用 AWS CLI

下列範例會啟用儲存貯體上的 S3 版本控制和 多重要素驗證 (MFA) 刪除。

```
aws s3api put-bucket-versioning --bucket example-s3-bucket1 --versioning-configuration
 Status=Enabled,MFADelete=Enabled --mfa "SERIAL 123456"
```

使用 REST API

如需使用 Amazon S3 REST API 指定 MFA 刪除的詳細資訊，請參閱 PutBucketVersioningAmazon 簡單儲存服務 API 參考。

使用已啟用版本控制之儲存貯體中的物件

設定版本控制狀態之前 Amazon S3 儲存貯體中所存放的物件版本 ID 為 null。當您啟用版本控制時，儲存貯體中的現有物件不會變更。Amazon S3 在未來請求中如何處理物件的方式會改變。

轉移物件版本

您可以定義物件的生命週期組態規則，而這些物件具有定義良好的生命週期可在物件生命週期的特定時間將物件版本轉移至 S3 Glacier Flexible Retrieval 儲存體類別。如需詳細資訊，請參閱 管理儲存生命週期。

本節主題說明啟用版本控制之儲存貯體中的各種物件操作。如需版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。

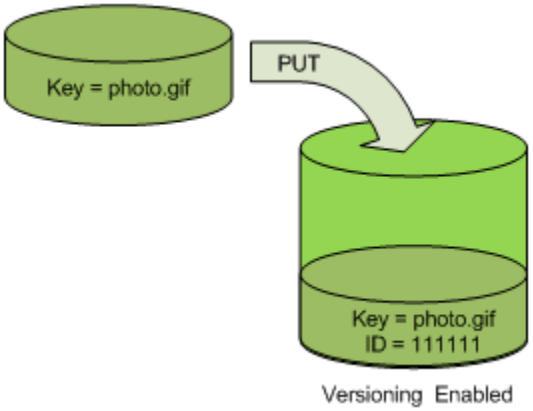
主題

- 將物件新增至啟用版本控制的儲存貯體
- 列出已啟用版本控制之儲存貯體中的物件
- 從啟用版本控制的儲存貯體擷取物件版本
- 刪除啟用版本控制功能之儲存貯體中的物件
- 設定已使用版本控制物件的許可

將物件新增至啟用版本控制的儲存貯體

當您在儲存貯體上啟用版本控制後，Amazon S3 便會自動將唯一的版本 ID 新增至儲存貯體中存放的每個物件 (使用 PUT、POST 或 CopyObject)。

下圖顯示將物件新增至啟用版本控制的儲存貯體時，Amazon S3 會將唯一的版本 ID 新增至物件。



> **Note**
>
> Amazon S3 所指派的版本 ID 值是 URL 安全 (可加入 URI 中)。

如需版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。您可以使用主控台、 AWS SDK 和 REST API，將物件版本新增至啟用版本控制的值區。

使用主控台

如需說明，請參閱「上傳物件」。

使用 AWS SDK

如需使用適用於 Java、.NET 和 PHP 的 AWS 開發套件上傳物件的範例，請參閱上傳物件。在非版本控制與已啟用版本控制的儲存貯體中上傳物件的範例相同；但是，如果是已啟用版本控制的儲存貯體，Amazon S3 會指派版本編號。否則，版本編號會是空值。

如需使用其他 AWS SDK 的相關資訊，請參閱AWS 開發人員中心。

使用 REST API

將物件新增至啟用版本控制的儲存貯體

1. 使用 PutBucketVersioning 要求啟用儲存貯體的版本控制。

如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 PutBucketVersioning。

2. 傳送 PUT、POST 或 CopyObject 要求，以將物件存放至儲存貯體。

將物件新增至已啟用版本控制的儲存貯體時，Amazon S3 會在 x-amz-version-id 回應標頭中傳回物件的版本 ID，如下例所示。

```
x-amz-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY
```

列出已啟用版本控制之儲存貯體中的物件

本節提供列出已啟用版本控制之儲存貯體中物件版本的範例。Amazon S3 會將物件版本資訊存放至與儲存貯體有關聯的 versions 子資源。如需詳細資訊，請參閱 儲存貯體組態選項。若要列出已啟用版本控制之儲存貯體，您需要 ListBucketVersions 許可。

使用 S3 主控台

請依照下列步驟使用 Amazon S3 主控台查看某個物件的不同版本。

查看物件的多個版本

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇包含該物件的儲存貯體名稱。
3. 若要查看儲存貯體中的物件版本清單，請選擇 Show versions (顯示版本) 切換開關。

   針對每個物件版本，主控台會顯示唯一版本 ID、物件版本建立日期與時間，以及其他屬性。(設定版本控制狀態之前儲存貯體中所存放的物件會有 null 的版本 ID。)

   若只要列出物件而不顯示版本，請選擇 List versions (列出版本) 切換開關。

您也可以在主控台的物件概觀面板中檢視、下載及刪除物件版本。如需詳細資訊，請參閱 在 Amazon S3 主控台中檢視物件概觀。

> **Note**
>
> 若要存取超過 300 個版本的物件版本，您必須使用 AWS CLI 或物件的 URL。

> **Important**
>
> 只有在刪除最新版 (目前版本) 的物件時，才能取消刪除物件。您無法取消刪除已刪除的舊版物件。如需詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。

使用 AWS 軟體開發套件

本節中的範例會示範如何從已啟用版本控制之儲存貯體中，擷取物件清單。除非您指定較小的數值，否則每個請求最多可回傳 1,000 個版本。如果儲存貯體含有比限制數量更多的版本，則您需要傳送一連串可擷取所有版本清單的要求。在「頁面」中傳回結果的過程稱為分頁。

為了解說分頁運作方式，這些範例會限制每一個物件版本的回應。在擷取結果的第一頁，每個範例都會檢查以確定是否截斷了版本清單。如果是，則該範例則會繼續擷取頁面，直到擷取到所有版本。

> **Note**
>
> 以下範例還適用於尚未啟用版本控制的儲存貯體，或者尚沒有獨立版本的物件。在這些情況下，Amazon S3 會傳回版本 ID 為 `null` 的物件清單。

如需使用其他 AWS SDK 的相關資訊，請參閱AWS 開發人員中心。

Java

如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListVersionsRequest;
import com.amazonaws.services.s3.model.S3VersionSummary;
import com.amazonaws.services.s3.model.VersionListing;

public class ListKeysVersioningEnabledBucket {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
```

```
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Retrieve the list of versions. If the bucket contains more versions
            // than the specified maximum number of results, Amazon S3 returns
            // one page of results per request.
            ListVersionsRequest request = new ListVersionsRequest()
                    .withBucketName(bucketName)
                    .withMaxResults(2);
            VersionListing versionListing = s3Client.listVersions(request);
            int numVersions = 0, numPages = 0;
            while (true) {
                numPages++;
                for (S3VersionSummary objectSummary :
versionListing.getVersionSummaries()) {
                    System.out.printf("Retrieved object %s, version %s\n",
                            objectSummary.getKey(),
                            objectSummary.getVersionId());
                    numVersions++;
                }
                // Check whether there are more pages of versions to retrieve. If
                // there are, retrieve them. Otherwise, exit the loop.
                if (versionListing.isTruncated()) {
                    versionListing =
s3Client.listNextBatchOfVersions(versionListing);
                } else {
                    break;
                }
            }
            System.out.println(numVersions + " object versions retrieved in " +
numPages + " pages");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
```

```
        }
    }
}
```

.NET

如需有關設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ListObjectsVersioningEnabledBucketTest
    {
        static string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main(string[] args)
        {
            s3Client = new AmazonS3Client(bucketRegion);
            GetObjectListWithAllVersionsAsync().Wait();
        }

        static async Task GetObjectListWithAllVersionsAsync()
        {
            try
            {
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    // You can optionally specify key name prefix in the request
                    // if you want list of object versions of a specific object.
```

```
                    // For this example we limit response to return list of 2
 versions.
                    MaxKeys = 2
                };
                do
                {
                    ListVersionsResponse response = await
 s3Client.ListVersionsAsync(request);
                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }

                    // If response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.KeyMarker = response.NextKeyMarker;
                        request.VersionIdMarker = response.NextVersionIdMarker;
                    }
                    else
                    {
                        request = null;
                    }
                } while (request != null);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

使用 REST API

Example — 列出儲存貯體中的所有物件版本

若要列出儲存貯體中物件的所有版本，請在 versions 要求中使用 GET Bucket 子資源。Amazon S3 最多能擷取 1,000 個物件，每個物件版本完全算為一個物件。因此，如果儲存貯體包含兩個金鑰 (例如，photo.gif 與 picture.jpg)，而且第一個金鑰有 990 個版本，第二個金鑰有 400 個版本，單一要求將會擷取 photo.gif 的全部 990 個版本，並且只會擷取 picture.jpg 的最新 10 個版本。

Amazon S3 會依存放順序來傳回物件版本，而最近存放的物件會先傳回。

在 GET Bucket 要求中，包含 versions 子資源。

```
GET /?versions HTTP/1.1
Host: bucketName.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 +0000
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

Example — 擷取索引鍵的所有版本

若要擷取物件版本子集，您可以使用 GET Bucket 的請求參數。如需詳細資訊，請參閱 GET Bucket。

1. 將 prefix 參數設為您想要擷取之物件的索引鍵。
2. 使用 GET Bucket 子資源與 versions 來傳送 prefix 要求。

   ```
   GET /?versions&prefix=objectName HTTP/1.1
   ```

Example — 使用字首擷取物件

下列範例會擷取其金鑰或開頭為 myObject 的物件。

```
GET /?versions&prefix=myObject HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

您可以使用其他要求參數來擷取物件之所有版本的子集。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 GET Bucket。

Example — 在截斷回應時擷取其他物件清單

如果可在 GET 要求中傳回的物件數目超過 max-keys 的值，則回應會包含 <isTruncated>true</isTruncated>，並包含滿足要求但未傳回的第一個金鑰 (在 NextKeyMarker 中) 與第一個版本 ID (在 NextVersionIdMarker 中)。您可以使用這些傳回的值作為後續要求中的開始位置，以擷取可滿足 GET 要求的其他物件。

使用下列程序可從儲存貯體中擷取可滿足原始 GET Bucket versions 要求的其他物件。如需 key-marker、version-id-marker、NextKeyMarker 和 NextVersionIdMarker 的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 GET Bucket。

以下是滿足原始 GET 請求的其他回應：

- 將 key-marker 的值設為前一個回應的 NextKeyMarker 中所傳回的金鑰。
- 將 version-id-marker 的值設為前一個回應的 NextVersionIdMarker 中所傳回的版本 ID。
- 使用 GET Bucket versions 與 key-marker 來傳送 version-id-marker 要求。

Example — 擷取開頭為所指定索引鍵與版本 ID 的物件

```
GET /?versions&key-marker=myObject&version-id-marker=298459348571 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

使用 AWS CLI

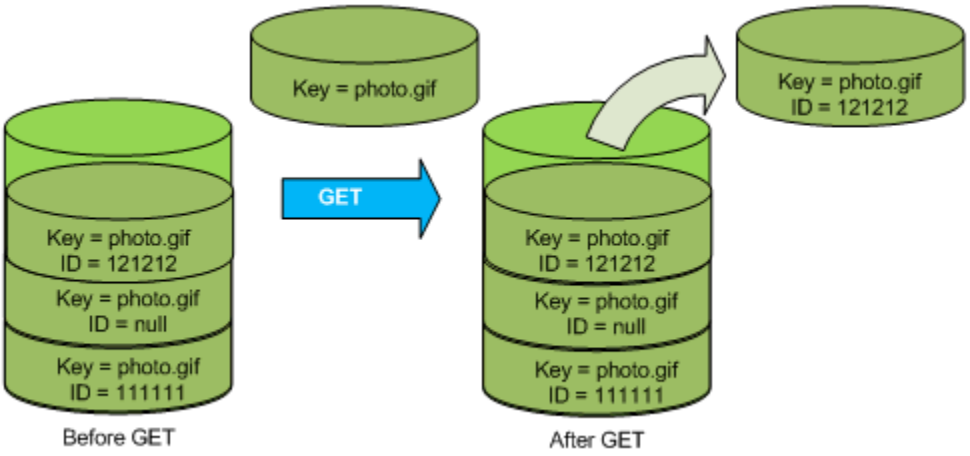
以下命令會傳回有關儲存貯體中所有版本的物件的中繼資料。

```
aws s3api list-object-versions --bucket example-s3-bucket1
```

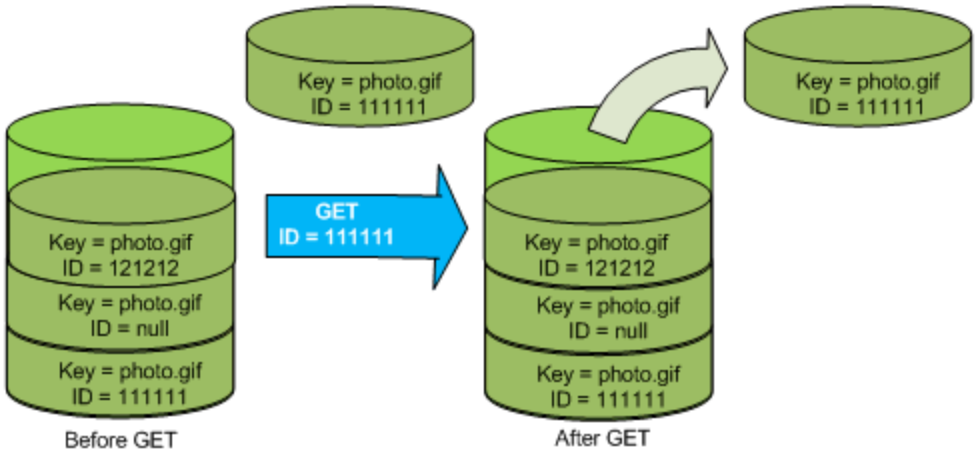
如需 list-object-versions 的詳細資訊，請參閱《AWS CLI 命令參考》中的 list-object-versions。

從啟用版本控制的儲存貯體擷取物件版本

在 Amazon S3 中使用版本控制是在相同儲存貯體中保留多個物件版本的一種方式。簡單 GET 要求會擷取物件的目前版本。下圖顯示 GET 如何傳回 photo.gif 物件的目前版本。



若要擷取特定版本，您必須指定其版本 ID。下圖顯示 `GET versionId` 要求如何擷取物件的指定版本 (不需要是目前版本)。



您可以使用主控台、 AWS 開發套件或 REST API 在 Amazon S3 中擷取物件版本。

> **Note**
>
> 若要存取超過 300 個版本的物件版本，您必須使用 AWS CLI 或物件的 URL。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇包含該物件的儲存貯體名稱。
3. 在 Objects (物件) 清單中，選擇物件的名稱。

4. 選擇 Versions (版本)。

   Amazon S3 會顯示該物件的所有版本。

5. 選取要擷取版本之 Version ID (版本 ID) 旁邊的核取方塊。
6. 選擇 Actions (動作)，選擇 Download (下載)，然後儲存物件。

您也可以在物件概觀面板中檢視、下載及刪除物件版本。如需詳細資訊，請參閱「在 Amazon S3 主控台中檢視物件概觀」。

> **Important**
>
> 只有在刪除最新版 (目前版本) 的物件時，才能取消刪除物件。您無法取消刪除已刪除的舊版物件。如需詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。

使用 AWS 軟體開發套件

上傳物件到未使用版本控制和啟用版本控制的儲存貯體中的範例相同。但是，對於啟用版本控制的儲存貯體，Amazon S3 會指派一個版本號碼。否則，版本編號會是空值。

如需使用適用於 Java、.NET 和 PHP 的 AWS 開發套件下載物件的範例，請參閱下載物件。

如需列出使用 .NET 和 Rust AWS 開發套件的物件版本的範例，請參閱列出 Amazon S3 儲存貯體中的物件版本。

使用 REST API

擷取特定物件版本

1. 將 `versionId` 設為您想要擷取之物件的版本 ID。
2. 傳送 `GET Object versionId` 要求。

Example — 擷取已使用版本控制的物件

下列要求會擷取 `L4kqtJlcpXroDTDmpUMLUo` 的版本 `my-image.jpg`。

```
GET /my-image.jpg?versionId=L4kqtJlcpXroDTDmpUMLUo HTTP/1.1
Host: bucket.s3.amazonaws.com
```

```
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

您可以只擷取物件 (而非內容) 的中繼資料。如需相關資訊，請參閱「the section called “擷取版本中繼資料”」。

如需有關還原舊版物件的資訊，請參閱 the section called “還原舊版本”。

擷取物件版本的中繼資料

如果您只想要擷取物件的中繼資料 (而非其內容)，則可以使用 HEAD 操作。根據預設，您會取得最新版本的中繼資料。若要擷取特定物件版本的中繼資料，請指定其版本 ID。

擷取物件版本的中繼資料

1. 將 `versionId` 設為您想要擷取其中繼資料之物件的版本 ID。
2. 傳送 `HEAD Object versionId` 要求。

Example — 擷取已使用版本控制之物件的中繼資料

下列請求會擷取 `my-image.jpg` 之版本 `3HL4kqCxf3vjVBH40Nrjfkd` 的中繼資料。

```
HEAD /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

下列顯示回應範例。

```
HTTP/1.1 200 OK
x-amz-id-2: ef8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC143432E5
x-amz-version-id: 3HL4kqtJlcpXroDTDmjVBH40Nrjfkd
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: text/plain
Connection: close
Server: AmazonS3
```

還原舊版本

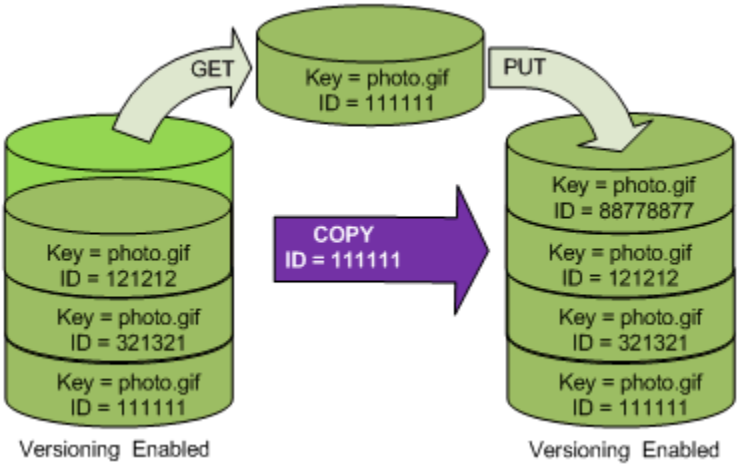
您可以使用版本控制來擷取舊版物件。有兩種方式可以達成：

- 將物件的舊版本複製至相同的儲存貯體。

  複製的物件會變成該物件目前的版本，並保留所有的物件版本。
- 永久刪除物件目前的版本。

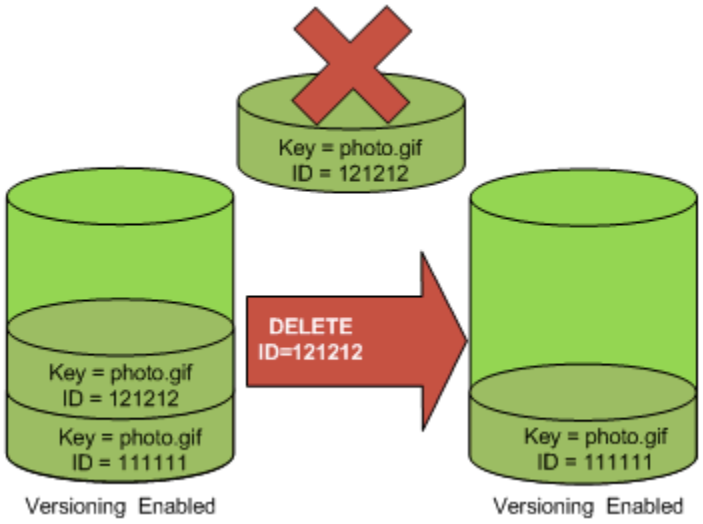
  當您刪除目前的物件版本時，實際上是將舊版本轉換成該物件目前的版本。

因為會保留所有物件版本，所以您可以將任何舊版本設為目前版本，方法是將物件的特定版本複製至相同的儲存貯體。在下圖中，來源物件 (ID = 111111) 會複製至相同的儲存貯體。Amazon S3 提供新的 ID (88778877)，這成為物件的目前版本。因此，儲存貯體同時具有原始物件版本 (111111) 和其複本 (88778877)。如需有關獲取先前版本然後上傳以使其成為目前版本的詳細資訊，請參閱從已啟用版本控制的儲存貯體中擷取物件版本和上傳物件。



隨後 GET 會擷取版本 88778877。

下圖顯示如何刪除將舊版本 (111111) 保留為目前物件之物件的目前版本 (121212)。有關刪除物件的詳細資訊，請參閱刪除單一物件。



隨後 GET 會擷取版本 111111。

> **Note**
>
> 若要以批次方式還原物件版本，您可以使用 CopyObject 操作。CopyObject 操作會複製資訊清單中指定的每個物件。不過，請注意物件不一定按照它們在資訊清單中出現的相同順序進行複製。對於使用版本控制的儲存貯體，如果保留當前/非當前版本順序很重要，則應該首先複製所有非當前版本。然後，在第一個任務完成後，在接下來的任務中複製目前版本。

還原先前物件版本

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇包含該物件的儲存貯體名稱。
3. 在 Objects (物件) 清單中，選擇物件的名稱。
4. 選擇 Versions (版本)。

   Amazon S3 會顯示該物件的所有版本。
5. 選取要擷取版本之 Version ID (版本 ID) 旁邊的核取方塊。
6. 選擇 Actions (動作)，選擇 Download (下載)，然後儲存物件。

您也可以在物件概觀面板中檢視、下載及刪除物件版本。如需詳細資訊，請參閱「在 Amazon S3 主控台中檢視物件概觀」。

> **Important**
>
> 只有在刪除最新版 (目前版本) 的物件時，才能取消刪除物件。您無法取消刪除已刪除的舊版物件。如需詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。

使用 AWS 軟體開發套件

如需使用其他 AWS SDK 的相關資訊，請參閱AWS 開發人員中心。

Python

透過刪除指定回復版本之後發生的所有版本，下列 Python 程式碼範例會還原已進行版本控制的物件的先前版本。

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
```

```
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
f"{object_key}."
        )
```

刪除啟用版本控制功能之儲存貯體中的物件

您可以隨時從 Amazon S3 儲存貯體中刪除物件版本。針對明確定義生命週期的物件，您還可以定義生命週期組態規則，以要求 Amazon S3 讓目前物件版本過期，或永久移除非目前的物件版本。當儲存貯體已啟用版本控制或暫停版本控制時，生命週期組態動作運作如下：

- 此 `Expiration` 動作會套用至目前物件版本。Amazon S3 並不會刪除目前物件版本，而是新增刪除標記，將目前版本保留為非目前版本，這之後會成為目前版本。
- `NoncurrentVersionExpiration` 動作會套用至非目前物件版本，而 Amazon S3 會永久移除這些物件版本。您無法復原永久移除的物件。

如需 S3 生命週期的詳細資訊，請參閱 管理儲存生命週期 和 S3 生命週期組態範例。

若要查看您的儲存貯體有多少個目前和非目前物件版本，您可以使用 Amazon S3 Storage Lens 指標。S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。如需詳細資訊，請參閱使用 S3 Storage Lens 最佳化儲存成本。如需完整的指標清單，請參閱 S3 Storage Lens 指標詞彙表。

> **Note**
>
> 一般 Amazon S3 費率適用於存放和傳輸物件的每個版本，包括非最新的物件版本。如需詳細資訊，請參閱 Simple Storage Service (Amazon S3) 定價。

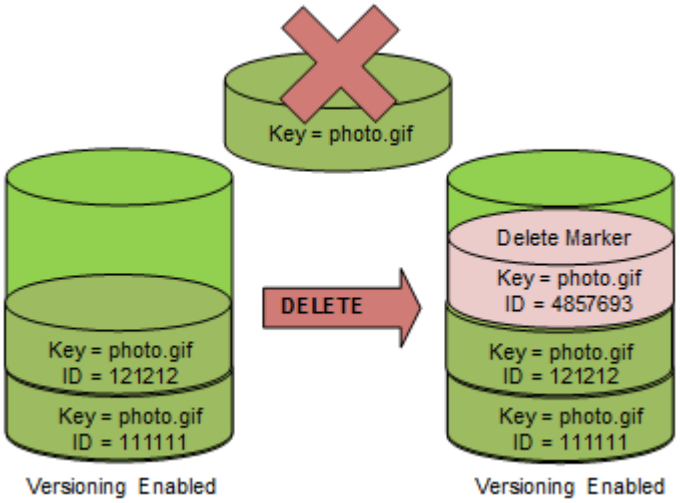
刪除要求使用案例

DELETE 要求的使用案例如下：

- 啟用版本控制時，簡單 DELETE 無法永久刪除物件。(簡單 DELETE 請求是指未指定版本 ID 的請求。) 相反地，Amazon S3 會在儲存貯體中插入刪除標記，而該刪除標記會成為具有新 ID 的目前物件版本。

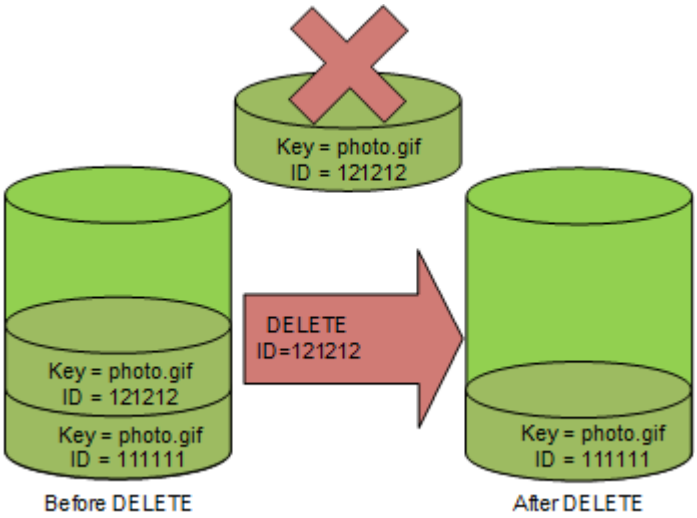
  當您嘗試其目前版本為刪除標記的 GET 物件時，Amazon S3 的行為就如同已刪除物件 (即使尚未清除也是一樣) 並傳回 404 錯誤。如需詳細資訊，請參閱 使用刪除標記。

  下圖顯示簡單 DELETE 不會實際移除指定的物件。相反地，Amazon S3 會插入刪除標記。



- 若要永久刪除已使用版本控制的物件，您必須使用 DELETE Object versionId。

  下圖顯示刪除所指定的物件版本會永久移除該物件。



刪除物件版本

您可以使用主控台、 AWS 開發套件、其餘 API 或. AWS Command Line Interface

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇包含該物件的儲存貯體名稱。
3. 在 Objects (物件) 清單中，選擇物件的名稱。
4. 選擇 Versions (版本)。

   Amazon S3 會顯示該物件的所有版本。
5. 選取要永久刪除版本之 Version ID (版本 ID) 旁邊的核取方塊。
6. 選擇 Delete (刪除)。
7. 在 Permanently delete objects? (永久刪除物件？) 中輸入 **`permanently delete`**。

   > **Warning**
   >
   > 永久刪除物件版本的動作無法復原。

8. 選擇 Delete objects (刪除物件)。

   Amazon S3 刪除物件版本。

使用 AWS 軟體開發套件

如需使用 Java、.NET 和 PHP AWS 開發套件刪除物件的範例，請參閱刪除 Amazon S3 物件。從未使用版本控制和啟用版本控制的儲存貯體中刪除物件的範例相同。但是，對於啟用版本控制的儲存貯體，Amazon S3 會指派一個版本號碼。否則，版本編號會是空值。

如需使用其他 AWS SDK 的相關資訊，請參閱AWS 開發人員中心。

Python

下列 Python 程式碼範例會透過刪除其所有版本，永久刪除已建立版本的物件。

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

使用 REST API

刪除物件的特定版本

- 在 DELETE 中，指定版本 ID。

Example — 刪除特定版本

下列範例會刪除 photo.gif 的 UIORUnfnd89493jJFJ 版本。

```
DELETE /photo.gif?versionId=UIORUnfnd89493jJFJ HTTP/1.1
```

```
Host: bucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
Content-Type: text/plain
Content-Length: 0
```

使用 AWS CLI

下列命令會將名為 test.txt 的物件從名為 *example-s3-bucket1* 的儲存貯體中刪除。若要移除特定版本的物件，您必須是儲存貯體擁有者，且必須使用版本 ID 子資源。

```
aws s3api delete-object --bucket example-s3-bucket1 --key test.txt --version-
id versionID
```

如需 `delete-object` 的詳細資訊，請參閱《AWS CLI 命令參考》中的 delete-object。

如需刪除物件版本的詳細資訊，請參閱下列主題：

- 使用刪除標記
- 移除刪除標記，使舊版本成為目前版本
- 刪除已啟用 MFA Delete 之儲存貯體中的物件

使用刪除標記

Amazon S3 中的刪除標記是簡單 DELETE 請求中所指定版本控制物件的預留位置 (或標記)。簡單 DELETE 請求是指未指定版本 ID 的請求。因為物件位在已啟用版本控制的儲存貯體中，所以未刪除物件。不過，刪除標記會讓 Amazon S3 顯示如同該物件已刪除的狀態。您可以在刪除標記上使用 Amazon S3 API DELETE 呼叫。若要這麼做，您必須使用具有適當權限的 AWS Identity and Access Management (IAM) 使用者或角色來提出DELETE要求。

刪除標記就像任何其他物件一樣，會有金鑰名稱 (或金鑰) 與版本 ID。不過，刪除標記與其他物件的差異如下：

- 刪除標記沒有任何相關聯的資料。
- 刪除標記不會與存取控制清單 (ACL) 值相關聯。
- 如果您發出刪除標記的 GET 請求，GET 請求不會擷取任何內容，因為刪除標記沒有資料。具體來說，當您的 GET 請求未指定 `versionId` 時，您會收到 404 (找不到) 錯誤。

刪除標記會累算在 Amazon S3 中因儲存而收取的最低費用中。刪除標記的儲存大小等於刪除標記的金鑰名稱大小。金鑰名稱是一連串的 Unicode 字元。金鑰名稱的 UTF-8 編碼會將 1 到 4 個位元組的儲存空間新增至名稱中每個字元的儲存貯體。刪除標記會存放在 S3 標準儲存類別中。

如果您想要了解您有多少個刪除標記以及存放它們的儲存類別，則可以使用 Amazon S3 Storage Lens。如需詳細資訊，請參閱 使用 Amazon S3 Storage Lens 評估儲存活動和使用量 及 Amazon S3 Storage Lens 指標詞彙表。

如需金鑰名稱的詳細資訊，請參閱「建立物件索引鍵名稱」。如需刪除刪除標記的資訊，請參閱「管理刪除標記」。

只有 Amazon S3 才能建立刪除標記，而且只要您對啟用或暫停版本控制的儲存貯體中的物件傳送 `DeleteObject` 要求，就會這麼做。`DELETE` 請求中所指定的物件實際上並不會刪除。相反地，刪除標記會成為物件的目前版本 物件的金鑰名稱 (或金鑰) 會成為刪除標記的金鑰。

如果您未在請求中指定 `versionId` 而取得物件，且其目前版本是刪除標記，則 Amazon S3 會回應以下內容：

- 404 (找不到) 錯誤
- 回應標頭 `x-amz-delete-marker: true`

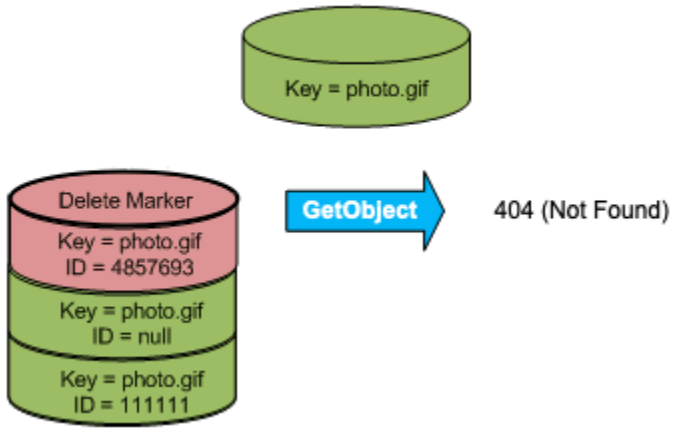
如果您在請求中指定 `versionId` 而取得物件，且指定的版本是刪除標記，則 Amazon S3 會回應以下內容：

- 405 (不允許的方法) 錯誤
- 回應標頭 `x-amz-delete-marker: true`
- 回應標頭 `Last-Modified: timestamp` (僅在使用HeadObject或 GetObjectAPI 作業時)

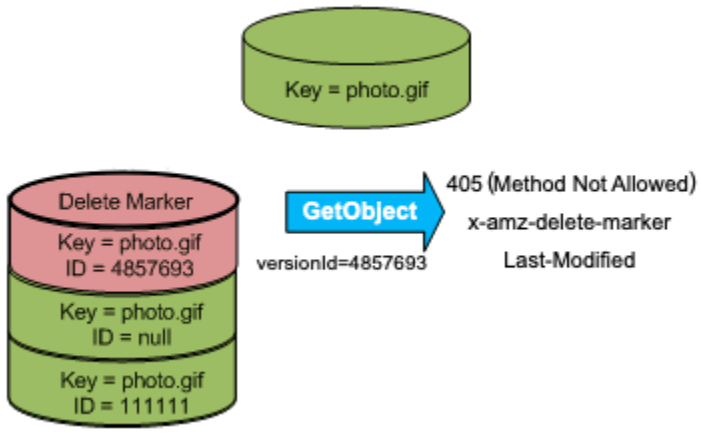
`x-amz-delete-marker: true` 回應標頭會告訴您所存取的物件是刪除標記。此回應標頭永遠不會傳回 `false`，因為當值為 `false` 時，物件的目前或指定版本不是刪除標記。

`Last-Modified` 回應標頭會提供刪除標記的建立時間。

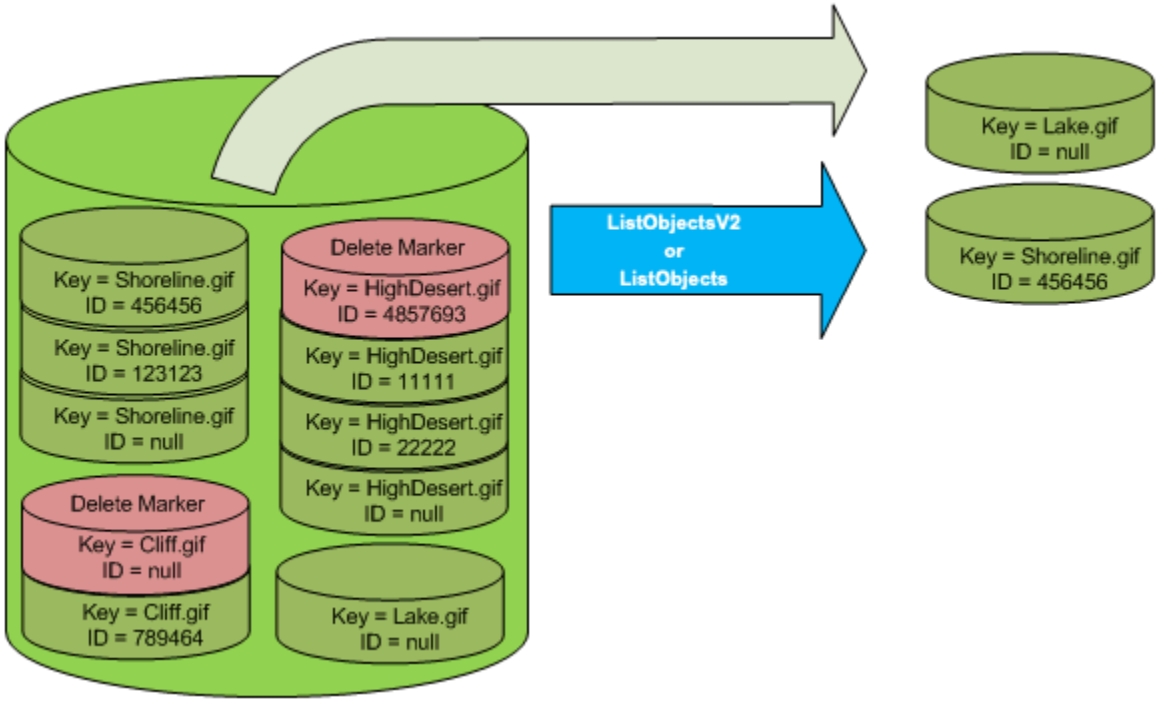
下圖顯示目前版本為刪除標記的物件上的 `GetObject` API 呼叫回應 404 (找不到) 錯誤，且回應標頭包含 `x-amz-delete-marker: true` 的情形。



如果您在請求中指定 `versionId` 來對物件進行 `GetObject` 呼叫，且指定的版本是刪除標記，Amazon S3 會回應 405 (方法不允許) 錯誤，且回應標頭包含 `x-amz-delete-marker: true` 和 `Last-Modified: timestamp`。



列出刪除標記 (以及其他版本的物件) 的唯一方式是在 `versions` 要求中使用 ListObjectVersions 子資源。下圖顯示 ListObjectsV2 或 ListObjects 請求不會傳回其目前版本是刪除標記的物件。



管理刪除標記

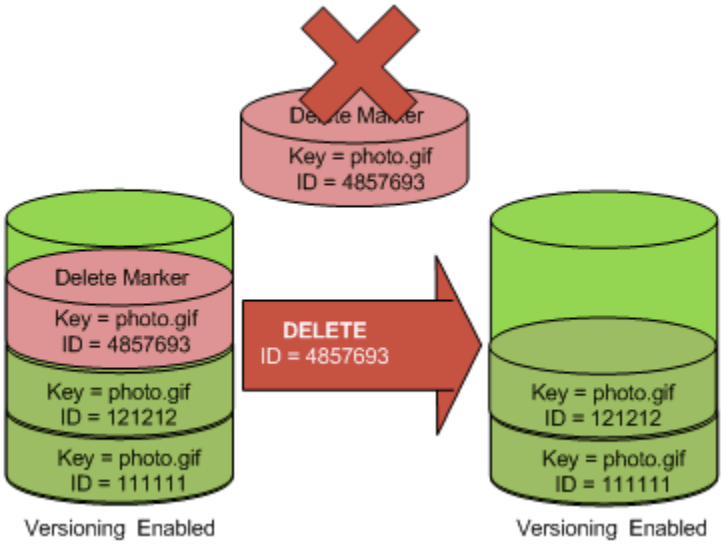
設定生命週期以自動清除過期的刪除標記

過期的物件刪除標記是刪除所有物件版本的標記，且僅保留單一刪除標記。如果生命週期組態設定為刪除目前版本，或明確設定 `ExpiredObjectDeleteMarker` 動作，則 Amazon S3 會移除過期物件的刪除標記。如需範例，請參閱 範例 7：移除過期物件刪除標記。

移除刪除標記，使舊版本成為目前版本

當您在已啟用版本控制的儲存貯體中刪除物件時，所有版本都會保留在儲存貯體中，而 Amazon S3 會為該物件建立刪除標記。若要取消刪除物件，您必須刪除此刪除標記。如需版本控制與刪除標記的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。

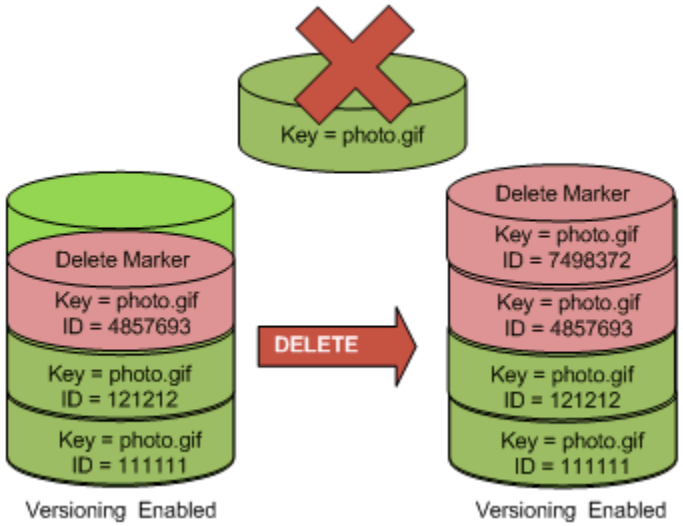
若要對刪除標記永久刪除，您必須在 `DeleteObject versionId` 要求中包含其版本 ID。下圖顯示 `DeleteObject versionId` 要求如何永久移除刪除標記。



移除刪除標記的效果是簡單的 GET 請求現在會擷取物件的目前版本 ID (121212)。

> **Note**
>
> 如果您使用 DeleteObject 請求，其中目前版本為刪除標記 (不指定刪除標記的版本 ID)，則 Amazon S3 不會刪去刪除標記，而是 PUTs 另一個刪除標記。

若要刪除具有 NULL 版本 ID 的刪除標記，必須在 DeleteObject 請求中作為版本 ID 傳遞 NULL。下圖顯示了如何作出沒有版本 ID 的簡單 DeleteObject 請求，其中目前版本為刪除標記，不會刪除任何內容，而是添加一個具有唯一版本 ID (7498372) 的額外刪除標記。



使用 S3 主控台

使用下列步驟，從 S3 儲存貯體中復原不是資料夾的已刪除物件，包括這些資料夾內的物件。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇您所需的儲存貯體名稱。
3. 若要查看儲存貯體中的物件版本清單，請選擇 List versions (列出版本) 切換開關。您將可以查看已刪除物件的刪除標記。
4. 若要取消刪除物件，您必須刪除刪除標記。選取刪除標記旁的核取方塊以復原物件，然後選擇 Delete (刪除)。
5. 在 Delete objects (刪除物件) 頁面上確認刪除。
   a. 對於 Permanently delete objects? (永久刪除物件？)，輸入 **`permanently delete`**。
   b. 選擇 Delete objects (刪除物件)。

> **Note**
>
> 您無法使用 Amazon S3 主控台來取消刪除資料夾。您必須使用 AWS CLI 或 SDK。例如，請參閱 AWS 知識中心的如何擷取在已啟用版本控制的儲存貯體中刪除的 Amazon S3 物件？。

使用 REST API

永久移除刪除標記

1. 將 `versionId` 設為您想要移除之刪除標記的版本 ID。
2. 傳送 `DELETE Object versionId` 要求。

Example — 移除刪除標記

下列範例會移除 `photo.gif` 版本 4857693 的刪除標記。

```
DELETE /photo.gif?versionId=4857693 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
```

```
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

當您刪去刪除標記時，Amazon S3 在回應中會包含下列各項：

```
204 NoContent
x-amz-version-id: versionID
x-amz-delete-marker: true
```

使用 AWS 軟體開發套件

如需使用其他 AWS SDK 的相關資訊，請參閱AWS開發人員中心。

Python

下列 Python 程式碼範例示範如何從物件移除刪除標記，從而使最新的非目前版本成為物件的目前版本。

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
```

```
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.", object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

刪除已啟用 MFA Delete 之儲存貯體中的物件

如果儲存貯體的版本控制組態已啟用 MFA Delete，則儲存貯體擁有者必須在要求中包含 `x-amz-mfa` 要求標頭，才能永久刪除物件版本或變更儲存貯體的版本控制狀態。包含 `x-amz-mfa` 的要求必須使用 HTTPS。

標頭的值是身分驗證裝置序號、空格與其上所顯示之身分驗證碼的組合。如果您未包含此要求標頭，則要求會失敗。

如需身分驗證裝置的詳細資訊，請參閱 Multi-factor Authentication。

Example — 刪除已啟用 MFA Delete 之儲存貯體中的物件

下列範例刪除 `my-image.jpg` (具有指定版本)，而其位於已設定啟用 MFA Delete 的儲存貯體中。

請注意 *`[SerialNumber] # [AuthenticationCode]`* 之間的空格。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 DeleteObject。

```
DELETE /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTPS/1.1
Host: bucketName.s3.amazonaws.com
x-amz-mfa: 20899872 301749
```

```
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

如需啟用 MFA Delete 的詳細資訊，請參閱「設定 MFA Delete」。

設定已使用版本控制物件的許可

Amazon S3 中物件的許可在版本層級進行設定。每個版本都有自己的物件擁有者。建立物件版本的是擁有者。 AWS 帳戶 因此，您可以為相同物件的不同版本設定不同的許可。要做到這一點，您必須在 PUT Object versionId acl 要求中指定您想要設定其許可之物件的版本 ID。如需使用 ACL 的詳細說明與指示，請參閱「適用於 Amazon S3 的 Identity and Access Management」。

Example — 設定物件版本的許可

下列請求會將被授予者 BucketOwner@amazon.com 的許可設為金鑰 my-image.jpg 上的 FULL_CONTROL，版本 ID 為 3HL4kqtJvjVBH40Nrjfkd。

```
PUT /my-image.jpg?acl&versionId=3HL4kqtJvjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
Content-Length: 124

<AccessControlPolicy>
  <Owner>
    <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>mtd@amazon.com</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
        <DisplayName>BucketOwner@amazon.com</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  </AccessControlPolicy>
```

同樣地，若要取得特定物件版本的許可，您必須在 GET Object versionId acl 要求中指定其版本 ID。您需要包含版本 ID；因為，根據預設，GET Object acl 會傳回物件之目前版本的許可。

Example — 擷取所指定物件版本的許可

在下列範例中，Amazon S3 傳回金鑰 my-image.jpg 版本 ID 為 DVBH40Nr8X8gUMLUo 的許可。

```
GET /my-image.jpg?versionId=DVBH40Nr8X8gUMLUo&acl HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU
```

如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 GetObjectAcl。

使用暫停版本控制之儲存貯體中的物件

在 Amazon S3 中，您可以暫停版本控制，以停止產生儲存貯體中相同物件的新版本。在您只希望在儲存貯體中保留一個物件版本時，您可能會這樣做。或者，可能因為您不想產生多個版本的費用。

當您暫停版本控制時，儲存貯體中的現有物件不會變更。Amazon S3 在未來請求中如何處理物件的方式會改變。本節主題說明暫停版本控制之儲存貯體中的各種物件操作，包括新增、擷取和刪除物件。

如需 S3 版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。如需擷取物件版本的詳細資訊，請參閱從啟用版本控制的儲存貯體擷取物件版本。

主題

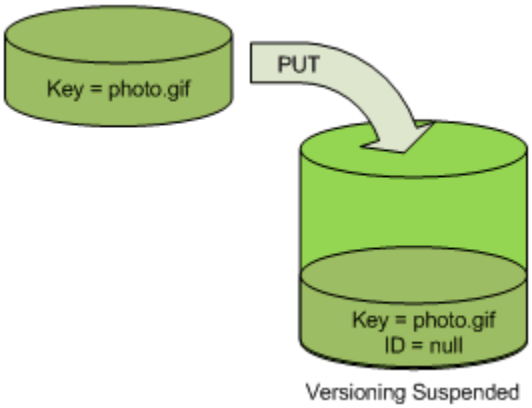
- 將物件新增至暫停版本控制的儲存貯體
- 從暫停版本控制的儲存貯體中擷取物件
- 刪除暫停版本控制之儲存貯體中的物件

將物件新增至暫停版本控制的儲存貯體

您可以在 Amazon S3 中將物件新增至已暫停版本控制的儲存貯體，以建立含 null 版本 ID 的物件，或覆寫任何具有相符版本 ID 的物件版本。

當您在儲存貯體上暫停版本控制後，Amazon S3 便會自動將 null 版本 ID 新增至此後儲存貯體中存放的每個後續物件 (使用 PUT、POST 或 CopyObject)。

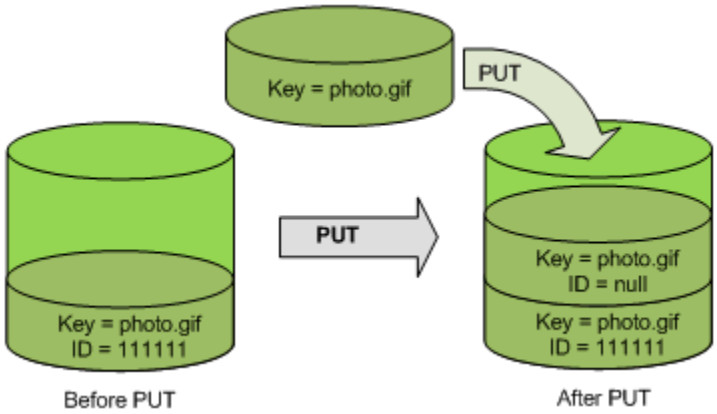
下圖顯示將物件新增至暫停版本控制的儲存貯體時，Amazon S3 如何將版本 ID null 新增至物件。



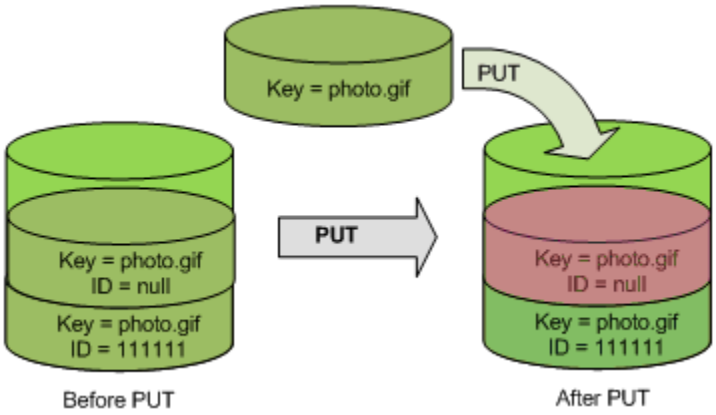
如果 null 版本已在儲存貯體中，而且您新增另一個具有相同金鑰的物件，則新增的物件會覆寫原始 null 版本。

如果儲存貯體中具有已啟用版本控制的物件，則您 PUT 的版本會變成物件的目前版本。下圖顯示如何將物件新增至包含已使用版本控制之物件的儲存貯體而不會覆寫已在儲存貯體中的物件。

在此情況下，111111 版本已在儲存貯體中。Amazon S3 會將 null 版本 ID 連接至所新增的物件，並將物件存放在儲存貯體中。不會覆寫 111111 版本。



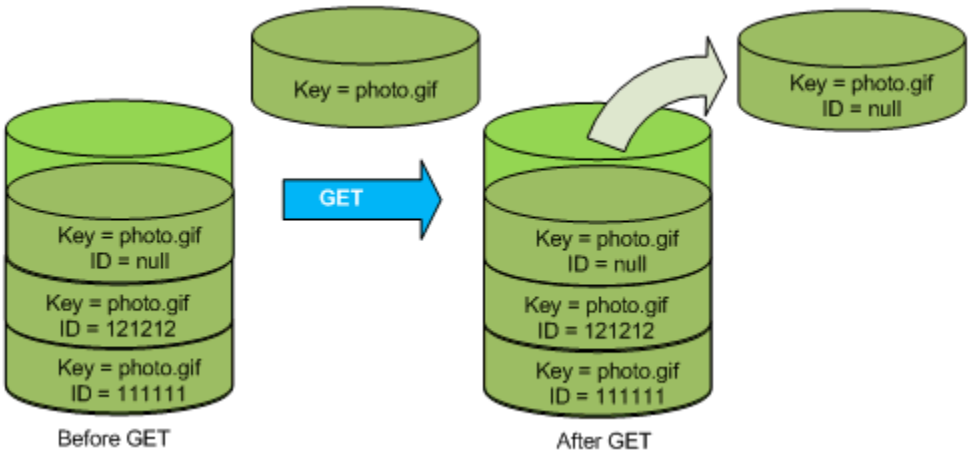
如果儲存貯體中已有 null 版本，則會覆寫 null 版本，如下圖所示。



雖然 PUT 前後之 null 版本的鍵與版本 ID (null) 相同，但是一開始存放在儲存貯體中 null 版本的內容會取代為 PUT 到儲存貯體中的物件內容。

從暫停版本控制的儲存貯體中擷取物件

不論是否已啟用儲存貯體的版本控制，GET Object 要求都會傳回物件的目前版本。下圖顯示簡單 GET 如何傳回物件的目前版本。



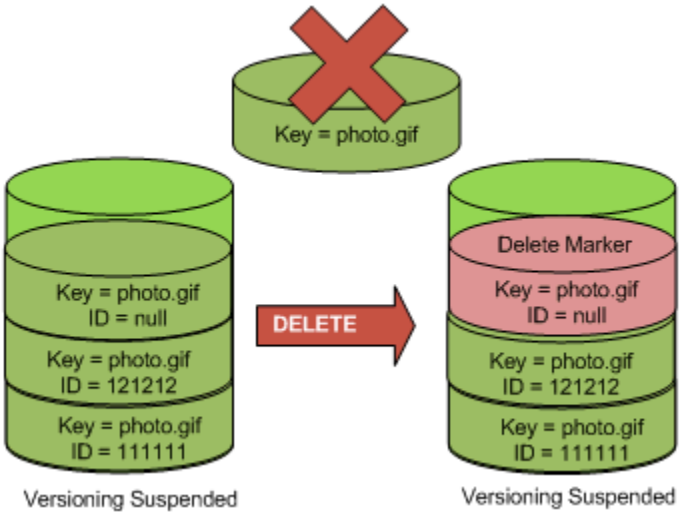
刪除暫停版本控制之儲存貯體中的物件

您可以刪除暫停版本控制之儲存貯體中的物件，以移除具有 null 版本 ID 的物件。

如果某個儲存貯體的版本控制處於暫停狀態，則 DELETE 要求：

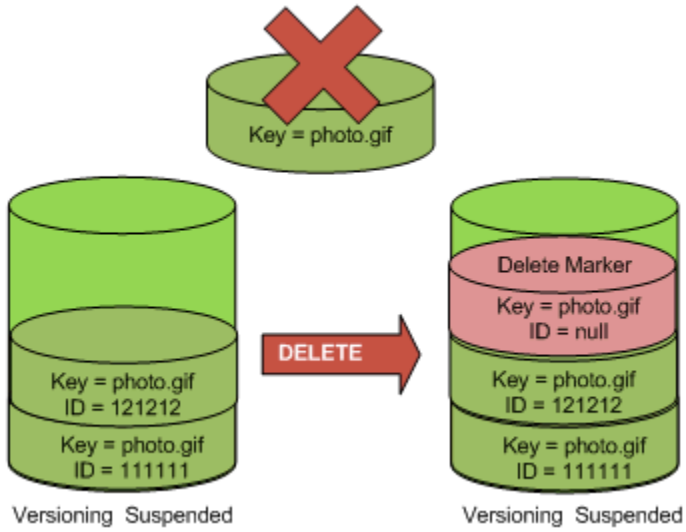
- 只能移除版本 ID 為 null 的物件。
- 如果儲存貯體中沒有物件的 null 版本，則不會移除任何項目。
- 將刪除標記插入至儲存貯體。

下圖顯示了一個簡單的 DELETE 如何刪除 null 版本。(簡單 DELETE 請求是指未指定版本 ID 的請求。) Amazon S3 在其位置插入一個版本 ID 為 null 的刪除標記。

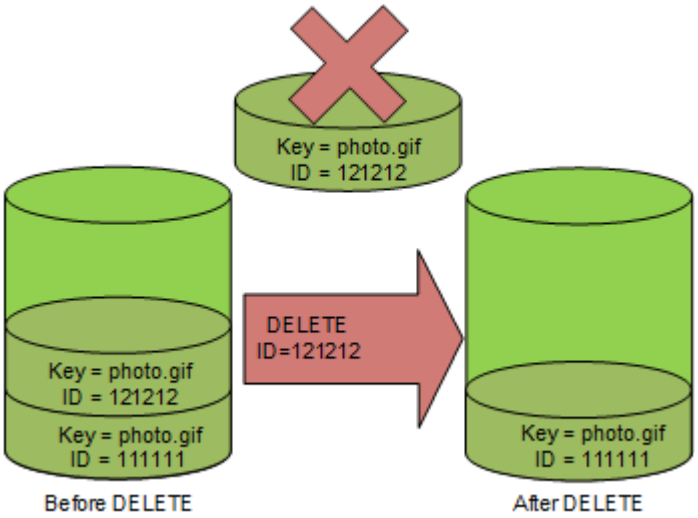


請記住，刪除標記沒有內容；因此，您會在以刪除標記來取代 null 版本時遺失其內容。

下圖顯示沒有 null 版本的儲存貯體。在此情況下，DELETE 不會移除任何項目；Amazon S3 只會插入刪除標記。



即使在暫停版本控制的儲存貯體中，儲存貯體擁有者也可以透過在 DELETE 請求中包含版本 ID 來永久刪除所指定的版本。下圖顯示刪除所指定的物件版本會永久移除該物件版本。只有儲存貯體擁有者才能刪除所指定的物件版本。



使用適用於 Amazon S3 的 AWS Backup

Amazon S3 與 AWS Backup 原生整合，這是一項以政策為基礎的全受管服務，您可以用來定義中央備份政策來保護 Amazon S3 資料。定義備份政策後，您可以將政策指派給 Amazon S3 資源，AWS Backup 即可自動建立 Amazon S3 備份，安全的將備份存放在您備份計畫中指定的加密文件庫當中。

您可以在使用適用於 Amazon S3 的 AWS Backup 時執行下列動作：

- 建立連續備份和定期備份。連續備份對於特定時間點還原、、 非常有用，而定期備份有助於滿足您長期資料-保留的需求。
- 集中設定備份政策，將備份排程與保留自動化。
- 將 Amazon S3 資料的備份還原至您指定的特定時間點。

連同 AWS Backup，您可以使用 S3 版本控制和 S3 複寫，協助從意外刪除中復原並執行自己的自我復原操作。

先決條件

您必須在儲存貯體上啟用 S3 版本控制，然後 AWS Backup 將其備份。

> **Note**
>
> 我們建議您為正在進行備份並啟用版本控制的儲存貯體設定生命週期過期規則。如果您沒有設定生命週期過期期限，則 Amazon S3 儲存成本可能會增加，因為 AWS Backup 會保留所有版本的 Amazon S3 資料。

入門

若要開始使用適用於 Amazon S3 的 AWS Backup，請參閱《AWS Backup 開發人員指南》中的建立 Amazon S3 備份。

法規與限制

若要了解限制，請參閱《AWS Backup 開發人員指南》中的建立 Amazon S3 備份。

使用封存的物件

若要降低不常存取物件的儲存成本，您可以封存這些物件。當您封存物件時，該物件會移入低成本的儲存空間，這表示您無法即時存取該物件。

雖然無法即時存取封存的物件，但您可以在幾分鐘或幾小時內還原它們 (根據儲存類別而定)。您可以使用 Amazon S3 主控台、S3 Batch 操作、REST API、 AWS 開發套件和 AWS Command Line Interface (AWS CLI) 來還原存檔物件。如需說明，請參閱還原已封存的物件。

下列儲存類別或層級的 Amazon S3 物件都會加以封存，且無法即時存取：

- S3 Glacier Flexible Retrieval 儲存體類別
- S3 Glacier Deep Archive 儲存體類別
- S3 Intelligent-Tiering Archive Access 層
- S3 Intelligent-Tiering Deep Archive Access 層

若要還原封存的物件，您必須執行以下操作：

- 對於 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 儲存類別中的物件，您必須啟動一個還原要求，然後等到物件的暫時複本可用。在建立還原物件的暫時複本時，該物件的儲存類別會保持不變。(HeadObject 或 GetObject API 操作請求會傳回 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 作為儲存類別。)
- 對於 S3 Intelligent-Tiering Archive Access 和 S3 Intelligent-Tiering Deep Archive Access 層中的物件，您必須啟動還原請求，然後等待物件移至經常存取層。

如需進一步了解所有 Amazon S3 儲存類別的異同，請參閱「使用 Amazon S3 儲存體方案」。如需 S3 Intelligent-Tiering 的詳細資訊，請參閱 the section called “S3 Intelligent-Tiering 的運作方式”。

從 S3 Glacier 還原物件

使用 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 時，Amazon S3 只會在指定的持續時間內還原物件的暫時複本。於該期間之後，會刪除還原的物件複本。您可以重新發出還原請求來修改還原複本的過期期間。在此情況下，Amazon S3 會以目前的時間為基準，更新過期期間。

> **Note**
>
> 當您從 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 還原封存的物件時，需要同時支付封存物件以及暫時復原之複本的費用。如需定價的資訊，請參閱 Amazon S3 定價。

從 S3 Intelligent-Tiering 還原物件

從 S3 Intelligent-Tiering Archive Access 層或 S3 Intelligent-Tiering Deep Archive Access 層還原物件時，物件會移回 S3 Intelligent-Tiering Frequent Access 層。如果物件連續 30 天未被存取，則會自動移至不常存取層。在連續至少 90 天未存取之後，物件會自動移至 S3 Intelligent-Tiering Archive Access 層。如果物件連續 180 天未被存取，則物件會移至 Deep Archive Access 層。

> **Note**
>
> 與 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 儲存類別不同，S3 Intelligent-Tiering 物件的還原要求不接受 `Days` 值。

搭配還原請求使用 S3 批次操作

若要使用單一請求還原超過一個 Amazon S3 物件，您可以使用 S3 批次操作。您可以為 S3 批次操作提供一份要進行操作的物件清單。S3 批次操作會呼叫相應的 API 操作來執行指定操作。單一批次作業任務可在包含數 EB 資料的數十億個物件上執行指定的操作。

還原時間

Amazon S3 計算還原物件複本之過期時間的方式，是將還原請求中所指定的天數，加上還原請求完成時的時間。Amazon S3 接著會將產生的時間四捨五入為國際標準時間 (UTC) 的隔天午夜。例如，假設還原物件複本是在 2012 年 10 月 15 日上午 10:30 UTC 建立，並且還原期間指定為 3 天。在此情況下，還原的複本會在 2012 年 10 月 19 日 00:00 UTC 時過期，Amazon S3 會在此時刪除物件複本。

還原任務完成所需的時間取決於您使用的封存儲存類別或儲存方案，以及指定的擷取選項：快速 (僅適用 S3 Glacier Flexible Retrieval 和 S3 Intelligent-Tiering Archive Access)、標準或大量。如需詳細資訊，請參閱 封存擷取選項。

您可以使用 Amazon S3 事件通知，以便在還原完成時收到通知。如需詳細資訊，請參閱 Amazon S3 事件通知。

主題

封存擷取選項

以下是在 Amazon S3 中還原已封存物件時可用的擷取選項：

- 快速 - 快速存取存放在 S3 Glacier Flexible Retrieval 儲存體類別或 S3 Intelligent-Tiering Archive Access 層中的資料。您在偶爾需要緊急請求一部分的存檔時，可以使用此選項。對於幾乎最大型的已封存物件 (250 MB 以上)，使用快速擷取所存取的資料，通常會在 1-5 分鐘內即可使用。

  > **Note**
  >
  > 快速擷取是一項高級功能，會按照快速要求和擷取費率收費。
  > 如需 Amazon S3 定價的相關資訊，請參閱 Amazon S3 定價。

  佈建容量有助於確保快速從 S3 Glacier Flexible Retrieval 擷取在需要時有可用的擷取容量。如需詳細資訊，請參閱 佈建的容量。

- 標準 - 您可在幾小時內存取任何已封存的物件。未指定擷取選項時，擷取請求的預設選項會是「標準」。對於存放在 S3 Glacier Flexible Retrieval 儲存類別或 S3 Intelligent-Tiering Archive Access 層中的物件，標準擷取一般會在 3 到 5 小時內完成。對於存放在 S3 Glacier Deep Archive 儲存體類別或 S3 Intelligent-Tiering Deep Archive 層中的物件，這些擷取通常會在 12 小時內完成。對於存放在 S3 Intelligent-Tiering 中的物件，可免費使用標準擷取。

  > **Note**
  >
  > - 對於存放在 S3 Glacier 彈性擷取儲存類別或 S3 智慧型分層存檔存取層中的物件，使用 S3 Batch 操作還原作業啟動的標準擷取通常在幾分鐘內開始，並在 3-5 小時內完成。

- 對於 S3 Glacier 深度存檔儲存類別或 S3 智慧型分層深度存檔存取層中的物件，使用 Batch 操作還原作業啟動的標準擷取通常會在 9 小時內開始，並在 12 小時內完成。

- 大量 - 使用 Amazon S3 Glacier 中成本最低的擷取選項存取您的資料。您可以使用「大量」擷取，以低廉的方式擷取大量資料 (甚至數 PB)。

  對於存放在 S3 Glacier 彈性擷取儲存類別或 S3 智慧型分層封存存取層中的物件，大量擷取通常會在 5 到 12 小時內完成。對於存放在 S3 Glacier Deep Archive 儲存類別或 S3 智慧型分層深度存檔存取層中的物件，這些擷取通常會在 48 小時內完成。

  對於存放在 S3 Glacier 彈性擷取或 S3 智慧型分層儲存類別中的物件，可免費進行大量擷取。

下表摘要說明封存擷取選項。如需定價的資訊，請參閱 Amazon S3 定價。

要進行`Expedited`，`Standard`或`Bulk`檢索，請將 RestoreObjectREST API 操作`Tier`請求中的請求元素設置為所需的選項或 AWS Command Line Interface （AWS CLI）或 AWS SDK 中的等效選項。若已購買佈建的容量，則所有快速擷取都會自動透過佈建的容量提供服務。

佈建的容量

佈建的容量有助於確保在需要時，從 S3 Glacier Flexible Retrieval 進行「快速」擷取有可用的擷取容量。每個容量單位都提供每 5 分鐘至少可以執行三次「快速」擷取，並提供最多每秒 150 MB (MBps) 的擷取輸送量。

如果您的工作負載需要高度穩定且可預測的資料子集即時存取，請考慮購買佈建的擷取容量。若沒有佈建的容量，在高需求量期間可能不會接受快速擷取。若您需要無論情況如何，皆能存取快速擷取，建議您購買佈建的擷取容量。

已佈建的容量單位會配置給 AWS 帳戶. 因此，「快速」資料擷取的請求者應該購買佈建的容量單位，而不是由儲存貯體擁有者購買。

您可以使用 Amazon S3 主控台、Amazon S3 Glacier 主控台、購買佈建的容量 REST API 作業、AWS 開發套件 AWS CLI或. 如需佈建的容量定價資訊，請參閱 Amazon S3 定價。

S3 Glacier 還原啟動請求速率

對存放在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存體類別的物件啟動還原請求時，會為您的 AWS 帳戶套用擷取請求配額。S3 Glacier 支援速率高達每秒 1,000 筆交易的還原請求。如果超過此速率，則會對有效請求節流或拒絕請求，且 Amazon S3 會傳回 `ThrottlingException` 錯誤。

您也可以選擇性地使用 S3 批次操作，搭配單一請求擷取存放在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 的大量物件。如需詳細資訊，請參閱 在 Amazon S3 物件上執行大規模批次操作。

還原已封存的物件

下列儲存類別或層級的 Amazon S3 物件都會加以封存，且無法即時存取：

- S3 Glacier Flexible Retrieval 儲存體類別
- S3 Glacier Deep Archive 儲存體類別
- S3 Intelligent-Tiering Archive Access 層
- S3 Intelligent-Tiering Deep Archive Access 層

在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別中存放的 Amazon S3 物件無法立即存取。若要存取這些儲存類別中的物件，您必須在指定持續時間 (天數) 內將物件的暫時複本還原至其 S3 儲存貯體。如果您想要物件的永久複本，請還原物件，然後在 Amazon S3 儲存貯體中建立其複本。Amazon S3 主控台不支援複製還原的物件。對於這種類型的複製操作，請使用 AWS Command Line Interface (AWS CLI)、 AWS SDK 或 REST API。除非您進行複製並變更其儲存類別，否則物件仍會儲存在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別中。如需有關使用這些儲存類別的資訊，請參閱 很少存取物件的儲存區類別。

若要存取 S3 Intelligent-Tiering Archive Access 和 Deep Archive Access 層中的物件，您必須啟動還原請求，然後等待物件移至 Frequent Access 層。當您從 Archive Access 層或 Deep Archive Access 層還原物件時，物件會移回不經常存取層。如需有關使用這些儲存類別的資訊，請參閱 存取模式會變更或不明的自動最佳化資料的儲存體方案。

如需封存物件的一般資訊，請參閱 使用封存的物件。

> **Note**
>
> - 當您從 S3 Glacier 彈性擷取或 S3 Glacier 深層存檔儲存類別還原存檔物件時，您需要為存檔物件和暫時還原的副本付費。
> - 從 S3 智慧型分層還原物件時，標準擷取或大量擷取無需支付擷取費用。
> - 對已還原之封存物件呼叫的後續還原要求會以GET請求的方式計費。如需定價的資訊，請參閱 Amazon S3 定價。

還原已封存的物件

您可以使用 Amazon S3 主控台、Amazon S3 REST API、 AWS 開發套件、 AWS Command Line Interface (AWS CLI) 或 S3 Batch 操作來還原存檔物件。

使用 S3 主控台

使用 Amazon S3 主控台還原物件

下列程序可用來還原已封存至 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別，或是 S3 Intelligent-Tiering Archive Access 或 Deep Archive Access 儲存層的物件。

還原封存物件

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇儲存貯體的名稱，其中包含您要還原的物件。
4. 在 Objects (物件) 清單中，選取要還原的一或多個物件，選擇 Actions (動作)，然後選擇 Initiate restore (啟動還原)。
5. 如果要從 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 還原，請在已還原副本可用天數方塊中輸入您希望封存資料可供存取的天數。
6. 在擷取方案中，執行下列任一項操作：

   - 選擇大量擷取或標準擷取，然後選擇啟動還原。
   - 選擇 Expedited retrieval (快速擷取) (僅適用於 S3 Glacier Flexible Retrieval 或 S3 Intelligent-Tiering Archive Access)。如果您要還原 S3 Glacier Flexible Retrieval 中的物件，可以選擇是否購買佈建的容量以進行「快速」擷取。如果您想要購買佈建的容量，請繼續執行下一個步驟。如果不要購買，請選擇啟動還原。

   > **Note**
   >
   > 來自 S3 Intelligent-Tiering Archive Access 和 Deep Archive Access 的物件會自動還原到 Frequent Access 層。

7. (選用) 如果您要還原 S3 Glacier Flexible Retrieval 中的物件，且您選擇了快速擷取，則可以選擇是否購買佈建的容量。佈建的容量僅適用於 S3 Glacier Flexible Retrieval 中的物件。如果您已有佈建的容量，請選擇啟動還原，以開始進行佈建的擷取。

如果您有佈建的容量，您所有的快速擷取都會透過佈建的容量提供服務。如需詳細資訊，請參閱佈建的容量。

- 如果您沒有佈建的容量，也不希望購買，請選擇啟動還原。
- 如果您沒有佈建的容量，但想要購買佈建的容量單位 (PCU)，請選擇購買 PCU。在購買 PCU 對話方塊中，選擇您要購買的 PCU 數量，確認購買，然後選擇購買 PCU。當您收到已成功購買訊息時，請選擇啟動還原以開始進行佈建的擷取。

使用 AWS CLI

從 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 還原物件

下列範例使用 `restore-object` 命令還原儲存貯體 *example-s3-bucket* 中的物件 *dir1/example.obj*，為期 25 天。

```
aws s3api restore-object --bucket example-s3-bucket --key dir1/example.obj --restore-request '{"Days":25,"GlacierJobParameters":{"Tier":"Standard"}}'
```

如果範例中使用的 JSON 語法導致 Windows 用戶端發生錯誤，請使用下列語法取代還原要求：

```
--restore-request Days=25,GlacierJobParameters={"Tier"="Standard"}
```

從 S3 Intelligent-Tiering Archive Access 和 Deep Archive Access 還原物件

下列範例使用 `restore-object` 命令將儲存貯體 *example-s3-bucket* 中的物件 *dir1/example.obj* 還原至 Frequent Access 層。

```
aws s3api restore-object --bucket example-s3-bucket --key dir1/example.obj --restore-request '{}'
```

> **Note**
>
> 與 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 儲存類別不同，S3 Intelligent-Tiering 物件的還原要求不接受 `Days` 值。

監控還原狀態

若要監控 `restore-object` 請求的狀態，請使用下列 `head-object` 命令：

```
aws s3api head-object --bucket example-s3-bucket --key dir1/example.obj
```

如需詳細資訊，請參閱 AWS CLI 命令參考中的 restore-object。

使用 REST API

Amazon S3 提供了 API 操作來讓您啟動已封存物件的還原操作。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 RestoreObject。

使用 AWS 軟體開發套件

如需如何使用 AWS 開發套件還原 S3 Glacier 彈性擷取或 S3 Glacier 深度存檔中的存檔物件的範例，請參閱搭RestoreObject配 AWS 開發套件或 CLI 使用。

使用 S3 批次操作

若要使用單一請求還原多個封存的物件，您可以使用 S3 批次操作。您可以為 S3 批次操作提供一份要進行操作的物件清單。S3 批次操作會呼叫相應的 API 操作來執行指定操作。單一批次作業任務可在包含數 EB 資料的數十億個物件上執行指定的操作。

若要建立批次操作任務，您必須擁有清單檔案，且當中只包含您要還原的物件。您可以使用 S3 庫存建立清單檔案，也可以提供包含必要資訊的 CSV 檔案。如需詳細資訊，請參閱 the section called “指定資訊清單”。

在建立和執行 S3 批次操作任務之前，您必須將許可授予 Amazon S3，以便代表您執行 S3 批次操作。如要了解必要的許可，請參閱 the section called “授予許可”。

> **Note**
>
> 批次操作任務可以在 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 儲存類別物件，或者 S3 Intelligent-Tiering Archive Access 和 Deep Archive Access 儲存層物件上進行。批次操作無法在同一任務中同時對兩種類型的封存物件進行操作。若要還原這兩種類型的物件，您必須建立單獨批次操作任務。
>
> 如需使用批次操作來還原封存物件的詳細資訊，請參閱 the section called “還原物件”。

若要建立 S3 啟動還原物件批次操作任務

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Batch Operations (批次操作)。

3. 選擇建立作業。
4. 針對 AWS 區域，選擇要在其中建立任務的區域。
5. 在資訊清單格式下，選擇要使用的清單檔案類型。

   - 如果您選擇 S3 庫存報告，請在 CSV 格式的庫存報告中輸入 Amazon S3 所產生 `manifest.json` 物件的路徑。如果您要使用的清單檔案版本並非最新版，請輸入 `manifest.json` 物件的版本 ID。
   - 如果您選擇 CSV，請輸入 CSV 格式資訊清單物件的路徑。資訊清單物件必須遵循主控台中所描述的格式。如果您要使用的版本並非最新版，可以選擇包含清單檔案物件的版本 ID。
6. 選擇下一步。
7. 在作業區段中，選擇還原。
8. 在還原區段中，針對還原來源選擇 Glacier Flexible Retrieval 或 Glacier Deep Archive 或 Intelligent-Tiering Archive Access 層或 Deep Archive Access 層。

   如果您選擇 Glacier Flexible Retrieval 或 Glacier Deep Archive，請輸入代表已還原副本可用天數的數字。

   針對擷取方案選擇您要使用的層級。
9. 選擇下一步。
10. 在設定其他選項頁面上，填寫下列區段：

    - 在其他選項區段中，提供任務的說明，並指定任務的優先順序編號。數字越大表示優先順序越高。如需詳細資訊，請參閱 the section called “指派任務優先順序”。
    - 在完成報告區段中，選取「批次操作」是否應建立完成報告。如需完成報告的詳細資訊，請參閱 the section called “完成報告”。
    - 在許可區段中，您必須對 Amazon S3 授予許可，以便代表您執行批次操作。如要了解必要的許可，請參閱 the section called “授予許可”。
    - (選用) 在任務標籤區段中，新增鍵值對形式的標籤。如需詳細資訊，請參閱 the section called “使用標籤”。

    完成時，選擇下一步。
11. 請確認 Review (檢閱) 頁面上的設定。如需變更，請選擇 Previous (上一步)。否則選擇建立任務。

如需批次操作的詳細資訊，請參閱 使用批次操作還原物件 和 建立 S3 批次操作任務

檢查還原狀態和到期日

您可以使用 Amazon S3 主控台、Amazon S3 事件通知或 Amazon S3 REST API 來檢查還原請求的 AWS CLI狀態或到期日。

> **Note**
>
> 從 S3 Glacier 彈性擷取和 S3 Glacier Deep Archive 儲存類別還原的物件只會存放您指定的天數。下列程序會傳回這些複本的到期日。
> 從 S3 智慧型分層封存存取和深度封存存取儲存層還原的物件沒有到期日期，而是會移回頻繁存取層。

使用 S3 主控台

在 Amazon S3 主控台中查看物件的還原狀態和到期日期

1. 前往 https://console.aws.amazon.com/s3/ 開啟的 Amazon Simple Storage Service (Amazon S3) 主控台。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在儲存貯體清單中，選擇包含您要還原之物件的儲存貯體名稱。
4. 在物件清單中，選取您要還原的物件。物件的詳細資訊頁面隨即出現。
   - 如果還原未完成，您會在頁面頂端看見一個區段，指出還原進行中。
   - 如果還原已完成，您會在頁面頂端看見一個區段，指出還原完成。如果您要從 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 還原，此區段也會顯示還原過期日期。Amazon S3 會在此日期移除封存物件的還原複本。

使用 Amazon S3 事件通知

使用具有 Amazon S3 事件通知功能的`s3:ObjectRestore:Completed`動作，即可收到物件還原完成的通知。如需啟用事件通知的詳細資訊，請參閱使用 Amazon SQS、Amazon SNS 和 AWS Lambda啟用通知。如需各種`ObjectRestore`事件類型的詳細資訊，請參閱the section called “SQS、SNS 和 Lambda 支援的事件類型”。

使用 AWS CLI

檢查物件的還原狀態和到期日 AWS CLI

下列範例使用 head-object 命令檢視儲存貯體 *example-s3-bucket* 中物件 *dir1/example.obj* 的中繼資料。當您對要還原的物件執行此命令時，Amazon S3 會傳回還原是否正在進行，以及 (如適用) 到期日期。

```
aws s3api head-object --bucket example-s3-bucket --key dir1/example.obj
```

預期的輸出（正在恢復）：

```
{
    "Restore": "ongoing-request=\"true\"",
    "LastModified": "2020-06-16T21:55:22+00:00",
    "ContentLength": 405,
    "ETag": "\"b662d79adeb7c8d787ea7eafb9ef6207\"",
    "VersionId": "wbYaE2vtOV0iIBXrOqGAJt3fP1cHB8Wi",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {},
    "StorageClass": "GLACIER"
}
```

預期的輸出（恢復已完成）：

```
{
    "Restore": "ongoing-request=\"false\", expiry-date=\"Wed, 12 Aug 2020 00:00:00 GMT
\"",
    "LastModified": "2020-06-16T21:55:22+00:00",
    "ContentLength": 405,
    "ETag": "\"b662d79adeb7c8d787ea7eafb9ef6207\"",
    "VersionId": "wbYaE2vtOV0iIBXrOqGAJt3fP1cHB8Wi",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {},
    "StorageClass": "GLACIER"
}
```

若要取得更多資訊head-object，請參閱《AWS CLI 指令參考》head-object中的〈〉。

使用 REST API

Amazon S3 提供了 API 操作讓您擷取物件中繼資料。若要使用 REST API 查看已封存物件的還原狀態和到期日期，請參閱《Amazon Simple Storage Service API 參考》中的 HeadObject。

升級進行中還原的速度

在還原進行期間，您可以升級還原的速度。

將正在進行的還原升級至更快的方案

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇儲存貯體的名稱，其中包含您要還原的物件。
4. 在物件清單中，選取您要還原的物件。物件的詳細資訊頁面隨即出現。在物件的詳細資訊頁面上，選擇升級擷取層級。如需有關檢查物件還原狀態的資訊，請參閱 檢查還原狀態和到期日。
5. 選擇您想要升級的目標層級，然後選擇啟動還原。

使用 S3 物件鎖定

S3 物件鎖定可協助在一段固定時間內或無限期避免刪除或覆寫 Amazon S3 物件。物件鎖定使用 write-once-read-many(WORM) 模型來儲存物件。您可以使用「物件鎖定」來協助滿足需要 WORM 儲存的法規要求，或是新增另一層防護來防止物件變更或刪除。

> **Note**
>
> S3 物件鎖定經 Cohasset Associates 評定，可用於符合 SEC 17a-4、CFTC 和 FINRA 法規的環境。如需物件鎖定與上述法規有何相關性的詳細資訊，請參閱 Cohasset Associates Compliance Assessment。

物件鎖定提供兩種管理物件保留的方式： 保留期和法務保存。物件版本可以有保留期和/或法務保存。

- 保留期：保留期會指定一段讓物件保持鎖定狀態的固定期間。您可以為個別物件設定唯一的保留期間。此外，您可以在 S3 儲存貯體上設定預設保留期。您也可以使用儲存貯體政策中的`s3:object-lock-remaining-retention-days`條件鍵來限制允許的最小和最長保留期間。這可協助您建立保留期間範圍，並限制可能短於或長於此範圍的保留期間。
- 法務保存：法務保存提供與保留期相同的保護，但沒有過期日期。因此，在您明確移除法務保存之前，它都會持續保留。法律訴訟保留獨立於保留期間，且會置於個別物件版本上。

物件鎖定只能在已啟用 S3 版本控制的儲存貯體中使用。當您鎖定物件版本時，Amazon S3 會將鎖定資訊存放在該物件版本的中繼資料中。當您在物件上設定保留期或法務保存時，僅會保護請求中指定的

版本。保留期間和法律訴訟保留不會阻止建立物件的新版本，也不會刪除要新增至物件頂端的標記。如需 S3 版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。

若您將物件放入儲存貯體，而當中已包含具備相同物件索引鍵名稱的現有受保護物件，則 Amazon S3 會建立該物件的新版本。現有的受保護版本物件則會根據其保留組態，而保持鎖定狀態。

S3 物件鎖定的運作方式

主題

保留期

保留期可在一段固定期間保護物件版本。當您為物件版本設定保留期時，Amazon S3 會將時間戳記存放在物件版本的中繼資料內，以指出保留期何時過期。保留期過後，就可以覆寫或刪除物件版本。

您可以對個別物件版本或儲存貯體的屬性明確設定保留期，以便自動套用至儲存貯體中的所有物件。當您明確地將保留期套用至物件版本時，您可以為物件版本指定「保留截止日期」。Amazon S3 會將此日期儲存在物件版本的中繼資料中。

您也可以在儲存貯體的屬性中設定保留期。當您在儲存貯體上設定保留期時，可以指定一段以天或年為單位的時間，代表要保護放置在儲存貯體中的每個物件版本的期間。當您將物件放入儲存貯體時，Amazon S3 會將指定的期間與物件版本的建立時間戳記相加，以計算出物件版本的保留截止日期。物件版本即可受到保護，完全就像您明確使用保留期對物件版本設定個別鎖定一樣。

> **Note**
>
> 當您對儲存貯體中具有明確的個別保留模式和保留期的物件版本執行 PUT 時，物件版本的個別物件鎖定設定會覆寫任何儲存貯體屬性保留設定。

如同所有其他的物件鎖定設定一樣，保留期也可套用至個別的物件版本。單一物件的不同版本可以擁有不同的保留模式和保留期。

例如，假設您有一個保留期為 15 天到 30 天的物件，而且您將同名且保留期為 60 天的物件 PUT 到 Amazon S3。在此情況下，PUT 請求會成功，而且 Amazon S3 會建立新版本的物件，且保留期為 60 天。舊版本仍會維持原本的保留期，並在 15 天後變成可供刪除的狀態。

對物件版本套用保留設定之後，可以延長保留期。若要這麼做，請針對物件版本提交新的物件鎖定請求，並將保留截止日期設為晚於目前為物件版本設定的日期。Amazon S3 會以新的更長期間取代現有的保留期。只要使用者具備設定物件保留期的許可，就可以延長物件版本的保留期。若要設定保留期，您必須具備 `s3:PutObjectRetention` 許可。

在物件或 S3 儲存貯體上設定保留期時，必須選取兩種保留模式之一：合規或控管。

保留模式

S3 物件鎖定提供了兩種保留模式，可分別對物件套用不同程度的保護：

- 合規模式
- 控管模式

在合規模式下，任何使用者 (包括 AWS 帳戶中的根使用者) 都無法覆寫或刪除受保護的物件版本。當物件在合規模式中受到鎖定時，您無法變更物件的保留模式，亦無法縮短它的保留期。合規模式可協助確保物件版本在保留期間均不會受到覆寫或刪除。

> **Note**
>
> 在符合性模式下刪除物件的保留日期到期之前，唯一的方法是刪除相關物件 AWS 帳戶。

在控管模式中，除非使用者具備特殊許可，否則都無法覆寫或刪除物件版本，或更改其鎖定設定。使用控管模式時，您可以保護物件免於遭到大多數使用者刪除，但您仍可以將許可授予部分使用者，讓他們視需要更改保留設定或刪除物件。您也可以使用控管模式來測試保留期設定，之後再建立合規模式的保留期。

若要覆寫或移除控管模式的保留設定，您必須具備 `s3:BypassGovernanceRetention` 許可，且需在任何請求中明確加入 `x-amz-bypass-governance-retention:true` 作為請求標頭，才能請求覆寫控管模式。

> **Note**
>
> 根據預設，Amazon S3 主控台會加入 `x-amz-bypass-governance-retention:true` 標頭。如果您嘗試刪除受控管模式保護的物件，且您具備 `s3:BypassGovernanceRetention` 許可，則操作會成功。

法務保存

使用物件鎖定，您還可以對物件版本設定法務保存。法務保存就像保留期一樣，可避免物件版本遭到覆寫或刪除。不過，法務保存不具有一段相關聯的固定期間，除非將其移除，否則會持續有效。任何具備 `s3:PutObjectLegalHold` 許可的使用者均可自由設定並移除法務保存。

法務保存和保留期是彼此獨立的。當您為物件版本設定法務保存時，並不會影響該物件版本的保留模式或保留期。

例如，假設您在物件版本同時受到保留期保護的情況下，為物件版本設定法務保存。如果保留期到期，則物件不會失去其 WORM 保護。反之，法務保存會持續保護物件，直到授權使用者明確移除法務保存為止。同樣地，如果您移除法務保存但物件版本的保留期仍有效，則直到保留期過期為止物件版本仍會受到保護。

使用 S3 物件鎖定的最佳做法

如果您想要保護大多數使用者在預先定義的保留期間刪除物件，但同時希望某些具有特殊權限的使用者能夠彈性地變更保留設定或刪除物件，請考慮使用治理模式。

如果您不想讓任何使用者 (包括您的中的 root 使用者) 都能夠在預先定義的保留期間刪除物件 AWS 帳戶，請考慮使用符合性模式。如果您需要儲存合規資料，則可以使用此模式。

當您不確定您希望對象保持不可變的時間時，可以使用 Legal Hold。這可能是因為您即將對數據進行了外部審核，並希望在審核完成之前保持對象不可變。或者，您可能有一個正在進行的項目，利用您希望在項目完成之前保持不可變的數據集。

所需的許可

物件鎖定操作需要特定許可。根據您嘗試執行的確切操作而定，您可能需要以下任何許可：

- `s3:BypassGovernanceRetention`
- `s3:GetBucketObjectLockConfiguration`
- `s3:GetObjectLegalHold`

- `s3:GetObjectRetention`
- `s3:PutBucketObjectLockConfiguration`
- `s3:PutObjectLegalHold`
- `s3:PutObjectRetention`

如需 Amazon S3 許可的完整清單及說明，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

如需有關搭配有許可使用條件的資訊，請參閱 使用條件鍵值區政策範例。

物件鎖定的考量事項

Amazon S3 物件鎖定可協助在一段固定時間內或無限期避免刪除或覆寫物件。

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 Amazon S3 REST API 來檢視或設定物件鎖定資訊。如需 S3 物件鎖定功能的一般資訊，請參閱 使用 S3 物件鎖定。

> **Important**
>
> - 您在儲存貯體上啟用物件鎖定後，就無法停用該儲存貯體的物件鎖定或暫停版本控制。
> - 具有物件鎖定的 S3 儲存貯體不能用作伺服器存取日誌的目的地儲存貯體。如需詳細資訊，請參閱 the section called “記錄伺服器存取”。

主題

- 檢視鎖定資訊的許可
- 繞過控管模式
- 搭配 S3 複寫使用物件鎖定
- 將物件鎖定搭配 Amazon S3 庫存清單
- 使用物件鎖定管理 S3 生命週期政策
- 使用物件鎖定管理刪除標記
- 使用 S3 Storage Lens 搭配物件鎖定
- 將物件上傳至已啟用物件鎖定的值區
- 設定事件和通知

- 使用儲存貯體政策設定保留期的限制

檢視鎖定資訊的許可

您可以使用 HeadObject 或 GetObject 操作，以程式設計的方式檢視 Amazon S3 物件版本的物件鎖定狀態。這兩項操作都會傳回所指定物件版本的保留模式、保留截止日期和法務保存狀態。此外，您可以使用 S3 庫存檢視 S3 儲存貯體中多個物件的物件鎖定狀態。

若要檢視物件版本的保留模式和保留期，您必須具備 `s3:GetObjectRetention` 許可。若要檢視物件版本的法務保存狀態，您必須具備 `s3:GetObjectLegalHold` 許可。若要檢視儲存貯體的預設保留組態，您必須具備 `s3:GetBucketObjectLockConfiguration` 許可。如果您在儲存貯體上發出物件鎖定組態的請求，但該儲存貯體未啟用 S3 物件鎖定，則 Amazon S3 會傳回錯誤。

繞過控管模式

如果您具有 `s3:BypassGovernanceRetention` 許可，即可對控管模式中鎖定的物件版本執行操作，就像它們未受保護一樣。這些操作包括刪除物件版本、縮短保留期，或藉由設定包含空白參數的新 `PutObjectRetention` 請求來移除物件鎖定保留期。

若要繞過控管模式，您必須在請求中明確指出您要繞過此模式。若要這麼做，請在 `PutObjectRetention` API 作業要求中加入`x-amz-bypass-governance-retention:true`標頭，或使用對等參數搭配透過 AWS CLI 或 AWS SDK 發出的要求。如果您具有 `s3:BypassGovernanceRetention` 許可，則 S3 主控台會對透過 S3 主控台提出的請求自動套用此標頭。

> **Note**
>
> 繞過控管模式並不會影響物件版本的法務保存狀態。如果物件版本已啟用法務保存，則法務保存會保留，並防止請求覆寫或刪除物件版本。

搭配 S3 複寫使用物件鎖定

您可以使用物件鎖定搭配 S3 複寫，以在 S3 儲存貯體之間啟用自動非同步複製鎖定物件及其保留中繼資料。這表示對於複製的物件，Amazon S3 會採用來源儲存貯體的物件鎖定組態。換句話說，如果來源值區已啟用「物件鎖定」，則目的地值區也必須啟用「物件鎖定」。如果物件直接上傳到目的地儲存貯體 (S3 複寫之外)，則會採用目標儲存貯體上設定的物件鎖定。當您使用複寫時，在來源儲存貯體中的物件會複寫到一或多個目的地儲存貯體。

若要在啟用物件鎖定的儲存貯體上設定複寫，您可以使用 S3 主控台 AWS CLI、Amazon S3 REST API 或 AWS 開發套件。

> **Note**
>
> 若要將 Object Lock 與複寫搭配使用，您必須授與用來設定複寫的 AWS Identity and Access Management (IAM) 角色中的來源 S3 儲存貯體兩個額外許可。這兩項額外的許可為 `s3:GetObjectRetention` 和 `s3:GetObjectLegalHold`。若該角色有 `s3:Get*` 許可陳述式，該陳述式即符合需求。如需詳細資訊，請參閱 設定即時複製的權限。
> 如需有關 S3 複寫的一般資訊，請參閱 複製物件概觀。
> 如需設定 S3 複寫的範例，請參閱 設定即時複製的範例。

將物件鎖定搭配 Amazon S3 庫存清單

您可以設定 Amazon S3 庫存清單，以依照定義的排程在 S3 儲存貯體中建立物件的清單。您可以設定讓 Amazon S3 庫存清單包含物件的下列物件鎖定中繼資料：

- 保留截止日期
- 保留模式
- 法務保存狀態

如需詳細資訊，請參閱 Amazon S3 清查。

使用物件鎖定管理 S3 生命週期政策

受保護物件上的物件生命週期管理組態仍可正常運作，包括放置刪除標記的功能。但是，S3 生命週期到期政策無法刪除物件的鎖定版本。無論物件位於哪個儲存類別，以及在儲存類別之間的 S3 生命週期轉換期間，物件鎖定都會維持不變。

如需有關管理物件生命週期的詳細資訊，請參閱管理儲存生命週期。

使用物件鎖定管理刪除標記

雖然您無法刪除受保護的物件版本，但您仍可以為該物件版本建立刪除標記。將刪除標記放置在物件上，並不會刪除物件或其物件版本。不過，這會使得 Amazon S3 在大多數方面表現得彷彿物件已遭刪除一樣。如需詳細資訊，請參閱「使用刪除標記」。

> **Note**
>
> 不論底層物件上是否設有任何保留期或法務保存，刪除標記均不受 WORM 保護。

使用 S3 Storage Lens 搭配物件鎖定

若要查看啟用物件鎖定的儲存體位元組和物件計數的指標，您可以使用 Amazon S3 Storage Lens。S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。

如需詳細資訊，請參閱 使用 S3 Storage Lens 保護您的資料。

如需指標的完整清單，請參閱Amazon S3 Storage Lens 指標詞彙表。

將物件上傳至已啟用物件鎖定的值區

若要上傳使用「物件鎖定」設定保留期間的物件，任何要求都需要`Content-MD5`標頭。MD5 摘要是將物件上傳至值區之後驗證物件完整性的一種方法。上傳物件後，Amazon S3 會計算物件的 MD5 摘要，並將其與您提供的值進行比較。僅當兩個摘要匹配時，請求才會成功。S3 主控台會自動新增此標頭，但是您必須在使用 PutObjectAPI 時指定此標頭。

如需詳細資訊，請參閱 上傳物件時使用 Content-MD5。

設定事件和通知

您可以使用 Amazon S3 事件通知，透過使用追蹤對物件鎖定組態和資料的存取和變更 AWS CloudTrail。如需相關資訊 CloudTrail，請參閱什麼是 AWS CloudTrail？ 在《AWS CloudTrail 使用者指南》中。

您還可以使用 Amazon CloudWatch 根據這些數據生成提醒。有關詳細信息 CloudWatch，請參閱什麼是 Amazon CloudWatch？ 在 Amazon 用 CloudWatch 戶指南。

使用儲存貯體政策設定保留期的限制

您可以使用儲存貯體政策來設定儲存貯體允許的最短和最長保留期。最長保留期為 100 年。

下列範例顯示的儲存貯體原則使用 `s3:object-lock-remaining-retention-days` 條件索引鍵，設定最長 10 天的保留期間。

```
{
```

```
    "Version": "2012-10-17",
    "Id": "SetRetentionLimits",
    "Statement": [
        {
            "Sid": "SetRetentionPeriod",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:PutObjectRetention"
            ],
            "Resource": "arn:aws:s3:::example-s3-bucket1/*",
            "Condition": {
                "NumericGreaterThan": {
                    "s3:object-lock-remaining-retention-days": "10"
                }
            }
        }
    ]
}
```

> **Note**
>
> 如果您的儲存貯體是複寫組態的目的地儲存貯體，則您可以針對使用複寫建立的物件複本設定允許的最短和最長保留期。若要這麼做，您必須在儲存貯體政策中允許 `s3:ReplicateObject` 動作。如需複寫許可的詳細資訊，請參閱 the section called “設定許可”。

如需儲存貯體政策的詳細資訊，請參閱下列主題：

- 服務授權參考資料中適用於 Amazon S3 的動作、資源和條件金鑰
- 物件操作
- 使用條件鍵值區政策範例

設定 S3 物件鎖定

使用 Amazon S3 物件鎖定，您可以使用 write-once-read-many(WORM) 模型將物件存放在 Amazon S3 中。您可以使用 S3 物件鎖定，讓物件在固定期間或無限期免於遭到刪除或覆寫。如需物件鎖定功能的一般資訊，請參閱 使用 S3 物件鎖定。

在鎖定任何物件之前，您必須先在儲存貯體上啟用 S3 版本控制和物件鎖定。之後您就可以設定保留期和/或法務保存。

若要使用物件鎖定，您必須擁有特定許可。如需與各種物件鎖定操作相關的許可清單，請參閱 the section called “所需的許可”。

> **Important**
>
> - 您在儲存貯體上啟用物件鎖定後，就無法停用該儲存貯體的物件鎖定或暫停版本控制。
> - 具有物件鎖定的 S3 儲存貯體不能用作伺服器存取日誌的目的地儲存貯體。如需詳細資訊，請參閱 the section called “記錄伺服器存取”。

主題

在建立新的 S3 儲存貯體時啟用物件鎖定

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 Amazon S3 REST API，在建立新的 S3 儲存貯體時啟用物件鎖定。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 選擇 Create bucket (建立儲存貯體)。

   Create bucket (建立儲存貯體) 頁面隨即開啟。
4. 針對 Bucket name (儲存貯體名稱)，輸入儲存貯體的名稱。

> **Note**
>
> 建立儲存貯體後，便無法變更其名稱。如需儲存貯體命名的詳細資訊，請參閱 儲存貯體命名規則。

5. 在「區域」中，選擇您 AWS 區域 要儲存貯體的位置。
6. 在物件擁有權下，選擇停用或啟用上傳到儲存貯體中之物件的存取控制清單 (ACL) 和控制擁有權。
7. 在封鎖此儲存貯體的公開存取設定之下，選擇要套用至儲存貯體的封鎖公開存取設定。
8. 在儲存貯體版本控制下，選擇已啟用。

   物件鎖定只可搭配版本化的儲存貯體運作。
9. (選用) 在 Tags (標籤) 下，您可以選擇新增標籤至儲存貯體。標籤是用來分類儲存和分配成本的鍵值對。
10. 在進階設定下，尋找物件鎖定並選擇啟用。

    您必須確實了解，啟用物件鎖定將永久允許鎖定此儲存貯體中的物件。
11. 選擇建立儲存貯體。

使用 AWS CLI

下列 `create-bucket` 範例會建立名為 *`example-s3-bucket1`* 且啟用物件鎖定的新 S3 儲存貯體：

```
aws s3api create-bucket --bucket example-s3-bucket1 --object-lock-enabled-for-bucket
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 create-bucket。

> **Note**
>
> 您可以使用從控制台運行 AWS CLI 命令 AWS CloudShell。 AWS CloudShell 是一個以瀏覽器為基礎的預先驗證殼層，您可以直接從. AWS Management Console如需詳細資訊，請參閱什麼是 CloudShell？ 在《AWS CloudShell 使用者指南》中。

使用 REST API

您可以使用 REST API 建立已啟用物件鎖定的新 S3 儲存貯體。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 CreateBucket。

使用 AWS 軟體開發套件

如需使用 AWS SDK 建立新 S3 儲存貯體時如何啟用物件鎖定的範例，請參閱搭CreateBucket配 AWS 開發套件或 CLI 使用。

如需如何使用 AWS SDK 取得目前物件鎖定組態的範例，請參閱搭GetObjectLockConfiguration配 AWS 開發套件或 CLI 使用。

如需展示使用 AWS SDK 之不同物件鎖定功能的互動式案例，請參閱。使用 AWS 開發套件使用 Amazon S3 物件鎖定功能

如需使用不同 AWS SDK 的一般資訊，請參閱使用開發 AWS 套件使用 Amazon S3 進行開發。

在現有 S3 儲存貯體上啟用物件鎖定

您可以使用 Amazon S3 主控台、 AWS 開發套件或 Amazon S3 REST API 為現有 S3 儲存貯體啟用物件鎖定。 AWS CLI

使用 S3 主控台

> **Note**
>
> 物件鎖定只可搭配版本化的儲存貯體運作。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在儲存貯體清單中，選擇要啟用物件鎖定的儲存貯體名稱。
4. 選擇屬性索引標籤。
5. 在屬性下，向下捲動至物件鎖定區段，然後選擇編輯。
6. 在物件鎖定下，選擇啟用。

您必須確實了解，啟用物件鎖定將永久允許鎖定此儲存貯體中的物件。

7. 選擇儲存變更。

使用 AWS CLI

下列 `put-object-lock-configuration` 範例命令會在名為 *example-s3-bucket1* 的儲存貯體上設定 50 天的物件鎖定保留期：

```
aws s3api put-object-lock-configuration --bucket example-s3-bucket1 --object-lock-
configuration='{ "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention":
 { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 put-object-lock-configuration。

> **Note**
>
> 您可以使用從控制台運行 AWS CLI 命令 AWS CloudShell。 AWS CloudShell 是一個以瀏覽器為基礎的預先驗證殼層，您可以直接從. AWS Management Console如需詳細資訊，請參閱什麼是 CloudShell？ 在《AWS CloudShell 使用者指南》中。

使用 REST API

您可以使用 Amazon S3 REST API 在現有 S3 儲存貯體上啟用物件鎖定。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 PutObjectLockConfiguration。

使用 AWS 軟體開發套件

如需如何使用 AWS SDK 為現有 S3 儲存貯體啟用物件鎖定的範例，請參閱搭PutObjectLockConfiguration配 AWS 開發套件或 CLI 使用。

如需如何使用 AWS SDK 取得目前物件鎖定組態的範例，請參閱搭GetObjectLockConfiguration配 AWS 開發套件或 CLI 使用。

如需展示使用 AWS SDK 之不同物件鎖定功能的互動式案例，請參閱。使用 AWS 開發套件使用 Amazon S3 物件鎖定功能

如需使用不同 AWS SDK 的一般資訊，請參閱使用開發 AWS 套件使用 Amazon S3 進行開發。

設定或修改 S3 物件的法務保存

您可以使用 Amazon S3 主控台、 AWS CLI AWS 開發套件或 Amazon S3 REST API 來設定或移除 S3 物件的法律訴訟保留。

> **Important**
>
> - 如果您要設定物件的法務保存，則物件的儲存貯體必須已啟用物件鎖定。
> - 當您對儲存貯體中具有明確的個別保留模式和保留期的物件版本執行 PUT 時，物件版本的個別物件鎖定設定會覆寫任何儲存貯體屬性保留設定。

如需詳細資訊，請參閱「the section called “法務保存”」。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在儲存貯體清單中，選擇包含您要設定或移除法務保存之物件的儲存貯體名稱。
4. 在物件清單中，選取您要設定或修改法務保存的物件。
5. 在物件屬性頁面上，尋找物件鎖定法務保存區段，然後選擇編輯。
6. 選擇啟用以設定法務保存，或選擇停用以移除法務保存。
7. 選擇儲存變更。

使用 AWS CLI

下列 `put-object-legal-hold` 範例會在名為 *`example-s3-bucket1`* 的儲存貯體中設定物件 *`my-image.fs`* 的法務保存：

```
aws s3api put-object-legal-hold --bucket example-s3-bucket1 --key my-image.fs --legal-hold="Status=ON"
```

下列 `put-object-legal-hold` 範例會在名為 *`example-s3-bucket1`* 的儲存貯體中移除物件 *`my-image.fs`* 的法務保存：

```
aws s3api put-object-legal-hold --bucket example-s3-bucket1 --key my-image.fs --legal-hold="Status=OFF"
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 put-object-legal-hold。

> **Note**
>
> 您可以使用從控制台運行 AWS CLI 命令 AWS CloudShell。 AWS CloudShell 是一個以瀏覽器為基礎的預先驗證殼層，您可以直接從. AWS Management Console如需詳細資訊，請參閱什麼是 CloudShell？ 在《AWS CloudShell 使用者指南》中。

使用 REST API

您可以使用 REST API 設定或修改物件的法務保存。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 PutObjectLegalHold。

使用 AWS 軟體開發套件

如需如何使用 AWS SDK 對物件設定法律訴訟保留的範例，請參閱搭PutObjectLegalHold配 AWS 開發套件或 CLI 使用。

如需如何透過 AWS SDK 取得目前合法保留狀態的範例，請參閱使用 AWS 開發套件取得 Amazon S3 物件的合法保留組態。

如需展示使用 AWS SDK 之不同物件鎖定功能的互動式案例，請參閱。使用 AWS 開發套件使用 Amazon S3 物件鎖定功能

如需使用不同 AWS SDK 的一般資訊，請參閱使用開發 AWS 套件使用 Amazon S3 進行開發。

設定或修改 S3 物件的保留期

您可以使用 Amazon S3 主控台、 AWS CLI AWS 開發套件或 Amazon S3 REST API 來設定或修改 S3 物件的保留期。

> **Important**
>
> - 如果您要設定物件的保留期，則物件的儲存貯體必須已啟用物件鎖定。
> - 當您對儲存貯體中具有明確的個別保留模式和保留期的物件版本執行 PUT 時，物件版本的個別物件鎖定設定會覆寫任何儲存貯體屬性保留設定。

- 在符合性模式下刪除物件的保留日期到期之前，唯一的方法是刪除相關物件 AWS 帳戶。

如需詳細資訊，請參閱「保留期」。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在儲存貯體清單中，選擇包含您要設定或修改保留期之物件的儲存貯體名稱。
4. 在物件清單中，選取您要設定或修改保留期的物件。
5. 在物件屬性頁面上，尋找物件鎖定保留區段，然後選擇編輯。
6. 在保留下，選擇啟用以設定保留期，或選擇停用以移除保留期。
7. 如果您選擇啟用，在保留模式下選擇控管模式或合規模式。如需詳細資訊，請參閱 保留模式。
8. 在保留截止日期下，選擇您希望保留期結束的日期。在這段期間，您的物件會受到 WORM 保護，而無法將其覆寫或刪除。如需詳細資訊，請參閱 保留期。
9. 選擇 Save changes (儲存變更)。

使用 AWS CLI

下列 `put-object-retention` 範例會在名為 *`example-s3-bucket1`* 的儲存貯體中設定物件 *`my-image.fs`* 的保留期且結束日期為 2025 年 1 月 1 日：

```
aws s3api put-object-retention --bucket example-s3-bucket1 --key my-image.fs --
retention='{ "Mode": "GOVERNANCE", "RetainUntilDate": "2025-01-01T00:00:00" }'
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 put-object-retention。

> **Note**
>
> 您可以使用從控制台運行 AWS CLI 命令 AWS CloudShell。 AWS CloudShell 是一個以瀏覽器為基礎的預先驗證殼層，您可以直接從. AWS Management Console如需詳細資訊，請參閱什麼是 CloudShell？ 在《AWS CloudShell 使用者指南》中。

使用 REST API

您可以使用 REST API 設定物件的保留期。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 PutObjectRetention。

使用 AWS 軟體開發套件

如需如何在具有 AWS SDK 的物件上設定保留期的範例，請參閱搭PutObjectRetention配 AWS 開發套件或 CLI 使用。

如需如何使用 AWS SDK 取得物件的保留期範例，請參閱搭GetObjectRetention配 AWS 開發套件或 CLI 使用。

如需展示使用 AWS SDK 之不同物件鎖定功能的互動式案例，請參閱。使用 AWS 開發套件使用 Amazon S3 物件鎖定功能

如需使用不同 AWS SDK 的一般資訊，請參閱使用開發 AWS 套件使用 Amazon S3 進行開發。

設定或修改 S3 儲存貯體的預設保留期

您可以使用 Amazon S3 主控台、開發套件或 Amazon S3 REST AWS API AWS CLI，在 S3 儲存貯體上設定或修改預設保留期。您可以指定一段以天或年為單位的時間，代表要保護放置在儲存貯體中的每個物件版本的期間。

> **Important**
>
> - 如果您要設定儲存貯體的預設保留期，則儲存貯體必須已啟用物件鎖定。
> - 當您對儲存貯體中具有明確的個別保留模式和保留期的物件版本執行 PUT 時，物件版本的個別物件鎖定設定會覆寫任何儲存貯體屬性保留設定。
> - 在符合性模式下刪除物件的保留日期到期之前，唯一的方法是刪除相關物件 AWS 帳戶。

如需詳細資訊，請參閱「保留期」。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。

3. 在儲存貯體清單中，選擇您要設定或修改預設保留期的儲存貯體名稱。
4. 選擇屬性索引標籤。
5. 在屬性下，向下捲動至物件鎖定區段，然後選擇編輯。
6. 在預設保留下，選擇啟用以設定預設保留，或選擇停用以移除預設保留。
7. 如果您選擇啟用，在保留模式下選擇控管模式或合規模式。如需詳細資訊，請參閱 保留模式。
8. 在預設保留期下，選擇您希望保留期持續的天數或年數。放置在此儲存貯體中的物件將會鎖定長達此天數或年數。如需詳細資訊，請參閱 保留期。
9. 選擇 Save changes (儲存變更)。

使用 AWS CLI

下列 `put-object-lock-configuration` 範例命令會使用合規模式在名為 *`example-s3-bucket1`* 的儲存貯體上設定 50 天的物件鎖定保留期：

```
aws s3api put-object-lock-configuration --bucket example-s3-bucket1 --object-lock-
configuration='{ "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention":
 { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

下列 `put-object-lock-configuration` 範例會移除儲存貯體的預設保留組態：

```
aws s3api put-object-lock-configuration --bucket example-s3-bucket1 --object-lock-
configuration='{ "ObjectLockEnabled": "Enabled"}'
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 put-object-lock-configuration。

> **Note**
>
> 您可以使用從控制台運行 AWS CLI 命令 AWS CloudShell。 AWS CloudShell 是一個以瀏覽器為基礎的預先驗證殼層，您可以直接從. AWS Management Console如需詳細資訊，請參閱什麼是 CloudShell？ 在《AWS CloudShell 使用者指南》中。

使用 REST API

您可以使用 REST API 在現有 S3 儲存貯體上設定預設保留期。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 PutObjectLockConfiguration。

使用 AWS 軟體開發套件

如需如何使用 AWS SDK 在現有 S3 儲存貯體上設定預設保留期的範例，請參閱搭PutObjectLockConfiguration配 AWS 開發套件或 CLI 使用。

如需展示使用 AWS SDK 之不同物件鎖定功能的互動式案例，請參閱。使用 AWS 開發套件使用 Amazon S3 物件鎖定功能

如需使用不同 AWS SDK 的一般資訊，請參閱使用開發 AWS 套件使用 Amazon S3 進行開發。

使用 Amazon S3 儲存體方案

Amazon S3 中的每個物件都有與其相關聯的儲存體方案。例如，若您列出 S3 儲存貯體中的物件，主控台會在清單中顯示所有物件的儲存體方案。Amazon S3 為您存放的物件提供各種儲存體方案。您可以根據使用案例情境和效能存取需求，來選擇類別。所有儲存體方案都提供高耐用性。

下列各節將詳細說明各種儲存體方案，以及如何設定物件的儲存體方案。

主題

- 經常存取物件的儲存體方案
- 存取模式會變更或不明的自動最佳化資料的儲存體方案
- 不常存取物件的儲存體方案
- 很少存取物件的儲存區類別
- Amazon S3 on Outposts 的儲存方案
- 比較 Amazon S3 儲存方案
- 設定物件的儲存體方案

經常存取物件的儲存體方案

對於需要高效能的使用案例 (需要毫秒存取時間) 以及經常存取的資料，Amazon S3 提供以下儲存體方案：

- S3 標準 - 預設儲存體類別。若在上傳物件時未指定儲存體方案，Amazon S3 會指派 S3 Standard 儲存體方案。
- S3 Express One Zone：Amazon S3 Express One Zone 是一種高效能的單一區域 Amazon S3 儲存類別，專門為對延遲最敏感的應用程式提供一致的個位數毫秒資料存取。S3 Express One Zone 是目前可用的最低延遲雲端物件儲存類別，資料存取速度提高 10 倍，而且請求成本比 S3 標準低

50%。使用 S3 Express One Zone，您的資料會以備援方式儲存在單一可用區域內的多部裝置上。如需詳細資訊，請參閱 什麼是 S3 Express One Zone？。

- 低冗餘 - 低冗餘儲存 (RRS) 儲存體類別是專為非關鍵且可重現的資料所設計，能以低於 S3 標準儲存體類別的備援層級存放。

> **Important**
>
> 不建議使用此儲存體類別。S3 標準儲存體類別比較符合成本效益。

為達到耐用性，RRS 物件的平均年物件遺失率預計為百分之 0.01。如果遺失了 RRS 物件，Amazon S3 會在對該物件提出請求時傳回 405 錯誤。

存取模式會變更或不明的自動最佳化資料的儲存體方案

S3 Intelligent-Tiering 是一種 Amazon S3 儲存體類別，旨在透過自動將資料移至最具成本效益的存取層，使儲存成本最佳化，且不會影響效能或帶來額外負荷。S3 Intelligent-Tiering 是唯一的雲端儲存體類別，當存取模式改變時，其可在精細物件層級上兩個存取層之間移動資料，以自動節省成本。對於存取模式不明或不斷變化的資料，S3 Intelligent-Tiering 是最佳化儲存成本的理想儲存體類別。S3 Intelligent-Tiering 不收取擷取費。

只需每月支付少許的物件監控和自動化費用，S3 Intelligent-Tiering 即可監控存取模式，並自動將未存取的物件移至低成本存取層。S3 Intelligent-Tiering 可在三個低延遲和高輸送量存取層中自動節省儲存成本。對於可非同步存取的資料，您可以選擇啟用 S3 Intelligent-Tiering 儲存方案內的自動封存功能。S3 Intelligent-Tiering 旨在提供的設計可提供 99.9% 的可用性和 99.999999999% 的耐用性。

S3 Intelligent-Tiering 會自動將物件存放在三個存取層：

- 經常存取 - 上傳或轉換到 S3 Intelligent-Tiering 的物件自動存放在經常存取層。
- 不常存取 - S3 Intelligent-Tiering 會將連續 30 天未存取的物件移到不常存取層。
- Archive Instant Access - 使用 S3 Intelligent-Tiering，連續 90 天未存取的任何現存物件都會自動移至 Archive Instant Access 層。

除了這三個存取層之外，S3 Intelligent-Tiering 也會提供兩個選用的封存存取層：

- Archive Access - S3 Intelligent-Tiering 可讓您選擇啟用 Archive Access 層，以取得可以非同步存取的資料。啟用後，封存存取層會自動封存至少連續 90 天未存取過的物件。

- Deep Archive Access - S3 Intelligent-Tiering 可讓您選擇啟用 Deep Archive Access 層，以取得可以非同步存取的資料。啟用後，Deep Archive 存取層會自動封存至少連續 180 天未存取過的物件。

> **Note**
>
> - 只有在您想要略過 Archive Instant 存取層時，才啟用封存存取層 90 天。封存存取層提供的儲存成本略低，且 minute-to-hour 擷取時間較低。Archive Instant Access 層可提供毫秒存取和高輸送量效能。
> - 只有當應用程式可以非同步存取物件時，才可啟用 Archive Access 和 Deep Archive Access 層。如果您要擷取的物件存放在 Archive Access 或 Deep Archive Access 層中，請先使用 `RestoreObject` 還原物件。

您可以將新建立的資料移至 S3 Intelligent-Tiering，將它設定為預設儲存體類別。您也可以選擇使用 `PutBucketIntelligentTieringConfiguration`API 操作、或 Amazon S3 主控台啟用一個或兩個存檔存取層。 AWS CLI如需有關使用 S3 Intelligent-Tiering 和啟用封存存取層的資訊，請參閱 使用 S3 Intelligent-Tiering。

若要存取 Archive Access 或 Deep Archive Access 層中的物件，您必須先將它們還原。如需詳細資訊，請參閱 從 S3 Intelligent-Tiering 封存存取層和 Deep Archive 存取層還原物件。

> **Note**
>
> 若物件大小低於 128 KB，便不會受到監控且不符合自動分層的資格。較小的物件一律存放在經常存取層中。如需 S3 Intelligent-Tiering 的詳細資訊，請參閱 S3 Intelligent-Tiering 存取層。

不常存取物件的儲存體方案

S3 標準 – IA 和 S3 單區域 – IA 儲存體方案是針對存活時間較長且不常存取的資料所設計。(IA 表示不常存取。) S3 標準 – IA 和 S3 單區域 – IA 物件可供毫秒存取 (類似於 S3 標準型儲存體方案)。Amazon S3 對於這些物件會收取擷取費用，因此最適用於不常存取的資料。如需定價資訊，請參閱 Amazon S3 定價。

例如，您可以選擇 S3 標準 – IA 和 S3 單區域 – IA 儲存體方案來執行下列作業：

- 用於存放備份。

- 用於較舊且不常存取，但仍需要毫秒存取的資料。例如，當您上傳物件時，可以選擇 S3 Standard 儲存體方案，並使用生命週期組態告訴 Amazon S3 將轉換為 S3 標準型 – IA 或 S3 單區域型 – IA 方案。

  如需生命週期管理的詳細資訊，請參閱「管理儲存生命週期」。

> **Note**
>
> S3 標準 – IA 和 S3 單區域 – IA 儲存體方案適合大於 128 KB 且至少打算存放 30 天的大型物件。如果物件小於 128 KB，Amazon S3 還是會按 128 KB 計算收費。如果您在 30 天的最低儲存體持續期間結束前刪除物件，仍會收取 30 天的費用。在 30 天前刪除、覆寫或轉換至其他儲存類別的物件會產生一般儲存使用費，再加上最少 30 天的其餘部分按比例計費。如需定價資訊，請參閱 Amazon S3 定價。

這些儲存體方案不同之處如下：

- S3 標準 - IA - Amazon S3 會以備援方式在多個不同地理位置的可用區域存放物件資料 (與 S3 標準儲存體類別相似)。S3 標準 – IA 物件可在遺失可用區域時迅速恢復。此儲存體方案提供比 S3 單區域 – IA 方案更高的可用性和彈性。
- S3 單區域 - IA - Amazon S3 僅將物件資料存放在一個可用區域，而比 S3 標準 – IA 更經濟實惠。然而，該資料無法在碰到可用區域因地震和水災等災害而產生實體損失時恢復。S3 單區域 - IA 儲存體類別與 S3 標準 - IA 一樣耐用，但是具備較低的可用性和彈性。如需儲存體方案耐用性和可用性的比較，請參閱本節末尾的 比較 Amazon S3 儲存方案。如需定價資訊，請參閱 Amazon S3 定價。

我們建議下列作法：

- S3 標準 - IA - 用於您的主要資料或無法重建之資料的唯一複本。
- S3 單區域 - IA - 如果可用區域失敗時可重建資料，請使用此類別，也可在設定 S3 跨區域複寫 (CRR) 時用於物件複寫。

很少存取物件的儲存區類別

S3 Glacier 即時擷取、S3 Glacier 彈性擷取和 S3 Glacier 深度存檔儲存類別是專為低成本的長期資料儲存和資料存檔而設計。這些儲存類別可提供與 S3 Standard 和 S3 標準 – IA 儲存類別相同的耐久性和彈性。如需 S3 Glacier 儲存類別的詳細資訊，請參閱使用 S3 冰川儲存類別的長期資料儲存。

Amazon S3 提供下列 S3 冰川儲存類別：

- S3 Glacier 即時擷取 — 用於很少存取且需要幾毫秒擷取的長期資料。此儲存類別中的資料可用於即時存取。
- S3 Glacier Flexible Retrieval - 用於可能需要在幾分鐘內擷取部分資料的封存。此儲存類別中的資料已封存，無法用於即時存取。
- S3 Glacier Deep Archive - 用於很少需要存取的封存資料。此儲存類別中的資料已封存，無法用於即時存取。

擷取已封存的物件

您可以使用章節 設定物件的儲存體方案 中其他儲存類別相同的方式，將物件的儲存類別設為 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive。不過，S3 Glacier 彈性擷取和 S3 Glacier Deep Archive 物件會存檔，而且無法用於即時存取。如需詳細資訊，請參閱 存檔儲存。

> **Note**
>
> 當您使用 S3 冰川儲存類別時，您的物件會保留在 Amazon S3 中。您無法透過個別的 Amazon S3 Glacier 服務直接存取物件。如需 Amazon S3 冰川服務的相關資訊，請參閱 Amazon S3 冰川開發人員指南。

Amazon S3 on Outposts 的儲存方案

使用 Amazon S3 on Outposts，您可以在 AWS Outposts 資源上建立 S3 儲存貯體，並針對需要本機資料存取、本機資料處理和資料存放區的應用程式存放和擷取物件現場部署。您可以使用與 Amazon S3 相同的 AWS Outposts API 操作和功能，包括存取政策、加密和標記。您可以透過 AWS Management Console、 AWS CLI、 AWS 開發套件或 REST API 在 Outposts 上使用 S3。

S3 on Outposts 提供新的儲存體方案 S3 Outposts (OUTPOSTS)。S3 Outposts 儲存體方案僅適用於存放在 Outposts 儲存貯體中的物件。如果您嘗試將此儲存類別與中的 S3 儲存貯體搭配使用 AWS 區域，就會發生錯`InvalidStorageClass`誤。此外，如果您嘗試一起使用其他 S3 儲存體方案與儲存在 S3 on Outposts 中的物件，也會導致發生相同的錯誤。

存放在 S3 Outposts (OUTPOSTS) 儲存體方案中的物件一律使用伺服器端加密與 Amazon S3 受管加密金鑰 (SSE-S3) 進行加密。如需詳細資訊，請參閱 使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密。

您也可以明確選擇使用伺服器端加密與客戶提供的加密金鑰 (SSE-C) 來加密存放在 S3 Outposts 儲存體方案中的物件。如需詳細資訊，請參閱 搭配客戶提供的金鑰 (SSE-C) 使用伺服器端加密。

> **Note**
>
> Outposts 上的 S3 不支援使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 進行伺服器端加密。

如需 S3 on Outposts 的詳細資訊，請參閱 什麼是 Amazon S3 on Outposts？。

比較 Amazon S3 儲存方案

下表比較儲存體方案，包括其可用性、耐用性、最低儲存體持續期間及其他考量。

Storage Class	Designed for	Durability (designed for)	Availability (designed for)	Availability Zones	Min storage duration	Min billable object size	Other Considerations
STANDARD	Frequently accessed data	99.999999999%	99.99%	>= 3	None	None	None
STANDARD_IA	Long-lived, infrequently accessed data	99.999999999%	99.9%	>= 3	30 days	128 KB	Per GB retrieval fees apply.
INTELLIGENT_TIERING	Long-lived data with changing or unknown access patterns	99.999999999%	99.9%	>= 3	30 days	None	Monitoring and automation fees per object apply. No retrieval fees.
ONEZONE_IA	Long-lived, infrequently accessed, non-critical data	99.999999999%	99.5%	1	30 days	128 KB	Per GB retrieval fees apply. Not resilient to the loss of the Availability Zone.
GLACIER	Long-term data archiving with retrieval times ranging from minutes to hours	99.999999999%	99.99% (after you restore objects)	>= 3	90 days	None	Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects.
DEEP_ARCHIVE	Archiving rarely accessed data with a default retrieval time of 12 hours	99.999999999%	99.99% (after you restore objects)	>= 3	180 days	None	Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects.
RRS (Not recommended)	Frequently accessed, non-critical data	99.99%	99.99%	>= 3	None	None	None

* 對於每個封存的物件，S3 Glacier Flexible Retrieval 需要 40 KB 的額外中繼資料。這包括按 S3 Glacier Flexible Retrieval 費率計費的 32 KB 中繼資料 (識別和擷取資料所需)，以及按 S3 標準費率計費的額外 8 KB 資料。對於封存至 S3 Glacier Flexible Retrieval 的物件，若要維護使用者定義的名稱和中繼資料，需要 S3 標準費率。如需儲存體類別的詳細資訊，請參閱 Amazon S3 儲存體類別。

** S3 Glacier Deep Archive 存每個存檔物件需要 40 KB 的額外中繼資料。這包括按 S3 Glacier Deep Archive 費率計費的 32 KB 中繼資料 (識別和擷取資料所需)，以及按 S3 標準費率計費的額外 8 KB 資料。對於封存至 Amazon S3 Glacier Deep Archive 的物件，若要維護使用者定義的名稱和中繼資料，需要 S3 標準費率。如需儲存體類別的詳細資訊，請參閱 Amazon S3 儲存類別。

請注意，除了 S3 One Zone-IA 和 S3 Express One Zone 以外，所有儲存類別都具備彈性設計，可避免因災難造成的可用區域實體損失。此外，除了您應用程式案例的效能需求之外，請考慮成本。如需了解儲存體方案定價，請參閱 Amazon S3 定價。

設定物件的儲存體方案

若要設定和更新物件儲存類別，您可以使用 Amazon S3 主控台、 AWS 開發套件或 AWS Command Line Interface (AWS CLI)。所有這些方法都使用 Amazon S3 API 操作將請求傳送到 Amazon S3。

Amazon S3 API 操作支援依如下方式設定 (或更新) 物件的儲存體類別：

- 當建立新物件時，您可以指定其儲存體方案。例如，當使用 PUT 物件、POST 物件 和 Initiate Multipart Upload API 操作建立物件時，您可以新增 `x-amz-storage-class` 請求標頭來指定儲存體類別。若您沒有新增此標頭，Amazon S3 會使用 S3 標準型，即預設的儲存體類別。
- 您也可以使用 PUT 物件 - 複製 API 操作來建立物件複本，將已存放在 Amazon S3 中的物件變更為任何其他儲存體類別。不過，您無法使用 PUT 物件 - 複製來複製存放在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存體類別的物件。您也無法從 S3 單區域 – IA 轉換為 S3 Glacier Instant Retrieval。

  您使用相同的金鑰名稱，並指定如下的請求標頭，來複製相同儲存貯體中的物件：

  - 將`x-amz-metadata-directive` 標頭設為 `COPY`。
  - 將 `x-amz-storage-class` 標頭設為您想要使用的儲存體類別。

  在已啟用版本控制的儲存貯體中，您無法變更特定物件版本的儲存體方案。當您複製物件時，Amazon S3 會為其提供新的版本 ID。
- 如果物件大小小於 160 GB，您可以使用 Amazon S3 主控台變更物件的儲存類別。如果物件大小大於 160 GB，建議您新增 S3 生命週期組態，來變更物件的儲存類別。
- 如果您使用 Amazon S3 主控台變更具有使用者定義標籤之物件的儲存類別，則必須擁有該`s3:GetObjectTagging`權限。如果您要變更沒有使用者定義標籤但大小超過 16 MB 的物件的儲存空間類別，您也必須擁有`s3:GetObjectTagging`權限。如果目的地儲存貯體原則拒絕該`s3:GetObjectTagging`動作，則會更新物件的儲存區類別，但使用者定義的標籤會從物件中移除，而且您會收到錯誤訊息。
- 您可以在儲存貯體中新增 S3 生命週期組態，來指示 Amazon S3 變更物件的儲存方案。如需詳細資訊，請參閱「管理儲存生命週期」。
- 設定複寫組態時，您可以將複寫物件的儲存體方案設定為任何其他儲存體方案。不過，您無法複製存放在 S3 Glacier Instant Retrieval 或 S3 Glacier Deep Archive 儲存類別的物件。如需詳細資訊，請參閱 複寫組態。

將存取政策許可限制為特定儲存類別

當您授與 Amazon S3 操作的存取政策許可時，可以使用 `s3:x-amz-storage-class` 條件金鑰來限制儲存上傳物件時要使用的儲存類別。例如，當您授予 `s3:PutObject` 許可時，您可以限制物件上傳至特定儲存體類別。如需政策範例，請參閱 範例：限制物件上傳至具有特定儲存區類別的物件。

如需有關在政策中使用條件的詳細資訊，以及 Amazon S3 條件金鑰的完整清單，請參閱下列主題：

- 服務授權參考資料中適用於 Amazon S3 的動作、資源和條件金鑰
- 使用條件鍵值區政策範例

使用 S3 冰川儲存類別的長期資料儲存

Amazon S3 提供多種 S3 Glacier 儲存類別，這些類別旨在提供符合成本效益的解決方案，以存放不經常存取的長期資料。S3 冰川儲存類別包括：

- S3 Glacier Instant Retrieval
- S3 Glacier Flexible Retrieval
- S3 Glacier Deep Archive

您可以根據存取資料的頻率以及擷取資料的速度，選擇其中一個儲存類別。這些儲存類別都提供與 S3 標準儲存類別相同的耐久性和復原能力，但儲存成本較低。如需 S3 冰川儲存類別的詳細資訊，請參閱 https://aws.amazon.com/s3/storage-classes/glacier/。

主題

- 比較 S3 冰川儲存類別
- S3 Glacier Instant Retrieval
- S3 Glacier Flexible Retrieval
- S3 Glacier Deep Archive
- 存檔儲存
- 這些儲存類別與 S3 Glacier 服務有何不同

比較 S3 冰川儲存類別

每個 S3 Glacier 儲存類別都有所有物件的最短儲存持續時間。如果您在最低限度之前刪除、覆寫或將物件轉移到其他儲存類別，則需支付完整最短儲存期間的費用。

某些 S3 Glacier 儲存類別是封存的，這表示存放在這些類別中的物件會被封存，而且無法用於即時存取。如需詳細資訊，請參閱 存檔儲存。

專為較少頻率的存取模式而設計的儲存類別，擷取時間較長，可降低儲存成本。如需定價資訊，請參閱https://aws.amazon.com/s3/pricing/。

下表摘要說明選擇 S3 Glacier 儲存類別時要考量的重點：

S3 Glacier Instant Retrieval

我們建議對每季存取一次且需要毫秒擷取時間的長期資料使用 S3 Glacier 即時擷取。此儲存類別非常適合效能敏感的使用案例，例如影像託管、檔案共用應用程式，以及儲存醫療記錄以便在約會期間存取。

S3 Glacier 即時擷取儲存類別提供物件的即時存取，其延遲和輸送量效能與 S3 標準 — IA 儲存類別相同。與 S3 標準 — IA 相比，S3 Glacier 即時擷取具有較低的儲存成本，但資料存取成本較高。

S3 冰川即時擷取儲存類別中存放的資料物件大小下限為 128 KB。此儲存類別的最短儲存期限也為 90 天。

S3 Glacier Flexible Retrieval

對於每年存取一到兩次且不需要立即存取的存檔資料，建議使用 S3 Glacier 彈性擷取。S3 Glacier 彈性擷取提供彈性的擷取時間，協助您平衡成本，存取時間從數分鐘到數小時不等，以及免費的大量擷取。此儲存類別非常適合備份和災難復原。

存放在 S3 Glacier 中的物件彈性擷取會封存，而且無法用於即時存取。如需詳細資訊，請參閱 存檔儲存。若要存取這些物件，您必須先啟動還原要求，該要求會建立物件的暫時複本，您可以在要求完成時存取該複本。如需相關資訊，請參閱使用封存的物件。還原物件時，您可以選擇符合使用案例的擷取層，成本較低，還原時間更長。

下列擷取層適用於 S3 冰川彈性擷取：

- 快速擷取 — 通常會在 1-5 分鐘內還原物件。快速擷取會視需求而定，因此為了確保您擁有可靠且可預測的還原時間，建議您購買佈建的擷取容量。如需詳細資訊，請參閱 佈建的容量。
- 標準擷取 — 通常在 3 到 5 小時內還原物件，或在使用 S3 Batch 操作時在 1 分鐘到 5 小時內還原物件。如需詳細資訊，請參閱 使用批次操作還原物件。
- 大量擷取 — 通常會在 5 到 12 小時內還原物件。大量擷取是免費的。

S3 Glacier 彈性擷取儲存類別中物件的最短儲存持續時間為 90 天。

S3 Glacier 彈性擷取每個物件需要 40 KB 的額外中繼資料。這包括識別和擷取資料所需的 32 KB 中繼資料，按 S3 Glacier 彈性擷取的預設費率收費。需要額外的 8 KB 資料來維護存檔物件的使用者定義名稱和中繼資料，並以 S3 標準費率收費。

S3 Glacier Deep Archive

我們建議對每年存取少於一次的存檔資料使用 S3 Glacier 深度存檔。此儲存類別專為保留資料集多年而設計，以符合法規要求，也可用於備份或災難復原，或是任何不常存取的資料，您可以等待長達 72 小時才能擷取的資料。S3 Glacier Deep Archive 為 AWS中最低成本的儲存體選項。

存放在 S3 Glacier 深層封存中的物件會被封存，而且無法用於即時存取。如需詳細資訊，請參閱 存檔儲存。若要存取這些物件，您必須先啟動還原要求，該要求會建立物件的暫時複本，您可以在要求完成時存取該複本。如需相關資訊，請參閱使用封存的物件。還原物件時，您可以選擇符合使用案例的擷取層，成本較低，還原時間更長。

下列擷取層適用於 S3 Glacier Deep Archive：

- 標準擷取 — 通常可在 12 小時內還原物件，或在使用 S3 Batch 操作 9—12 小時內還原物件。如需詳細資訊，請參閱 使用批次操作還原物件。
- 大量擷取 — 通常在 48 小時內還原物件，成本僅為「標準」擷取層的一小部分。

S3 Glacier 深度存檔儲存類別中物件的最短儲存持續時間為 180 天。

S3 Glacier Deep Archive 每個物件需要 40 KB 的額外中繼資料。這包括識別和擷取資料所需的 32 KB 中繼資料，按 S3 Glacier Deep Archive 的預設費率收費。需要額外的 8 KB 資料來維護存檔物件的使用者定義名稱和中繼資料，並以 S3 標準費率收費。

存檔儲存

S3 冰川彈性擷取和 S3 Glacier Deep Archive 是存檔儲存類別。這意味著當您將對象存儲在這些存儲類中時，該對象已歸檔，並且無法直接訪問。若要存取已封存物件，請提交它的還原要求，然後等待服務還原物件。還原要求會還原物件的暫時複本，並在您在要求中指定的持續時間到期時刪除該副本。如需更多資訊，請參閱使用封存的物件。

對於每個封存物件，這些儲存類別需要 40 KB 的額外中繼資料。這包括識別和擷取資料所需的 32 KB 中繼資料，按照該儲存類別的預設費率收費。需要額外的 8 KB 資料來維護存檔物件的使用者定義名稱和中繼資料，並以 S3 標準費率收費。

當您使用多部分上傳來上傳這些儲存類別中的物件時，會以 S3 標準儲存類別費率計費。如需詳細資訊，請參閱 分段上傳與定價。

您可以還原這些儲存類別中的封存物件，每個帳戶每個帳戶最多可有 1,000 個交易 (TPS) 物件還原要求。 AWS 區域

這些儲存類別與 S3 Glacier 服務有何不同

S3 冰川儲存類別是 Amazon S3 服務的一部分，會將資料當做物件存放在 S3 儲存貯體中。您可以使用 S3 主控台或使用 S3 API 或 SDK 以程式設計方式管理這些儲存類別中的物件。將物件存放在 S3 Glacier 儲存類別時，您可以使用 S3 功能 (例如進階加密、物件標記和 S3 生命週期組態) 來協助管理資料可存取性和成本。

> **Important**
>
> 我們建議您對所有長期資料使用 Amazon S3 服務中的 S3 Glacier 儲存類別。

Amazon S3 Glacier (S3 Glacier) 服務是一項獨立的服務，可將資料以存檔形式存放在文件庫中。此服務不支援 Amazon S3 功能，也不提供資料上傳和下載操作的主控台支援。我們不建議您將 S3 Glacier 服務用於您的長期資料。無法從 Amazon S3 服務存取存放在此服務中的資料。如果您正在尋找 S3 冰川服務的相關資訊，請參閱 Amazon S3 冰川開發人員指南。若要將資料從 Amazon S3 Glacier 服務傳輸到 Amazon S3 中的儲存類別，請參閱 AWS 解決方案庫中的資料從 Amazon S3 冰川儲存庫傳輸到 Amazon S3。

Amazon S3 Intelligent Tiering

S3 Intelligent-Tiering 儲存類別旨在透過在存取模式變更時自動將資料移動到最具成本效益的存取層，將儲存成本最佳化，且帶來額外營運負荷或不會影響效能。只需每月支付少許的物件監控和自動化費用，S3 Intelligent-Tiering 即可監控存取模式，並自動將未存取的物件移至低成本層。

S3 Intelligent-Tiering 可在三個低延遲和高輸送量存取層中自動節省儲存成本。對於可非同步存取的資料，您可以選擇啟用 S3 Intelligent-Tiering 儲存方案內的自動封存功能。S3 Intelligent-Tiering 不會產生擷取費。如果物件稍後在不常存取層或 Archive Instant 存取層受到存取，則它會自動將物件移回經常存取層。在 S3 Intelligent-Tiering 儲存方案內於存取層間移動物件時，不會產生額外的分層費用。

S3 Intelligent-Tiering 是推薦用於存取模式未知、不斷變化或無法預測，且與物件大小或保留期無關的儲存方案，例如資料湖、資料分析和新應用程式。

如需有關使用 S3 Intelligent-Tiering 的資訊，請參閱下列各節：

主題

S3 Intelligent-Tiering 的運作方式

Amazon S3 Intelligent-Tiering 儲存類別會自動將物件存放在三個存取層。一個層針對頻繁存取進行最佳化，一個較低成本層針對不常存取進行最佳化，另一個非常低成本的層針對很少存取的資料進行最佳化。對於低廉的物件監控和自動化月費，S3 Intelligent-Tiering 會監控存取模式，並自動將連續 30 天未存取過的物件移至不常存取層。未存取達到 90 天之後，物件會移至 Archive Instant 存取層，而不會影響效能或操作額外負荷。

若要取得可在幾分鐘到幾小時內存取的最低資料儲存成本，請啟用封存功能以新增兩個額外存取層。您可以將物件分層至 Archive Access 層、Deep Archive Access 層或兩者。透過 Archive Access，S3 智慧型分層服務會將最少連續 90 天未存取過的物件移至 Archive Access 層。透過 Deep Archive Access，S3 智慧型分層服務會將最少連續 180 天未存取過的物件移至 Deep Archive Access 層。對於這兩個層，您可以根據需要配置無法存取的天數。

以下動作構成存取，可防止物件分層至 Archive Access 層或 Deep Archive Access 層：

- 透過 Amazon S3 主控台下載或複製物件。
- 使用 S3 批次複寫叫用 CopyObject、UploadPartCopy 或複寫物件。在這些情況下，複製或複寫操作的來源物件會分層。
- 呼叫 GetObject、PutObject、RestoreObject、CompleteMultipartUpload、ListParts 或 SelectObjectContent。

例如，如果在指定的無存取天數 (例如 180 天) 之前透過 `SelectObjectContent` 存取物件，則該動作會重設計時器。直到最後一個 `SelectObjectContent` 請求到達指定的天數為止，您的物件不會移至 Archive Access 層或 Deep Archive Access 層。

如果物件稍後在不常存取層或 Archive Instant 存取層受到存取，則它會自動將物件移回經常存取層。

下列動作會自動將物件從不常存取層或 Archive Instant 存取層移回經常存取層：

- 透過 Amazon S3 主控台下載或複製物件。
- 使用 Batch 複寫呼叫 CopyObject、UploadPartCopy 或複寫物件。在這些情況下，複製或複寫操作的來源物件會分層。

- 呼叫 GetObject、PutObject、RestoreObject、CompleteMultipartUpload 或 ListParts。

其他動作不會自動將物件從不常存取層至物件移回經常存取層。以下是此類動作的範例，而非明確清單：

- 呼叫 HeadObject、GetObjectTagging、PutObjectTagging、ListObjects、ListObjectsV2 或 ListObjectVersions。
- 呼叫 SelectObjectContent 並不構成將物件層級至頻繁存取層的存取權。此外，這不會防止物件從 Frequent Access 層下移至 Infrequent Access 層，然後再移到 Archive Instant Access 層。

您可以將 S3 Intelligent-Tiering 設定為新建立資料的預設儲存方案，方法是在您的 PutBucketIntelligentTieringConfiguration 請求標頭中指定 `INTELLIGENT-TIERING`。S3 Intelligent-Tiering 旨在提供的設計可提供 99.9% 的可用性和 99.999999999% 的耐用性。

> **Note**
>
> 若物件大小低於 128 KB，便不會受到監控且不符合自動分層的資格。較小的物件一律存放在經常存取層中。

S3 Intelligent-Tiering 存取層

下一節說明不同的自動和選用存取層。當物件在存取層之間移動時，儲存體類別會保持不變 (S3 Intelligent-Tiering)。

經常存取層 (自動)

這是在其生命週期中建立或轉換到 S3 Intelligent-Tiering 的任何物件的預設存取層。物件在被存取時，就會保留在此層中。Frequent Access 層可提供低延遲和高輸送量效能。

不常存取層 (自動)

如果物件連續 30 天未被存取，則物件會移至不常存取層。Infrequent Access 層可提供低延遲和高輸送量效能。

Archive Instant 存取層 (自動)

如果物件連續 90 天未被存取，則物件會移至 Archive Instant 存取層。Archive Instant Access 層可提供低延遲和高輸送量效能。

Archive Access 層 (選用)

S3 Intelligent-Tiering 可讓您選擇啟用封存存取層，以取得可以非同步存取的資料。啟用後，封存存取層會自動封存至少連續 90 天未存取過的物件。您可以將封存的最後存取時間延長至最多 730 天。封存存取層的效能與 S3 Glacier Flexible Retrieval 儲存類別相同。

此存取層的標準擷取時間範圍為 3-5 小時。如果您使用 S3 批次操作啟動還原請求，還原會在幾分鐘內開始。如需擷取選項和時間的詳細資訊，請參閱 the section called “從 S3 Intelligent-Tiering 封存存取層和 Deep Archive 存取層還原物件”。

> **Note**
>
> 只有在您想要略過 Archive Instant 存取層時，才啟用封存存取層 90 天。Archive Access 層提供略低的儲存成本，擷取時間在幾分鐘到幾小時之間。Archive Instant Access 層可提供毫秒存取和高輸送量效能。

Deep Archive Access 層 (選用)

S3 Intelligent-Tiering 可讓您選擇啟用 Deep Archive 存取層，以取得可以非同步存取的資料。啟用後，Deep Archive 存取層會自動封存至少連續 180 天未存取過的物件。您可以將封存的最後存取時間延長至最多 730 天。Deep Archive Access 層的效能與 S3 Glacier Deep Archive 儲存方案相同。

此存取層中的物件標準擷取會在 12 小時內進行。如果您使用 S3 批次操作啟動還原請求，還原會在 9 小時內開始。如需擷取選項和時間的詳細資訊，請參閱 the section called “從 S3 Intelligent-Tiering 封存存取層和 Deep Archive 存取層還原物件”。

> **Note**
>
> 只有當應用程式可以非同步存取物件時，才可啟用 Archive Access 和 Deep Archive Access 層。如果您要擷取的物件存放在 Archive Access 或 Deep Archive Access 層中，則必須先使用 `RestoreObject` 操作還原物件。

使用 S3 Intelligent-Tiering

您可以使用 S3 Intelligent-Tiering 儲存方案，以自動最佳化儲存成本。當存取模式改變時，S3 Intelligent-Tiering 可在精密物件層級上兩個存取層之間移動資料，以自動節省成本的雲端儲存。對於可

非同步存取的資料，您可以選擇使用 AWS Management Console、AWS CLI 或 Amazon S3 API 啟用 S3 Intelligent-Tiering 儲存方案內的自動封存功能。

將資料移至 S3 Intelligent-Tiering

有兩種方法可以將資料移至 S3 Intelligent-Tiering。您可以在 `x-amz-storage-class` 標頭中指定 `INTELLIGENT_TIERING` 以直接將 PUT 資料轉換為 S3 Intelligent-Tiering，或是設定 S3 生命週期組態，以將物件從 S3 Standard 或 S3 標準 - 不常存取轉換為 S3 Intelligent-Tiering。

使用 Direct Put 將資料上傳至 S3 Intelligent-Tiering

當您使用 PUT API 操作將物件上傳至 S3 Intelligent-Tiering 儲存方案時，您可在 `x-amz-storage-class` 請求標頭中指定 S3 Intelligent-Tiering。

以下請求會將映像 `my-image.jpg` 存放在 `myBucket` 儲存貯體中。此要求會使用 `x-amz-storage-class` 標頭，以請求使用 S3 Intelligent-Tiering 儲存方案存放物件。

Example

```
PUT /my-image.jpg HTTP/1.1
Host: myBucket.s3.<Region>.amazonaws.com (http://amazonaws.com/)
Date: Wed, 1 Sep 2021 17:50:00 GMT
Authorization: authorization string
Content-Type: image/jpeg
Content-Length: 11434
Expect: 100-continue
x-amz-storage-class: INTELLIGENT_TIERING
```

使用 S3 生命週期將資料從 S3 Standard 或 S3 標準 - 不常存取轉換為 S3 Intelligent-Tiering

您可以在 S3 生命週期組態中新增規則，命令 Amazon S3 將物件從一個儲存方案轉換至另一個儲存方案。如需有關支援的轉換和相關限制的資訊，請參閱使用 S3 生命週期轉換物件。

您可以在儲存貯體或字首層級指定 S3 生命週期組態。在此 S3 生命週期組態規則中，篩選條件指定了一個金鑰字首 (key prefix) (`documents/`)。因此，規則將會套用至其金鑰名稱字首為 `documents/` 的物件，例如 `documents/doc1.txt` 與 `documents/doc2.txt`。該規則會指定 `Transition` 動作，指示 Amazon S3 在建立物件的 0 天後，將其轉換為 S3 Intelligent-Tiering 儲存方案。在此情況下，物件在建立之後 UTC 時間的午夜，即符合轉換至 S3 Intelligent-Tiering 的條件。

Example

```
<LifecycleConfiguration>
```

```
  <Rule>
    <ID>ExampleRule</ID>
    <Filter>
       <Prefix>documents/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>INTELLIGENT_TIERING</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

啟用 S3 Intelligent-Tiering Archive Access 和 Deep Archive Access 層

若要為在數分鐘至數小時內存取的資料取得最低儲存成本，您可以使用 AWS Management Console、AWS CLI 或 Amazon S3 API 建立儲存貯體、字首或物件標籤層級組態，進而啟用一個或兩個封存存取層。

使用 S3 主控台

若要啟用 S3 Intelligent-Tiering 自動封存

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在 Buckets (儲存貯體) 清單中，選擇您所需的儲存貯體名稱。
3. 選擇 Properties (屬性)。
4. 導覽至 S3 Intelligent-Tiering Archive configurations (S3 Intelligent-Tiering Archive 組態) 區段並選擇 Create configuration (建立組態)。
5. 在 Archive configuration settings (Archive 組態設定) 區段中，為您的 S3 Intelligent-Tiering Archive 組態指定一個描述性組態名稱。
6. 在 Choose a configuration scope (選擇組態範圍) 下，選擇要使用的組態範圍。除此之外，您可以使用共用字首、物件標籤或兩者的組合，將組態範圍限制在儲存貯體內的指定物件。

   a. 若要限制組態的範圍，請選取 Limit the scope of this configuration using one or more filters (使用一或多個篩選條件限制此組態的範圍)。

   b. 若要使用單一字首限制組態的範圍，請在 Prefix (字首) 下輸入字首。

   c. 若要使用物件標籤限制組態的範圍，請選取 Add tag (新增標籤) 並輸入金鑰的值。

7. 在 Status (狀態) 下，選取 Enable (啟用)。
8. 在 Archive settings (封存設定) 區段中，選取一個或兩個要啟用的 Archive Access 層。
9. 選擇 Create (建立)。

使用 AWS CLI

您可以使用下列 AWS CLI 命令來管理 S3 Intelligent-Tiering 組態：

- delete-bucket-intelligent-tiering-configuration
- get-bucket-intelligent-tiering-configuration
- list-bucket-intelligent-tiering-configurations
- put-bucket-intelligent-tiering-configuration

如需設定 AWS CLI 的說明，請參閱 使用 AWS CLI 來透過 Amazon S3 進行開發。

當您使用 AWS CLI 時，您無法將組態指定為 XML 檔案。您必須改為指定 JSON。下列是 XML S3 Intelligent-Tiering 組態範例與可在 AWS CLI 命令中指定的對等 JSON。

下列範例會將 S3 Intelligent-Tiering 組態放入指定的儲存貯體中。

Example **put-bucket-intelligent-tiering-configuration**

JSON

```
{
  "Id": "string",
  "Filter": {
    "Prefix": "string",
    "Tag": {
      "Key": "string",
      "Value": "string"
    },
    "And": {
      "Prefix": "string",
      "Tags": [
        {
          "Key": "string",
          "Value": "string"
        }
```

```
        ...
      ]
    }
  },
  "Status": "Enabled"|"Disabled",
  "Tierings": [
    {
      "Days": integer,
      "AccessTier": "ARCHIVE_ACCESS"|"DEEP_ARCHIVE_ACCESS"
    }
    ...
  ]
}
```

XML

```
PUT /?intelligent-tiering&id=Id HTTP/1.1
Host: Bucket.s3.amazonaws.com
<?xml version="1.0" encoding="UTF-8"?>
<IntelligentTieringConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Id>string</Id>
   <Filter>
      <And>
         <Prefix>string</Prefix>
         <Tag>
            <Key>string</Key>
            <Value>string</Value>
         </Tag>
         ...
      </And>
      <Prefix>string</Prefix>
      <Tag>
         <Key>string</Key>
         <Value>string</Value>
      </Tag>
   </Filter>
   <Status>string</Status>
   <Tiering>
      <AccessTier>string</AccessTier>
      <Days>integer</Days>
   </Tiering>
   ...
</IntelligentTieringConfiguration>
```

使用 PUT API 操作

您可以針對指定的儲存貯體使用 PutBucketIntelligentTieringConfiguration 操作，且每個儲存貯體最多 1,000 個 S3 Intelligent-Tiering 組態。您可以使用共用字首或物件標籤，來定義儲存貯體內的哪些物件適用於封存存取層。使用共用字首或物件標籤，可讓您符合特定商業應用程式、工作流程或內部組織。您也可以靈活地啟用 Archive Access 層、Deep Archive Access 層，或兩者。

開始使用 S3 智慧型分層服務

若要進一步了解如何使用 S3 智慧型分層服務，請參閱教學課程：開始使用 S3 智慧型分層服務。

管理 S3 Intelligent-Tiering

S3 Intelligent-Tiering 儲存類別可在三個低延遲和高輸送量存取層中自動節省儲存成本。其還提供選用封存功能，協助您為可在數分鐘至數小時內存取的雲端資料取得最低儲存成本。S3 Intelligent-Tiering 儲存類別支援所有 Amazon S3 功能，包括：

- S3 庫存，用於驗證物件的存取層
- S3 複寫，用於將資料複寫到任何 AWS 區域
- S3 Storage Lens，用於檢視儲存用量和活動指標
- 伺服器端加密，用於保護物件資料
- S3 物件鎖定，用於防止意外刪除資料
- AWS PrivateLink，用於透過虛擬私有雲端 (VPC) 中的私有端點存取 Amazon S3

識別儲存在哪些 S3 Intelligent-Tiering 存取層物件中

若要取得物件及其對應中繼資料 (包括其 S3 智慧型分層存取層) 的清單，您可以使用。the section called “管理清查”S3 庫存提供 CSV、ORC 或 Parquet 輸出檔案，當中列出您的物件及其對應中繼資料。您可以每天或每週接收 Amazon S3 儲存貯體或共用字首的這些庫存報告。(共用字首是指名稱以共用字串為開頭的物件。)

在 S3 Intelligent-Tiering 中檢視物件的封存狀態

您可以設定 S3 事件通知，以便在 S3 Intelligent-Tiering 儲存類別中的物件移至 Archive Access 層或 Deep Archive Access 層時收到通知。如需詳細資訊，請參閱啟用事件通知。

Amazon S3 可以將事件通知發佈到 Amazon Simple Notification Service (Amazon SNS) 主題、Amazon Simple Queue Service (Amazon SQS) 佇列或 AWS Lambda 函數。如需詳細資訊，請參閱 Amazon S3 事件通知。

以下是 Amazon S3 傳送以發佈 s3:IntelligentTiering 事件的訊息範例。如需詳細資訊，請參閱 the section called “事件訊息結構”。

```
{
   "Records":[
      {
         "eventVersion":"2.3",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"IntelligentTiering",
         "userIdentity":{
            "principalId":"s3.amazonaws.com"
         },
         "requestParameters":{
            "sourceIPAddress":"s3.amazonaws.com"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"mybucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::mybucket"
            },
            "object":{
               "key":"HappyFace.jpg",
               "size":1024,
               "eTag":"d41d8cd98f00b204e9800998ecf8427e",
            }
         },
         "intelligentTieringEventData":{
            "destinationAccessTier": "ARCHIVE_ACCESS"
            }
      }
   ]
```

```
}
```

您也可以使用 HEAD 物件請求，以檢視物件的封存狀態。如果物件儲存在 S3 Intelligent-Tiering 儲存類別中且位於其中一個封存層，則 HEAD 物件回應將會顯示目前的封存層。若要顯示封存層，請求會使用 x-amz-archive-status 標頭。

下列 HEAD 物件請求會傳回物件的中繼資料 (此案例中為 *my-image.jpg*)。

Example

```
HEAD /my-image.jpg HTTP/1.1
Host: bucket.s3.region.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:02236Q3V0RonhpaBX5sCYVf1bNRuU=
```

您也可以使用 HEAD 物件請求監控 `restore-object` 請求的狀態。如果封存還原正在進行中，HEAD 物件回會包含 x-amz-restore 標頭。

以下 HEAD 物件回應範例顯示使用 S3 Intelligent-Tiering 封存的物件，且還原請求正在進行中。

Example

```
HTTP/1.1 200 OK
x-amz-id-2: FSVaTMjrmBp3Izs1NnwBZeu7M19iI8UbxMbi0A8AirHANJBo+hEftBuiESACOMJp
x-amz-request-id: E5CEFCB143EB505A
Date: Fri, 13 Nov 2020 00:28:38 GMT
Last-Modified: Mon, 15 Oct 2012 21:58:07 GMT
ETag: "1accb31fcf202eba0c0f41fa2f09b4d7"
x-amz-storage-class: 'INTELLIGENT_TIERING'
x-amz-archive-status: 'ARCHIVE_ACCESS'
x-amz-restore: 'ongoing-request="true"'
x-amz-restore-request-date: 'Fri, 13 Nov 2020 00:20:00 GMT'
Accept-Ranges: bytes
Content-Type: binary/octet-stream
Content-Length: 300
Server: AmazonS3
```

從 S3 Intelligent-Tiering 封存存取層和 Deep Archive 存取層還原物件

若要存取 S3 智慧型分層封存存取和深層封存存取層中的物件，您必須啟動還原請求，然後等待物件移入頻繁存取層。如需封存物件的詳細資訊，請參閱the section called “使用封存的物件”。

當您從 Archive Access 層或 Deep Archive Access 層還原物件時，物件會移回不經常存取層。之後，如果物件連續 30 天未被存取，則會自動移至不常存取層。然後，在至少連續 90 天無存取的情況下，物件會移至 Archive Access 層。在至少連續 180 天無存取的情況下，物件會移至 Deep Archive Access 層。如需詳細資訊，請參閱 the section called “S3 Intelligent-Tiering 的運作方式”。

您可以使用 Amazon S3 主控台、S3 Batch 操作、Amazon S3 REST API、 AWS 開發套件或 AWS Command Line Interface (AWS CLI) 來還原存檔物件。如需更多詳細資訊，請參閱 the section called “使用封存的物件”。

管理儲存生命週期

若要管理物件以便在整個生命週期中以符合成本效益的方式存放物件，請建立 Amazon S3 生命週期組態。Amazon S3 生命週期組態是一組規則，用於定義 Amazon S3 套用至一組物件的動作。有兩種類型的動作：

- 轉換動作 – 定義物件何時轉換成另一個儲存類別。例如，您可以選擇在建立物件後的 30 天，將物件轉換為 S3 標準 – IA 儲存類別，或者在建立物件一年後，將物件封存到 S3 Glacier Flexible Retrieval 儲存類別。如需詳細資訊，請參閱 使用 Amazon S3 儲存體方案。

  這些是與生命週期轉換請求相關聯的成本。如需定價資訊，請參閱 Amazon S3 定價。
- 過期動作 – 這些動作會定義物件何時過期。Amazon S3 會為您刪除已過期的物件。

  生命週期過期成本，取決於您選擇物件的過期時間。如需詳細資訊，請參閱 即將到期的物件。

> **Important**
>
> 您無法使用儲存貯體政策來防止 S3 生命週期規則刪除或轉換。例如，即使儲存貯體政策拒絕所有主體的所有動作，S3 生命週期組態仍可正常運作。

現有物件和新物件

當您新增儲存貯體的生命週期組態時，組態規則會套用至現有物件以及稍後新增的物件。例如，如果您今天新增生命週期組態規則，其中的到期動作會導致物件在建立後 30 天到期，Amazon S3 將會排入佇列移除任何超過 30 天的現有物件。

帳單變更

如果物件符合生命週期動作的資格，以及 Amazon S3 傳輸或讓物件過期之間有任何延遲，則只要物件符合生命週期動作的資格，便會即刻套用帳單變更。例如，如果物件排定到期，且 Amazon S3 不會立即使該物件過期，則在到期時間之後不會向您收取儲存費用。

此行為的例外狀況為，如果您擁有轉換至 S3 Intelligent-Tiering 儲存類別的生命週期規則。在此情況下，直到物件轉換至 S3 Intelligent-Tiering 儲存類別後，才會發生計費變更。

如需 S3 生命週期規則的詳細資訊，請參閱 生命週期組態元素。

監視生命週期規則的效果

若要監視使用中生命週期規則所做更新的影響，請參閱the section called “如何監視生命週期規則所採取的動作？”。

管理物件生命週期

為已妥善定義生命週期的物件，定義 S3 生命週期組態規則。例如：

- 如果您將定期日誌上傳到儲存貯體，您的應用程式可能需要使用它們一週或一個月。之後，您可能會想刪除它們。
- 某些文件在一段有限的期間內會經常受到存取。而在該期間之後，存取它們的頻率很低。在某些時候，您可能不需要即時存取它們，但您的組織或法規可能要求您將其封存一段特定時間。之後，您可以刪除它們。
- 您可能會將某些類型的資料上傳到 Amazon S3，主要為目的檔案用。例如，您可以使用它來封存數位媒體、財務及醫療保健記錄、原始基因序列資料、長期資料庫備份，以及為遵守法規而所必須保留的資料。

使用 S3 生命週期組態規則，您可以指示 Amazon S3 將物件轉換為較便宜的儲存體類別、封存或刪除物件。

建立生命週期組態

S3 生命週期組態是一個 XML 檔案，其中包含一組具備預先定義動作的規則，您可請求 Amazon S3 在物件生命週期內，對其執行這些動作。

您可以使用 Amazon S3 主控台、REST API、 AWS 開發套件和 AWS Command Line Interface (AWS CLI) 來建立生命週期組態。如需詳細資訊，請參閱 在值區上設定生命週期組態。

Amazon S3 提供一組 REST API 操作，可用於管理儲存貯體上的生命週期組態。Amazon S3 會將組態存放為附加到儲存貯體的生命週期子資源。如需詳細資訊，請參閱以下：

- PutBucketLifecycleConfiguration
- GetBucketLifecycleConfiguration
- DeleteBucketLifecycle

如需建立生命週期組態的詳細資訊，請參閱下列主題：

主題

- 使用 Amazon S3 生命週期轉換物件
- 即將到期的物件
- 在值區上設定生命週期組態
- 生命週期及其他儲存貯體組態
- 設定生命週期事件通知
- 生命週期組態元素
- S3 生命週期組態範例

使用 Amazon S3 生命週期轉換物件

您可以在 S3 生命週期組態中新增規則，命令 Amazon S3 將物件轉換至另一個 Amazon S3 儲存類別。如需儲存體方案的詳細資訊，請參閱「使用 Amazon S3 儲存體方案」。您可以透過這種方式使用 S3 生命週期組態的一些範例，範例包括以下內容：

- 當您知道不常存取的物件時，您可以將它們轉換為 S3 標準 – IA 儲存類別。
- 您可能想要將不需要即時存取的物件存檔至 S3 Glacier 彈性擷取或 S3 Glacier 深層存檔儲存類別。

現有物件和新物件

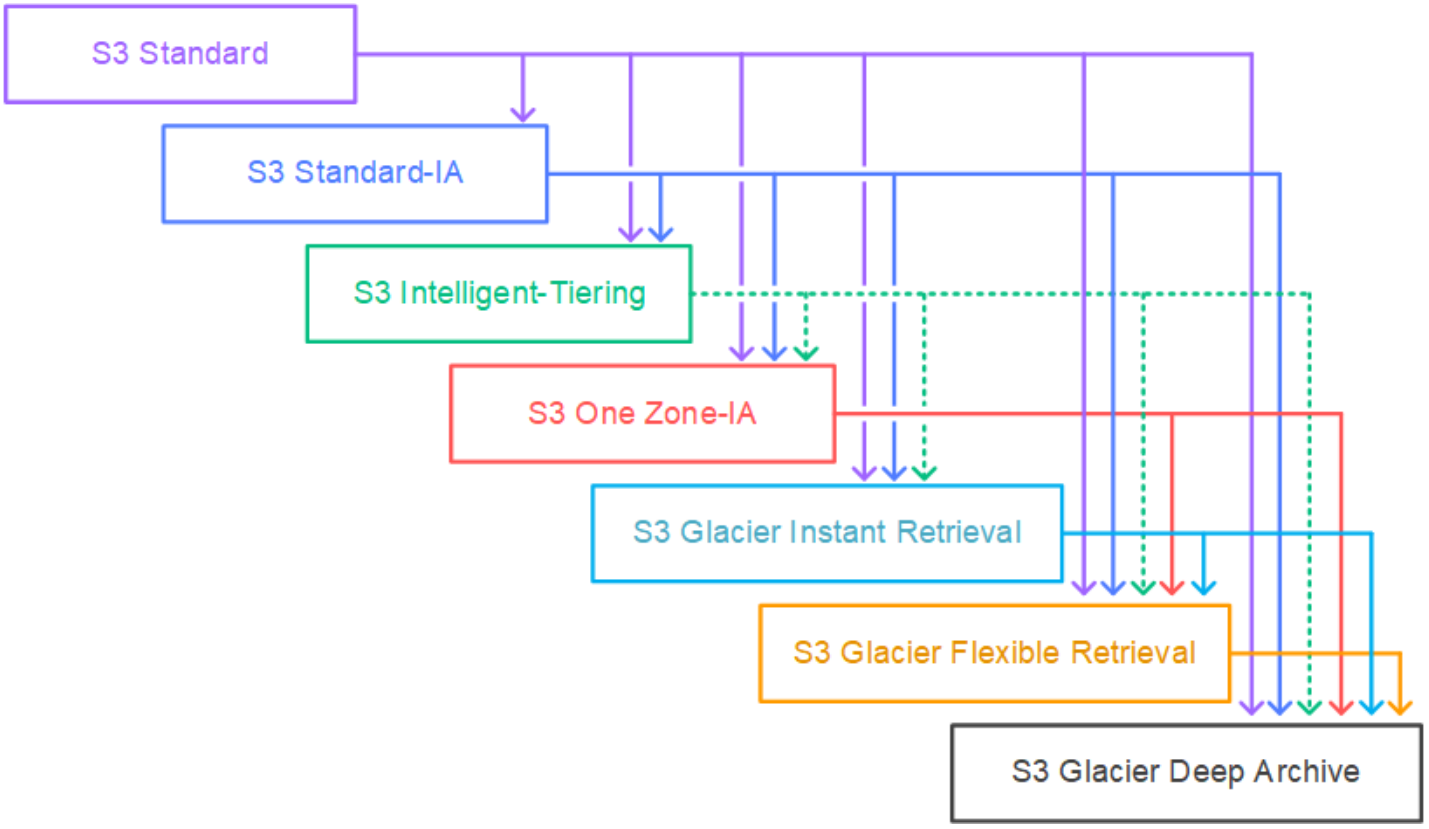
當您新增儲存貯體的生命週期組態時，組態規則會套用至現有物件以及稍後新增的物件。例如，如果您今天使用轉換動作新增生命週期組態規則，使具有特定前置詞的物件在建立後 30 天轉換為不同的儲存類別，Amazon S3 將會排入佇列，轉換超過 30 天且具有指定前置碼的任何現有物件。

> **Important**
>
> 您無法使用儲存貯體政策來防止 S3 生命週期規則刪除或轉換。例如，即使儲存貯體政策拒絕所有主體的所有動作，S3 生命週期組態仍可正常運作。

支援的轉換及相關限制

您可於 S3 生命週期組態中，定義將物件從一個儲存類別轉換為另一個儲存類別的規則，並存於儲存體成本。當您不清楚物件的存取模式時，或如果您的存取模式會隨時間變更，您可以將物件轉換為 S3 Intelligent-Tiering 儲存類別，自動節省成本。如需儲存體方案的資訊，請參閱「使用 Amazon S3 儲存體方案」。

Amazon S3 支援瀑布模型以在儲存類別間轉換，如下圖所示。



支援的生命週期轉換

Amazon S3 支援使用 S3 生命週期組態在儲存類別間轉換生命週期。

您可以從下列項目轉換：

- S3 Standard 儲存類別轉換為任何其他儲存類別。
- S3 標準-IA 儲存類別到 S3 Intelligent-Tiering、S3 單區域-IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別。
- S3 Intelligent-Tiering 儲存類別到 S3 單區域-IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別。

> **Note**
>
> 將物件從 S3 智慧型分層儲存類別轉換到 S3 單區域 — IA 和某些 S3 Glacier 儲存類別時，有一些例外狀況。如需詳細資訊，請參閱 the section called “不支援的生命週期轉換”。

- S3 單區域-IA 儲存類別到 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別。
- S3 Glacier Instant Retrieval 儲存類別到 S3 Glacier Flexible Retrieval，或 S3 Glacier Deep Archive 儲存類別。
- S3 Glacier Flexible Retrieval 儲存類別到 S3 Glacier Deep Archive 儲存類別。
- S3 Glacier Deep Archive 儲存類別的任何儲存類別。

> **Note**
>
> 生命週期轉換不會產生資料擷取費用。不過，使用、或生命週期規則將資料移至任何 S3 儲存類別時 PUTCOPY，需要支付每個請求擷取費用。將物件移至任何儲存類別之前，請考慮擷取或轉換成本。如需成本考量的詳細資訊，請參閱 Amazon S3 定價。

不支援的生命週期轉換

Amazon S3 不支援下列任何生命週期轉換。

您無法從以下內容轉換：

- 任何儲存類別轉換為 S3 Standard 儲存類別。
- 任何儲存類別到低冗餘儲存 (RRS) 類別。
- S3 單區域 – IA 儲存類別到 S3 Intelligent-Tiering、S3 標準 – IA、或 S3 Glacier Instant Retrieval 儲存類別。
- S3 智慧型分層儲存類別 (所有層) 至 S3 標準 — IA 儲存類別。
- S3 智慧型分層儲存類別將即時存取層封存至 S3 單區域 — IA。
- S3 智慧型分層儲存類別存檔存取層，可存取 S3 單區域 — IA 或 S3 Glacier 即時擷取。
- S3 智慧型分層儲存類別深度存檔存取層，可存取 S3 單區域 — IA、S3 Glacier 即時擷取或 S3 Glacier 彈性擷取。

限制

生命週期儲存體方案轉換有下列限制：

物件大小和從 S3 Standard 或 S3 標準 – IA 轉為 S3 Intelligent-Tiering 、S3 標準 – IA 或 S3 單區域 – IA 的轉換

當您將物件從 S3 Standard 或 S3 標準 – IA 儲存類別轉為 S3 Intelligent-Tiering 、S3 標準 – IA 或 S3 單區域 – IA 時，系統會套用下列物件大小限制：

- 較大物件 – 對於下列轉換，轉換大型物件會有成本效益：
  - 從 S3 Standard 或 S3 標準 – IA 儲存類別轉換為 S3 Intelligent-Tiering 。
  - 從 S3 Standard 儲存類別轉為 S3 標準 – IA 或 S3 單區域 – IA。
- 小於 128 KiB 的物件 — 對於下列轉換，Amazon S3 不會轉換小於 128 KiB 的物件：
  - 從 S3 Standard 或 S3 標準 – IA 儲存類別到 S3 Intelligent-Tiering 或 S3 Glacier Instant Retrieval。
  - 從 S3 Standard 儲存類別轉為 S3 標準 – IA 或 S3 單區域 – IA。

> **Note**
>
> 您可以根據物件大小篩選生命週期規則。

> **Important**
>
> 當您在 S3 生命週期組態中有多個規則時，一個物件可能會在同一天符合多個 S3 生命週期動作的資格。在這種情況下，Amazon S3 遵循以下一般規則：
>
> - 永久刪除優先於轉換。
> - 轉移優先於刪除標記的建立。
> - 當物件同時符合 S3 Glacier Flexible Retrieval 和 S3 標準 – IA (或 S3 單區域 – IA) 轉換的資格時，Amazon S3 會選擇 S3 Glacier Flexible Retrieval 轉換。
>
> 如需範例，請參閱 範例 5：篩選條件重疊、生命週期動作相衝突，以及 Amazon S3 對為進行版本控制的儲存貯體所採取的動作。

轉換至 S3 Standard-IA 或 S3 One Zone-IA 的最少天數

將物件轉換為 S3 Standard-IA 或 S3 One Zone-IA 之前，您必須將它們儲放在 Amazon S3 中至少 30 天。例如，您無法在建立一則生命週期規則後的隔天就將其轉換為 S3 標準 – IA 儲存類別。Amazon S3 在前 30 天內不支援此轉換，因為新的物件通常會比 S3 標準 – IA 或 S3 單區域 – IA 更頻繁地被存取或刪除。

相同地，若要轉換非最新版本的物件 (在控制儲存貯體版本)，只能在物件脫離目前版本的 30 天後將其轉換為 S3 標準 – IA 或 S3 單區域 – IA 儲存。如需所有儲存類別的最短儲存持續時間清單，請參閱比較 Amazon S3 儲存方案。

S3 標準 – IA 和 S3 單區域 – IA 的最低 30 天儲存費用

S3 標準 – IA 和 S3 單區域 – IA 儲存類別具有最短 30 天的儲存費用。因此，當 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 轉換發生在 S3 標準-IA 或 S3 單區域-IA 轉換後未滿 30 天時，您無法為 S3 標準-IA 或 S3 單區域-IA 轉換或 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 轉換指定生命週期規則。

當您指定從 S3 標準 – IA 儲存轉換到 S3 單區域 – IA 時，同樣適用 30 天最小儲存天數方案。您可以指定兩個規則完成此操作，但您仍需要支付最低的儲存體費用。如需成本考量的詳細資訊，請參閱 Amazon S3 定價。

管理物件的完整生命週期

您可以結合這些 S3 生命週期操作，來管理物件的完整生命週期。例如，假設您建立的物件具備了定義妥善的生命週期。一開始時，物件在 30 天內會頻繁受到存取。然後，物件在長達 90 天內，不會頻繁存取。之後，不再需要這些物件，因此，您可以選擇封存或刪除。

在本案例中，您可以建立 S3 生命週期規則，在規則中指定轉換至 S3 Intelligent-Tiering 、S3 標準 – IA 或 S3 單區域 – IA 儲存的初始化動作，另外指定轉換至 S3 Glacier Flexible Retrieval 儲存的動作以供封存，還有指定過期動作。將物件從某個儲存類別移至另一個儲存類別時，可以節省儲存成本。如需成本考量的詳細資訊，請參閱 Amazon S3 定價。

轉換為 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 儲存類別 (物件封存)

透過使用 S3 生命週期組態，您可以將物件轉換為 S3 Glacier 彈性擷取或 S3 Glacier 深度存檔儲存類別以進行存檔。選擇 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別時，您的物件會保留在 Amazon S3 中。您無法透過個別的 Amazon S3 Glacier 服務直接存取物件。如需有關 S3 Glacier 的更多一般資訊，請參閱《Amazon S3 Glacier 開發人員指南》中的什麼是 Amazon S3 Glacier。

封存物件之前，請先檢閱下列章節以了解相關考量。

一般考量

以下為進行物件封存之前，建議先行考量的一般事宜：

- 在儲存體方案轉換程序中，加密的物件仍會維持加密。
- 存放在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別的物件無法提供即時存取。

  封存的物件也是 Amazon S3 物件，但在進行存取之前，必須先還原一份暫存複本。還原的物件複本只能在您在還原要求中指定的持續時間內使用。之後，Amazon S3 刪除臨時複本，並且該物件仍然封存在 S3 Glacier Flexible Retrieval 中。

  您可以使用 Amazon S3 主控台或在程式碼中使用 AWS 開發套件包裝函式庫或 Amazon S3 REST API 以程式設計方式還原物件。如需詳細資訊，請參閱 還原已封存的物件。
- 存放在 S3 Glacier Flexible Retrieval 儲存類別中的物件只能轉換到 S3 Glacier Deep Archive 儲存類別。

  您僅可以使用 S3 生命週期組態規則，將物件的儲存類別從 S3 Glacier Flexible Retrieval 轉換為 S3 Glacier Deep Archive 儲存類別。若希望將在 S3 Glacier Flexible Retrieval 中存放之物件的儲存類別變更為 S3 Glacier Deep Archive 以外的儲存類別，您必須先使用還原操作製作該物件的暫存複本。然後使用複製操作，覆寫指定 S3 Standard、S3 Intelligent-Tiering 、S3 標準 – IA、S3 單區域 – IA 或低冗餘為儲存類別的物件。
- 將物件轉換為 S3 Glacier Deep Archive 儲存類別只能是單向。

  您無法使用 S3 生命週期組態規則，將物件的儲存類別從 S3 Glacier Deep Archive 轉換為任何其他儲存類別。若希望將已存檔物件的儲存體方案變更為其他儲存體方案，您必須先使用還原操作製作該物件的暫存複本。然後使用複製操作，覆寫指定 S3 Standard、S3 Intelligent-Tiering、S3 標準 – IA、S3 單區域 – IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 或低冗餘儲存為儲存類別的物件。

> **Note**
>
> Amazon S3 主控台不支援對 Amazon S3 Glacier Flexible Retrive 或 S3 Glacier Deep Archive 儲存類別中的物件進行還原物件的複製操作。對於這種類型的複製操作，請使用 AWS Command Line Interface (AWS CLI)、 AWS SDK 或 REST API。

存放在 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 儲存類別的物件僅能透過 Amazon S3 看見和取得。它們無法透過個別的 Amazon S3 Glacier服務取得。

這些是 Amazon S3 物件，您只能透過使用 Amazon S3 主控台或 Amazon S3 API 加以存取。您無法透過個別的 Amazon S3 Glacier 主控台或 Amazon S3 Glacier API 存取已封存的物件。

成本考量

若預計會將不常存取的資料封存數個月或數年之久，則 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 儲存類別可以降低儲存費用。不過，若要確保 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別適合您，請考慮下列項目：

- 儲存體經常性費用 – 當您將物件轉換為 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別時，會為每個物件新增固定大小的儲存空間，以容納管理物件所需要的中繼資料。
  - Amazon S3 會為每個封存至 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 的物件，使用 8 KB 的儲存體空間供物件的名稱及其他中繼資料使用。Amazon S3 會存於此中繼資料，以便於可利用 Amazon S3 API 取得封存物件的即時清單。如需詳細資訊，請參閱 Get 儲存貯體 (列出物件)。將就這項額外的儲存體向您收取 S3 Standard 費率。
  - 對於封存至 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 的每個物件，Amazon S3 為索引和相關中繼資料新增 32 KB 的儲存空間。為了能識別及還原您的物件，將需要這項額外的資料。系統會以 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 費率向您收取此額外儲存體的費用。

  若要封存小型物件，建議您將這些儲存體費用納入考量。此外，建議您將多個的小型物件彙總為幾個的大型物件，以降低經常性成本。
- 預計封存物件的天數 – S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 是長期性的封存解決方案。S3 Glacier Flexible Retrieval 儲存類別的最低儲存體持續期間為 90 天，S3 Glacier Deep Archive 則為 180 天。如果您刪除的物件封存的時間超過最低儲存體持續期間，即可免費刪除封存至 Amazon S3 Glacier 的資料。如果在最低持續期間內刪除或覆寫封存的物件，Amazon S3 會依比例向您索取提早刪除的費用。如需提前刪除費用的詳細資訊，請參閱「刪除 Amazon S3 Glacier 中不到 90 天的物件時，如何收費？」 問題 (位於 Amazon S3 常見問答集)。
- S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 轉換請求費用 – 轉換至 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別的每個物件，都會形成一個轉換請求。而每項要求都會產生一筆費用。若預計會轉換大量的物件，建議您將要求成本納入考慮。如果您要封存包含小型物件的混合物件 (尤其是 128KB 以下的物件)，建議您使用生命週期物件大小篩選器，從轉

換中篩選出小型物件，以降低請求成本。S3 冰川彈性擷取和 S3 Glacier Deep Archive 不會自動封鎖 128KB 以下物件的轉換。

- S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 資料還原費用 – S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 專門針對不常存取的長期封存資料所設計。如需資料還原費用的詳細資訊，請參閱「從 Amazon S3 Glacier 擷取資料需要多少費用？」 問題 (位於 Amazon S3 常見問答集)。如需如何從 Amazon S3 Glacier 還原資料的資訊，請參閱 還原已封存的物件。

當您使用 S3 生命週期管理將物件封存到 Amazon S3 Glacier時，Amazon S3 會以非同步方式轉換這些物件。S3 生命週期組態規則設定的轉換日期與實際轉換的日期之間，可能會有出現延遲的狀況。收取的 Amazon S3 Glacier 費用取決於規則中指定的轉換日期。如需詳細資訊，請參閱 Amazon S3 常見問題集中的「Amazon S3 Glacier」一節。

Amazon S3 產品詳細資訊頁面提供封存 Amazon S3 物件的定價資訊及計算範例。如需詳細資訊，請參閱下列主題：

- 「將 Amazon S3 物件封存到 Amazon S3 Glacier 時，如何計算儲存費？」 (位於 Amazon S3 常見問答集)。
- 「刪除 Amazon S3 Glacier 中不到 90 天的物件時，如何收費？」 (位於 Amazon S3 常見問答集)。
- 「從 Amazon S3 Glacier 擷取資料的費用為何？」 (位於 Amazon S3 常見問答集)。
- Amazon S3 定價提供不同儲存類別的儲存體費用。

還原存檔物件

封存的物件無法即時存取。必須先啟動還原要求，並等到暫存複本在要求中指定的期間內可供使用時，才可進行存取。收到還原物件的暫存複本之後，物件的儲存方案將會維持為 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive。(A HeadObject或 GetObjectAPI 操作請求將 S3 冰川彈性擷取或 S3 Glacier Deep Archive 作為儲存類別傳回。)

> **Note**
>
> 當您還原封存時，需要同時支付封存物件 (依 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 費率計價) 及暫時復原的複本的費用 (S3 標準儲存費率)。如需定價的資訊，請參閱 Amazon S3 定價。

您可以程式設計方式或使用 Amazon S3 主控台，還原物件複本。Amazon S3 同一時間只會為每個物件處理一項還原要求。如需詳細資訊，請參閱「還原已封存的物件」。

即將到期的物件

當物件根據其生命週期組態到達其生命週期結束時，Amazon S3 會根據儲存貯體所處的 S3 版本控制狀態採取動作。

- 非版本化儲存貯體 — Amazon S3 會將物件排入佇列進行移除，並以非同步方式移除物件，永久移除該物件。
- 已啟用版本控制的儲存貯體 - 如果目前的物件版本不是刪除標記，則 Amazon S3 會新增具有唯一版本 ID 的刪除標記。如此會讓目前的版本成為非目前的版本，而刪除標記成為目前版本。
- 暫停版本控制的儲存貯體 - Amazon S3 會建立以 null 為版本 ID 的刪除標記。此刪除標記會以 null 版本 ID 取代版本階層中所有的物件版本。這是刪除物件最有效的方法。

對於版本控制的儲存貯體 (亦即，已啟用版本控制或暫停版本控制的儲存貯體)，有數個考量，引導 Amazon S3 如何處理過期動作。對於已啟用版本控制或暫停版本控制的儲存貯體，適用下列情況：

- 物件過期只適用於物件的目前版本 (其對非目前的物件版本沒有影響)。
- 當有一或多個物件版本，且刪除標記為目前的版本時，Amazon S3 不會採取任何動作。
- 若目前的物件版本是唯一的物件版本，同時也是刪除標記 (亦稱為過期物件刪除標記，這會刪除所有的物件版本，只留下刪除標記)，Amazon S3 會移除過期物件刪除標記。您也可以使用過期動作，指示 Amazon S3 移除任何過期物件刪除標記。如需範例，請參閱 範例 7：移除過期物件刪除標記。
- 您可以使用`NoncurrentVersionExpiration`動作元素指示 Amazon S3 永久刪除非最新版本的物件。無法復原這些已刪除的物件。您可以根據物件變成非目前的特定天數來建立此到期日。除了天數之外，您還可以提供要保留的最大封存版本數目 (介於 1 到 100 之間)。此值會指定必須有多少個較新的非目前版本存在，Amazon S3 才可對指定的版本執行相關聯的動作。若要指定封存版本的最大數目，您還必須提供`Filter`元素。如果您未指定`Filter`元素，Amazon S3 會在您提供最大數量的非最新版本時產生`InvalidRequest`錯誤。如需有關使用`NoncurrentVersionExpiration`動作元素的詳細資訊，請參閱the section called “描述生命週期動作的元素”。

如需詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。

> **Important**
>
> 當您在 S3 生命週期組態中有多個規則時，一個物件可能會在同一天符合多個 S3 生命週期動作的資格。在這種情況下，Amazon S3 遵循以下一般規則：
>
> - 永久刪除優先於轉換。

- 轉移優先於刪除標記的建立。
- 當物件同時符合 S3 冰川彈性擷取和 S3 標準 — IA (或 S3 單區域 — IA) 轉換的資格時，Amazon S3 會選擇 S3 冰川彈性擷取轉換。

如需範例，請參閱 範例 5：篩選條件重疊、生命週期動作相衝突，以及 Amazon S3 對為進行版本控制的儲存貯體所採取的動作。

現有物件和新物件

當您新增儲存貯體的生命週期組態時，組態規則會套用至現有物件以及稍後新增的物件。例如，如果您今天使用到期動作新增生命週期組態規則，使具有特定前置詞的物件在建立後 30 天到期，Amazon S3 將會排入佇列移除任何超過 30 天且具有指定前置碼的現有物件。

Important

您無法使用儲存貯體政策來防止 S3 生命週期規則刪除或轉換。例如，即使儲存貯體政策拒絕所有主體的所有動作，S3 生命週期組態仍可正常運作。

如何找出物件何時過期

若要尋找物件排程到期的時間，請使用HeadObject或 GetObjectAPI 作業。這些 API 操作會傳回回應標頭，提供不再可快取物件的日期和時間。

Note

- 在過期日期及 Amazon S3 移除物件的日期之間，可能會有所延遲。您無須支付與過期物件相關聯的過期或儲存時間費用。
- 更新、停用或刪除生命週期規則之前，請使用 LIST API 操作 (例如ListObjectsV2ListObjectVersions、和 ListMultipartUploads)，或Amazon S3 清查根據您的使用案例確認 Amazon S3 已轉換並過期合格物件。

最低儲存期間費用

如果您建立的 S3 生命週期到期規則導致 S3 標準 – IA 或 S3 單區域 – IA 儲存中的物件在 30 天內過期，您仍要支付 30 天的費用。如果您建立的生命週期到期規則導致 S3 Glacier Flexible Retrieval 儲存中的物件在 90 天內過期，則系統仍會向您收取 90 天的儲存體費用。如果您建立的生命週期到期規則導致 S3 Glacier Deep Archive 儲存中的物件在 180 天內過期，則系統仍會向您收取 180 天的儲存體費用。

如需詳細資訊，請參閱 Simple Storage Service (Amazon S3) 定價。

在值區上設定生命週期組態

本節說明如何使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 Amazon S3 REST API 在儲存貯體上設定 Amazon S3 生命週期組態。如需 S3 生命週期組態的資訊，請參閱管理儲存生命週期。

您可以使用生命週期規則，定義要讓 Amazon S3 在物件生命週期內採取的動作 (例如，將物件轉移至另一個儲存類別、封存物件，或在指定期限後刪除物件)。

在設定生命週期組態之前，請注意下列事項：

生命週期組態傳播延

當您新增儲存貯體的 S3 生命週期組態時，在新的或已更新的 S3 生命週期組態完全傳播至所有 Amazon S3 系統之前通常會一些延遲。在組態完全生效之前，預期會有幾分鐘的延遲。當您刪除 S3 生命週期組態時，可能會發生這項延遲。

轉換或到期延遲

滿足生命週期規則到規則的動作完成之間會有延遲。例如，假設一組物件在 1 月 1 日被生命週期規則到期。即使已在 1 月 1 日滿足到期規則，Amazon S3 可能要等到幾天甚至幾週後才真正刪除這些物件。發生此延遲的原因是 S3 生命週期會以非同步方式將物件排入佇列以進行轉換 不過，即使動作未完成，通常會在生命週期規則滿足時套用計費變更。如需詳細資訊，請參閱帳單變更。若要監視使用中生命週期規則所做更新的影響，請參閱 the section called “如何監視生命週期規則所採取的動作？”

停用或刪除生命週期規則

停用或刪除生命週期規則時，Amazon S3 會在短暫延遲後停止排程要刪除或轉換的新物件。任何已排定的物件都會取消排定，而且不會刪除或轉換。

> **Note**
>
> 更新、停用或刪除生命週期規則之前，請使用 LIST API 操作 (例如ListObjectsV2ListObjectVersions、和 ListMultipartUploads)，或Amazon S3 清查根據您的使用案例確認 Amazon S3 已轉換並過期合格物件。如果您在更新、停用或刪除生命週期規則時遇到任何問題，請參閱針對 Amazon S3 生命週期問題進行疑難排解。

現有物件和新物件

當您新增儲存貯體的生命週期組態時，組態規則會套用至現有物件以及稍後新增的物件。例如，如果您今天使用到期動作新增生命週期組態規則，使具有特定前置詞的物件在建立後 30 天到期，Amazon S3 將會排入佇列移除任何超過 30 天且具有指定前置碼的現有物件。

監視生命週期規則的效果

若要監視使用中生命週期規則所做更新的影響，請參閱 the section called “如何監視生命週期規則所採取的動作？”

帳單變更

在滿足生命週期組態規則與執行符合規則所觸發的動作之間，可能會有延遲。不過，只要滿足生命週期組態規則，就會立即變更計費，即使尚未採取動作也一樣。

例如，在物件到期時間之後，即使物件未立即刪除，也不會向您收取儲存費用。同樣地，一旦物件轉換時間過後，即使物件未立即轉換為 S3 Glacier 彈性擷取儲存類別，您仍需支付 S3 Glacier 彈性擷取儲存費率的費用。

不過，生命週期轉換至 S3 智慧型分層儲存類別是例外。在物件轉換為 S3 智慧型分層儲存類別之後，帳單變更才會發生。

多個或衝突的規則

當您在 S3 生命週期組態中有多個規則時，一個物件可能會在同一天符合多個 S3 生命週期動作的資格。在這種情況下，Amazon S3 遵循以下一般規則：

- 永久刪除優先於轉換。
- 轉移優先於刪除標記的建立。
- 當物件同時符合 S3 冰川彈性擷取和 S3 標準 — IA (或 S3 單區域 — IA) 轉換的資格時，Amazon S3 會選擇 S3 冰川彈性擷取轉換。

如需範例，請參閱 範例 5：篩選條件重疊、生命週期動作相衝突，以及 Amazon S3 對為進行版本控制的儲存貯體所採取的動作。

使用 S3 主控台

您可以使用共用前置詞 (以通用字串開頭的物件名稱) 或標籤，為值區中的所有物件或物件子集定義生命週期規則。在您的生命週期規則中，您可以定義目前與非目前物件版本特定的動作。如需詳細資訊，請參閱下列內容：

- 管理儲存生命週期
- 在 S3 儲存貯體中使用版本控制

建立生命週期規則

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇要建立生命週期規則的儲存貯體名稱。
3. 選擇 Management (管理) 標籤，然後選擇 Create lifecycle rule (建立生命週期規則)。
4. 在 Lifecycle rule name (生命週期規則名稱) 中，輸入規則的名稱。

   在儲存貯體內，名稱必須是唯一的。
5. 選擇生命週期規則的範圍：

   - 若要將此生命週期規則套用至帶有特定前綴或標籤的所有物件，請選擇 Limit the scope to specific prefixes or tags (將範圍限制為特定前綴或標籤)。
     - 若要依字首限制範圍，請在 Prefix (字首) 中輸入字首。
     - 若要依標籤限制範圍，請選擇 Add tag (新增標籤)，然後輸入標籤鍵和值。

     如需物件名稱字首的詳細資訊，請參閱「建立物件索引鍵名稱」。如需物件標籤的詳細資訊，請參閱 使用標籤分類儲存空間。
   - 若要將此生命週期規則套用至值區中的所有物件，請選擇 [此規則適用於值區中的所有物件]，然後選擇 [我確認此規則適用於值區中的所有物件]。
6. 若要依物件大小篩選規則，您可以選取「指定最小物件大小」、「指定物件大小上限」或兩個選項。

   - 當您指定物件大小下限或物件大小上限的值時，該值必須大於 0 個位元組且最多 5 TB。您可以指定該值，以位元組、KB、MB 或 GB 為單位。

- 當您指定這兩個值時，最大物件大小必須大於最小物件大小。

> **Note**
>
> 「最小物件大小」和「最大物件大小」篩選器會排除指定的值。例如，如果您將篩選器設定為使物件大小下限為 128 KB 的物件過期，則只有 128 KB 的物件不會過期。而是，此規則只會套用至大於 128 KB 的物件。

7. 在 Lifecycle rule actions (生命週期規則動作) 下，選擇您希望生命週期規則執行的動作：

   - 在儲存類別之間轉移物件的目前版本
   - 在儲存類別之間轉移物件的先前版本
   - 讓物件的目前版本過期

   > **Note**
   >
   > 對於未啟用 S3 版本控制的儲存貯體，目前版本過期會導致 Amazon S3 永久刪除物件。如需詳細資訊，請參閱 the section called “生命週期動作與儲存貯體版本控制的狀態”。

   - 永久刪除物件的先前版本
   - 刪除過期的刪除標記或未完成的分段上傳

   視您選擇的動作而定，將會出現不同的選項。

8. 若要在儲存類別之間轉移物件的目前版本，請在 Transition current versions of objects between storage classes (在儲存類別之間轉移物件的目前版本) 下：

   a. 在儲存類別轉換中，選擇要轉換至的儲存空間類別。如需可能轉變的清單，請參閱the section called “支援的生命週期轉換”。您可以從下列儲存類別中進行選擇：

      - S3 標準 – IA
      - S3 Intelligent-Tiering
      - S3 單區域 – IA
      - S3 Glacier Flexible Retrieval
      - S3 Glacier Deep Archive

   b. 在 Days after object creation (物件建立後的天數) 中，輸入建立後幾天要轉移物件。

如需儲存體方案的詳細資訊，請參閱「使用 Amazon S3 儲存體方案」。您可以為目前的物件版本或舊的物件版本定義轉換，也可以同時為兩者定義轉換。您可利用版本控制在單一儲存貯體中保留物件的多個版本。如需版本控制的詳細資訊，請參閱「使用 S3 主控台」。

> **Important**
>
> 選擇 S3 Glacier Flexible Retrieval 或 Glacier Deep Archive 儲存類別時，您的物件會保留在 Amazon S3 中。您無法透過個別的 Amazon S3 Glacier 服務直接存取物件。如需詳細資訊，請參閱 使用 Amazon S3 生命週期轉換物件。

9. 若要在儲存類別之間轉換物件的非目前版本，請在儲存類別之間的轉換物件的非目前版本下：

   a. 在儲存類別轉換中，選擇要轉換至的儲存空間類別。如需可能轉變的清單，請參閱the section called “支援的生命週期轉換”。您可以從下列儲存類別中進行選擇：

      - S3 標準 – IA
      - S3 Intelligent-Tiering
      - S3 單區域 – IA
      - S3 Glacier Flexible Retrieval
      - S3 Glacier Deep Archive

   b. 在物件變為非目前物件後的天數中，輸入建立後轉移物件的天數。

10. 若要讓物件的目前版本過期，請在 Expire current versions of objects (讓物件的目前版本過期) 下的 Number of days after object creation (物件建立後的天數) 中，輸入天數。

> **Important**
>
> 在未版本控制的儲存貯體中，到期動作會導致 Amazon S3 永久移除該物件。如需生命週期動作的詳細資訊，請參閱「描述生命週期動作的元素」。

11. 若要永久刪除物件的先前版本，請在 Permanently delete noncurrent versions of objects (永久刪除物件的非現行版本) 下的 Days after objects become noncurrent (物件變成非現行版本後的天數) 中，輸入天數。您可以選擇指定要保留的較新版本數目，方法是在要保留的較新版本數目下方輸入值。

12. 在 Delete expired delete markers or incomplete multipart uploads (刪除過期刪除標記或未完成的分段上傳) 下，選擇 Delete expired object delete markers (刪除過期物件刪除標記) 和 Delete

incomplete multipart uploads (刪除未完成的分段上傳)。然後，輸入分段上傳啟動後幾天要結束並清理未完成的分段上傳。

如需分段上傳的詳細資訊，請參閱「使用分段上傳來上傳和複製物件」。

13. 選擇建立規則。

如果規則未包含任何錯誤，Amazon S3 會啟用該規則，您可以在 Lifecycle rules (生命週期規則) 下的 Management (管理) 標籤上看到該規則。

若要取得有關 AWS CloudFormation 範本和範例的資訊，請參閱〈使用範 AWS CloudFormation 本〉和AWS::S3::Bucket〈AWS CloudFormation 使用指南〉。

使用 AWS CLI

您可以使用下列命 AWS CLI 令來管理 S3 生命週期組態：

- `put-bucket-lifecycle-configuration`
- `get-bucket-lifecycle-configuration`
- `delete-bucket-lifecycle`

如需有關設定的指示 AWS CLI，請參閱使用 AWS CLI 來透過 Amazon S3 進行開發。

Amazon S3 生命週期組態是 XML 檔案。但是當您使用時 AWS CLI，您無法指定 XML 格式。您必須改為指定 JSON 格式。以下是 XML 生命週期組態範例，以及您可以在 AWS CLI命令中指定的對等 JSON 組態。

請考慮下列範例 S3 生命週期組態。

Example 範例 1

Example

XML

```
<LifecycleConfiguration>
    <Rule>
        <ID>ExampleRule</ID>
        <Filter>
           <Prefix>documents/</Prefix>
        </Filter>
        <Status>Enabled</Status>
```

```
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
             <Days>3650</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

JSON

```
{
    "Rules": [
        {
            "Filter": {
                "Prefix": "documents/"
            },
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 365,
                    "StorageClass": "GLACIER"
                }
            ],
            "Expiration": {
                "Days": 3650
            },
            "ID": "ExampleRule"
        }
    ]
        }
```

Example 範例 2

Example

XML

```
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Rule>
```

```
        <ID>id-1</ID>
        <Expiration>
            <Days>1</Days>
        </Expiration>
        <Filter>
            <And>
                <Prefix>myprefix</Prefix>
                <Tag>
                    <Key>mytagkey1</Key>
                    <Value>mytagvalue1</Value>
                </Tag>
                <Tag>
                    <Key>mytagkey2</Key>
                    <Value>mytagvalue2</Value>
                </Tag>
            </And>
        </Filter>
        <Status>Enabled</Status>
    </Rule>
</LifecycleConfiguration>
```

JSON

```
{
    "Rules": [
        {
            "ID": "id-1",
            "Filter": {
                "And": {
                    "Prefix": "myprefix",
                    "Tags": [
                        {
                            "Value": "mytagvalue1",
                            "Key": "mytagkey1"
                        },
                        {
                            "Value": "mytagvalue2",
                            "Key": "mytagkey2"
                        }
                    ]
                }
            },
```

```
            "Status": "Enabled",
            "Expiration": {
                "Days": 1
            }
        }
    ]
}
```

您可以如下來測試 put-bucket-lifecycle-configuration。

測試組態

1. 將 JSON 生命週期組態儲存在檔案中 (例如，*lifecycle.json*)。
2. 執行下列 AWS CLI 命令，在值區上設定生命週期組態。以您自己的資訊取代 *user input placeholders*。

```
$ aws s3api put-bucket-lifecycle-configuration  \
--bucket DOC-EXAMPLE-BUCKET  \
--lifecycle-configuration file://lifecycle.json
```

3. 若要驗證，請使用命令擷取 S3 生get-bucket-lifecycle-configuration AWS CLI 命週期組態，如下所示：

```
$ aws s3api get-bucket-lifecycle-configuration  \
--bucket DOC-EXAMPLE-BUCKET
```

4. 若要刪除 S3 生命週期組態，請依照下列方式使用delete-bucket-lifecycle AWS CLI 指令：

```
aws s3api delete-bucket-lifecycle \
--bucket DOC-EXAMPLE-BUCKET
```

使用 AWS 軟體開發套件

Java

您可以使用 AWS SDK for Java 來管理儲存貯體的 S3 生命週期組態。如需管理 S3 生命週期組態的詳細資訊，請參閱管理儲存生命週期。

> **Note**
>
> 當您新增儲存貯體的 S3 生命週期組態時，Amazon S3 會取代儲存貯體的 S3 生命週期組態，如果有的話。若要更新現有生命週期組態，請先擷取，再進行所要的變更，然後將修正過的組態新增至儲存貯體。

下列範例顯示如何使用 AWS SDK for Java 新增、更新和刪除值區的生命週期組態。此範例執行下列操作：

- 新增儲存貯體的生命週期組態。
- 擷取生命週期組態，並新增其他規則來更新此組態。
- 將經過修改的生命週期組態組態，加入到儲存貯體中。Amazon S3 取代現有的組態。
- 再次擷取組態，並透過列印規則數目驗證其具有正確數目的規則。
- 刪除生命週期組態，並嘗試再次擷取，以驗證它是否已刪除。

如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketLifecycleConfiguration;
import com.amazonaws.services.s3.model.BucketLifecycleConfiguration.Transition;
import com.amazonaws.services.s3.model.StorageClass;
import com.amazonaws.services.s3.model.Tag;
import com.amazonaws.services.s3.model.lifecycle.LifecycleAndOperator;
import com.amazonaws.services.s3.model.lifecycle.LifecycleFilter;
import com.amazonaws.services.s3.model.lifecycle.LifecyclePrefixPredicate;
import com.amazonaws.services.s3.model.lifecycle.LifecycleTagPredicate;

import java.io.IOException;
import java.util.Arrays;

public class LifecycleConfiguration {

        public static void main(String[] args) throws IOException {
```

```
                Regions clientRegion = Regions.DEFAULT_REGION;
                String bucketName = "*** Bucket name ***";

                // Create a rule to archive objects with the "glacierobjects/"
prefix to Glacier
                // immediately.
                BucketLifecycleConfiguration.Rule rule1 = new
BucketLifecycleConfiguration.Rule()
                                .withId("Archive immediately rule")
                                .withFilter(new LifecycleFilter(new
LifecyclePrefixPredicate("glacierobjects/")))
                                .addTransition(new
Transition().withDays(0).withStorageClass(StorageClass.Glacier))
                                .withStatus(BucketLifecycleConfiguration.ENABLED);

                // Create a rule to transition objects to the Standard-Infrequent
Access storage
                // class
                // after 30 days, then to Glacier after 365 days. Amazon S3 will
delete the
                // objects after 3650 days.
                // The rule applies to all objects with the tag "archive" set to
"true".
                BucketLifecycleConfiguration.Rule rule2 = new
BucketLifecycleConfiguration.Rule()
                                .withId("Archive and then delete rule")
                                .withFilter(new LifecycleFilter(new
LifecycleTagPredicate(new Tag("archive", "true"))))
                                .addTransition(new Transition().withDays(30)

.withStorageClass(StorageClass.StandardInfrequentAccess))
                                .addTransition(new
Transition().withDays(365).withStorageClass(StorageClass.Glacier))
                                .withExpirationInDays(3650)
                                .withStatus(BucketLifecycleConfiguration.ENABLED);

                // Add the rules to a new BucketLifecycleConfiguration.
                BucketLifecycleConfiguration configuration = new
BucketLifecycleConfiguration()
                                .withRules(Arrays.asList(rule1, rule2));

                try {
                        AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
```

```
                                        .withCredentials(new
ProfileCredentialsProvider())
                                        .withRegion(clientRegion)
                                        .build();

                        // Save the configuration.
                        s3Client.setBucketLifecycleConfiguration(bucketName,
configuration);

                        // Retrieve the configuration.
                        configuration =
s3Client.getBucketLifecycleConfiguration(bucketName);

                        // Add a new rule with both a prefix predicate and a tag
predicate.
                        configuration.getRules().add(new
BucketLifecycleConfiguration.Rule().withId("NewRule")
                                        .withFilter(new LifecycleFilter(new
LifecycleAndOperator(
                                                        Arrays.asList(new
LifecyclePrefixPredicate("YearlyDocuments/"),
                                                                        new
LifecycleTagPredicate(new Tag(

   "expire_after",

   "ten_years"))))))
                                        .withExpirationInDays(3650)

.withStatus(BucketLifecycleConfiguration.ENABLED));

                        // Save the configuration.
                        s3Client.setBucketLifecycleConfiguration(bucketName,
configuration);

                        // Retrieve the configuration.
                        configuration =
s3Client.getBucketLifecycleConfiguration(bucketName);

                        // Verify that the configuration now has three rules.
                        configuration =
s3Client.getBucketLifecycleConfiguration(bucketName);
                        System.out.println("Expected # of rules = 3; found: " +
configuration.getRules().size());
```

```
                        // Delete the configuration.
                        s3Client.deleteBucketLifecycleConfiguration(bucketName);

                        // Verify that the configuration has been deleted by
 attempting to retrieve it.
                        configuration =
 s3Client.getBucketLifecycleConfiguration(bucketName);
                        String s = (configuration == null) ? "No configuration
 found." : "Configuration found.";
                        System.out.println(s);
                } catch (AmazonServiceException e) {
                        // The call was transmitted successfully, but Amazon S3
 couldn't process
                        // it, so it returned an error response.
                        e.printStackTrace();
                } catch (SdkClientException e) {
                        // Amazon S3 couldn't be contacted for a response, or the
 client
                        // couldn't parse the response from Amazon S3.
                        e.printStackTrace();
                }
        }
}
```

.NET

您可以使用 AWS SDK for .NET 來管理儲存貯體上的 S3 生命週期組態。如需管理生命週期組態的詳細資訊，請參閱「管理儲存生命週期」。

> **Note**
>
> 當您新增生命週期組態時，Amazon S3 會取代在指定儲存貯體的任何現有組態。若要更新生命週期組態，您必須先擷取現有的生命週期組態，並進行變更，然後將修正過的生命週期組態新增至儲存貯體。

下列範例顯示如何使用新 AWS SDK for .NET 增、更新和刪除值區的生命週期組態。此程式法範例可做到以下：

- 新增儲存貯體的生命週期組態。

- 擷取生命週期組態，並新增其他規則來更新此組態。
- 將經過修改的生命週期組態組態，加入到儲存貯體中。Amazon S3 取代現有的生命週期組態。
- 再次擷取組態，並透過列印規則數，驗證其具有正確數量的規則。
- 刪除生命週期組態並驗證刪除。

如需有關設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class LifecycleTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddUpdateDeleteLifecycleConfigAsync().Wait();
        }

        private static async Task AddUpdateDeleteLifecycleConfigAsync()
        {
            try
            {
                var lifeCycleConfiguration = new LifecycleConfiguration()
                {
                    Rules = new List<LifecycleRule>
                        {
                            new LifecycleRule
                            {
                                 Id = "Archive immediately rule",
```

```
                                Filter = new LifecycleFilter()
                                {
                                    LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                    {
                                        Prefix = "glacierobjects/"
                                    }
                                },
                                Status = LifecycleRuleStatus.Enabled,
                                Transitions = new List<LifecycleTransition>
                                {
                                     new LifecycleTransition
                                     {
                                          Days = 0,
                                          StorageClass = S3StorageClass.Glacier
                                     }
                                 },
                            },
                            new LifecycleRule
                            {
                                 Id = "Archive and then delete rule",
                                  Filter = new LifecycleFilter()
                                 {
                                     LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                     {
                                         Prefix = "projectdocs/"
                                     }
                                 },
                                 Status = LifecycleRuleStatus.Enabled,
                                 Transitions = new List<LifecycleTransition>
                                 {
                                      new LifecycleTransition
                                      {
                                           Days = 30,
                                           StorageClass =
S3StorageClass.StandardInfrequentAccess
                                      },
                                      new LifecycleTransition
                                      {
                                        Days = 365,
                                        StorageClass = S3StorageClass.Glacier
                                      }
                                 },
```

```
                                Expiration = new LifecycleRuleExpiration()
                                {
                                      Days = 3650
                                }
                            }
                        }
                };

                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client,
lifeCycleConfiguration);

                // Retrieve an existing configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

                // Add a new rule.
                lifeCycleConfiguration.Rules.Add(new LifecycleRule
                {
                    Id = "NewRule",
                    Filter = new LifecycleFilter()
                    {
                        LifecycleFilterPredicate = new LifecyclePrefixPredicate()
                        {
                            Prefix = "YearlyDocuments/"
                        }
                    },
                    Expiration = new LifecycleRuleExpiration()
                    {
                        Days = 3650
                    }
                });

                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client,
lifeCycleConfiguration);

                // Verify that there are now three rules.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);
                Console.WriteLine("Expected # of rulest=3; found:{0}",
lifeCycleConfiguration.Rules.Count);

                // Delete the configuration.
                await RemoveLifecycleConfigAsync(client);
```

```
                // Retrieve a nonexistent configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
LifecycleConfiguration configuration)
        {

            PutLifecycleConfigurationRequest request = new
PutLifecycleConfigurationRequest
            {
                BucketName = bucketName,
                Configuration = configuration
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }

        static async Task<LifecycleConfiguration>
RetrieveLifecycleConfigAsync(IAmazonS3 client)
        {
            GetLifecycleConfigurationRequest request = new
GetLifecycleConfigurationRequest
            {
                BucketName = bucketName
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }

        static async Task RemoveLifecycleConfigAsync(IAmazonS3 client)
        {
```

```
            DeleteLifecycleConfigurationRequest request = new
 DeleteLifecycleConfigurationRequest
            {
                BucketName = bucketName
            };
            await client.DeleteLifecycleConfigurationAsync(request);
        }
    }
}
```

Ruby

您可以使用類別組態 AWS SDK for Ruby 來管理儲存貯體上的 S3 生命週期 AWS::S3::BucketLifecycle組態。如需管理生命週期組態的詳細資訊，請參閱「管理儲存生命週期」。

使用 REST API

Amazon Simple Storage Service API 參考 中的下列章節說明與 S3 生命週期組態相關的 REST API。

- PutBucketLifecycleConfiguration
- GetBucketLifecycleConfiguration
- DeleteBucketLifecycle

進行 S3 生命週期疑難排解

如需使用 S3 生命週期時可能發生的常見問題，請參閱the section called “針對生命週期問題進行疑難排解”。

生命週期及其他儲存貯體組態

除了 S3 生命週期組態之外，還可以將其他組態與儲存貯體產生關聯。本節說明 S3 生命週期組態與其他儲存貯體組態的相關性。

生命週期與版本控制

您可以對未版本控制及已啟用版本控制的儲存貯體，新增 S3 生命週期組態。如需詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。

已啟用版本控制的儲存貯體會維持一個目前的物件版本，以及零或多個非目前的物件版本。您可以為目前及非目前的物件版本定義另外的生命週期規則。

如需詳細資訊，請參閱 生命週期組態元素。

> **Important**
>
> 當您在 S3 生命週期組態中有多個規則時，一個物件可能會在同一天符合多個 S3 生命週期動作的資格。在這種情況下，Amazon S3 遵循以下一般規則：
>
> - 永久刪除優先於轉換。
> - 轉移優先於刪除標記的建立。
> - 當物件同時符合 S3 Glacier Flexible Retrieval 和 S3 標準 – IA (或 S3 單區域 – IA) 轉換的資格時，Amazon S3 會選擇 S3 Glacier Flexible Retrieval 轉換。
>
> 如需範例，請參閱 範例 5：篩選條件重疊、生命週期動作相衝突，以及 Amazon S3 對為進行版本控制的儲存貯體所採取的動作。

啟動 MFA 儲存貯體的生命週期組態

已啟用 Multi-Factor Authentication (MFA) 之儲存貯體上不支援生命週期組態。

生命週期與記錄

AWS CloudTrail 物件層級記錄不會擷取 Amazon S3 生命週期動作。 CloudTrail擷取對外部 Amazon S3 端點發出的 API 請求，而 S3 生命週期動作則是使用內部 Amazon S3 端點執行。您可以在 S3 儲存貯體中啟用 Amazon S3 伺服器存取日誌，以擷取 S3 生命週期相關的動作，例如物件轉換到另一個儲存類別和物件到期，從而導致永久刪除或邏輯刪除。如需詳細資訊，請參閱「the section called “記錄伺服器存取”」。

如果您在儲存貯體上啟用日誌功能，Amazon S3 伺服器存取日誌就會回報以下操作的結果。

操作日誌	描述
`S3.EXPIRE.OBJECT`	Amazon S3 會因為生命週期到期動作而永久刪除該物件。

操作日誌	描述
S3.CREATE.DELETEMARKER	Amazon S3 會以邏輯方式刪除目前版本，並在啟用版本控制的儲存貯體中新增刪除標記。
S3.TRANSITION_SIA.OBJECT	Amazon S3 將物件轉換至 S3 標準 – IA 儲存類別。
S3.TRANSITION_ZIA.OBJECT	Amazon S3 會將物件轉換為 S3 單區域 – IA 儲存類別。
S3.TRANSITION_INT.OBJECT	Amazon S3 將物件轉換至 S3 Intelligent-Tiering 儲存類別。
S3.TRANSITION_GIR.OBJECT	Amazon S3 會啟動物件轉換至 S3 冰川即時擷取儲存類別。
S3.TRANSITION.OBJECT	Amazon S3 會啟動物件轉換至 S3 冰川彈性擷取儲存類別。
S3.TRANSITION_GDA.OBJECT	Amazon S3 會啟動物件轉換至 S3 冰川深層存檔儲存類別。
S3.DELETE.UPLOAD	Amazon S3 中止不完整的多部分上傳。

> **Note**
>
> Amazon S3 伺服器存取日誌記錄通常會依最佳作法交付，而無法完整日誌所有 Amazon S3 請求。

進行 S3 生命週期疑難排解

如需對 S3 生命週期的常見問題進行疑難排解的詳細資訊，請參閱 針對 Amazon S3 生命週期問題進行疑難排解。

詳細資訊

- 生命週期組態元素

- 轉換為 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 儲存類別 (物件封存)
- 在值區上設定生命週期組態

設定生命週期事件通知

您可以設定 Amazon S3 事件通知，以便在 Amazon S3 按照 S3 生命週期規則刪除物件或將物件轉換到另一個 Amazon S3 儲存類別時收到通知。

透過使用`LifecycleExpiration`事件類型，您可以在 Amazon S3 根據 S3 生命週期組態刪除物件時收到通知。`s3:LifecycleExpiration:Delete` 事件類型會在刪除未進行版本控制之儲存貯體中的物件時通知您。當 S3 生命週期組態永久刪除物件版本時，它也會通知您。當刪除版本化儲存貯體中物件的目前版本時，S3 生命週期建立刪除標記時，`s3:LifecycleExpiration:DeleteMarkerCreated`事件類型會通知您。如需詳細資訊，請參閱刪除物件版本。

透過使用`s3:LifecycleTransition`事件類型，您可以在 S3 生命週期組態將物件從一個 Amazon S3 儲存類別轉換到另一個儲存類別時收到通知。

Amazon S3 可以將事件通知發佈到 Amazon Simple Notification Service (Amazon SNS) 主題、Amazon Simple Queue Service (Amazon SQS) 佇列或 AWS Lambda 函數。如需詳細資訊，請參閱 Amazon S3 事件通知。

如需如何設定 Amazon S3 事件通知的指示，請參閱啟用事件通知。

以下是 Amazon S3 傳送以發佈 `s3:LifecycleExpiration:Delete` 事件的訊息範例。如需詳細資訊，請參閱事件訊息結構。

```
{
   "Records":[
      {
         "eventVersion":"2.3",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"LifecycleExpiration:Delete",
         "userIdentity":{
            "principalId":"s3.amazonaws.com"
         },
         "requestParameters":{
            "sourceIPAddress":"s3.amazonaws.com"
```

```
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"example-s3-bucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::example-s3-bucket"
            },
            "object":{
               "key":"expiration/delete",
               "sequencer":"0055AED6DCD90281E5",
            }
         }
      }
   ]
}
```

Amazon S3 傳送以發佈`s3:LifecycleTransition`事件的訊息也包含下列資訊。

```
"lifecycleEventData":{
    "transitionEventData": {
        "destinationStorageClass": the destination storage class for the object
    }
}
```

生命週期組態元素

主題

- ID 元素
- Status 元素
- Filter 元素
- 描述生命週期動作的元素

您可以將 Amazon S3 生命週期組態指定為 XML，包含一或多個生命週期規則。

```
<LifecycleConfiguration>
    <Rule>
         ...
    </Rule>
    <Rule>
         ...
    </Rule>
</LifecycleConfiguration>
```

每項規則皆包含下列各項：

- 包含規則識別碼的規則中繼資料，以及指出規則是啟用或停用的狀態。當規則停用時，Amazon S3 不會執行規則中指定的任何動作。
- 可識別套用規則之物件的篩選器。您可以使用物件大小、物件 key prefix、一或多個物件標籤或篩選組合來指定濾鏡。
- 當您想要 Amazon S3 執行指定的動作時，物件的生命週期中必須包含具有日期或時段的一或多個轉換或過期動作。

下列各節描述 S3 生命週期組態中的 XML 元素。如需組態範例，請參閱 S3 生命週期組態範例。

ID 元素

S3 生命週期組態最多可有 1,000 項規則。此限制不可調整。元`<ID>`素可唯一識別規則。ID 長度不得超過 255 個字元。

Status 元素

元`<Status>`素值可以是`Enabled`或`Disabled`。當規則停用時，Amazon S3 不會執行規則中定義的任何動作。

Filter 元素

根據您在生命週期規則中指定的`<Filter>`元素，生命週期規則可以套用至值區中的所有物件或子集。

您可以依金鑰字首 (key prefix)、物件標籤或兩者的組合來篩選物件 (在此情況下，Amazon S3 會使用邏輯 AND 來合併篩選條件)。請考量下列範例：

- 使用金鑰前置詞指定篩選器 — 此範例顯示 S3 生命週期規則，該規則會根據金鑰名稱前置詞 (logs/) 套用至物件子集。例如，生命週期規則適用於logs/mylog.txtlogs/temp1.txt、和的物件logs/test.txt。此規則不會套用到物件 example.jpg。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Prefix>logs/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>
    ...
</LifecycleConfiguration>
```

如果您要根據不同的索引鍵名稱首碼，將生命週期動作套用至物件子集，請指定個別的規則。請在每項規則中，指定使用字首的篩選條件。例如，若要描述具有索引鍵前置詞projectA/之物件的生命週期動作projectB/，您可以指定兩個規則，如下所示：

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Prefix>projectA/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>

    <Rule>
        <Filter>
           <Prefix>projectB/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>
</LifecycleConfiguration>
```

如需物件金鑰的詳細資訊，請參閱「建立物件索引鍵名稱」。

- 根據物件標籤指定篩選器 — 在下列範例中，生命週期規則會根據標籤 (*key*) 和值 (*value*) 指定篩選器。此規則只會套用到其中具有特定標籤的物件。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Tag>
              <Key>key</Key>
              <Value>value</Value>
           </Tag>
        </Filter>
        transition/expiration actions
        ...
    </Rule>
</LifecycleConfiguration>
```

您可以依據多個標籤指定篩選條件。您必須將標籤包裝在<And>元素中，如下列範例所示。此規則會指示 Amazon S3 對具有這兩個標籤 (具有特定標籤金鑰與值) 的物件執行生命週期動作。

```
<LifecycleConfiguration>
    <Rule>
      <Filter>
         <And>
            <Tag>
               <Key>key1</Key>
               <Value>value1</Value>
            </Tag>
            <Tag>
               <Key>key2</Key>
               <Value>value2</Value>
            </Tag>
             ...
         </And>
      </Filter>
      transition/expiration actions
    </Rule>
</Lifecycle>
```

此生命週期規則會套用到同時具有這兩個指定標籤的物件。Amazon S3 會執行邏輯 AND。注意下列事項：

- 每個標籤都必須完全匹配鍵和值。如果您只指定<Key>元素而沒有<Value>元素，則規則只會套用至符合標籤鍵且沒有指定值的物件。

- 此規則會套用至擁有規則中所有指定標籤的物件子集。如果物件指定了其他標籤，則仍會套用規則。

> **Note**
>
> 當您在篩選條件中指定多個標籤時，每個標籤金鑰只能指定一次。

- 根據字首和一個或多個標籤指定篩選 — 在生命週期規則中，您可以根據 key prefix 和一或多個標籤來指定篩選。同樣地，您必須將所有這些篩選元素包裝在<And>元素中，如下所示：

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
          <And>
             <Prefix>key-prefix</Prefix>
             <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
             </Tag>
             <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
             </Tag>
              ...
          </And>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

  Amazon S3 通過使用邏輯結合這些過濾器AND。也就是說，該規則適用於具有指定 key prefix 和指定標籤的物件子集。篩選條件只能有一個字首，以及零或多個標籤。

- 您可以指定 empty filter (空的篩選條件)，讓規則套用到儲存貯體中的所有物件。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
```

```
    </Rule>
</LifecycleConfiguration>
```

- 若要依據 object size (物件大小) 篩選規則，您可以指定最小大小 (ObjectSizeGreaterThan) 或最大大小 (ObjectSizeLessThan)，或者您可以指定物件大小的範圍。

  物件大小值以位元組為單位。過濾器大小上限為 5 TB。某些儲存類別具有最小物件大小限制。如需詳細資訊，請參閱 比較 Amazon S3 儲存方案。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

> **Note**
>
> ObjectSizeGreaterThan和ObjectSizeLessThan篩選器會排除指定的值。例如，如果您將 128 KB 的物件設定為 1024 KB，以便從 S3 標準儲存類別移至 S3 標準 — IA 儲存類別，則只有 1024 KB 和 128 KB 的物件將不會轉換到 S3 標準 — IA。相反地，此規則只會套用至大於 128 KB 且大小小於 1024 KB 的物件。

如果您指定的是物件大小範圍，則 ObjectSizeGreaterThan 整數必須小於 ObjectSizeLessThan 值。使用多個篩選條件時，您必須將篩選條件包裝在 <And> 元素中。下列範例會示範如何指定範圍介於 500 位元組和 64,000 個位元組之間的物件。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <And>
                <Prefix>key-prefix</Prefix>
                <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
                <ObjectSizeLessThan>64000</ObjectSizeLessThan>
            </And>
        </Filter>
```

```
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

描述生命週期動作的元素

您可以在 S3 生命週期規則中，指定一或多個下列預先定義的動作，指示 Amazon S3 在物件生命週期中執行特定動作。這些動作的效用取決於儲存貯體的版本控制狀態。

- **`Transition`動作元素** — 您可以指定將物件從一個儲存類別轉移到另一個儲存類別的`Transition`動作。如需轉換物件的詳細資訊，請參閱「支援的轉換及相關限制」。當物件生命週期的指定日期或時段一到，Amazon S3 便會執行轉換。

  對於使用版本控制的儲存貯體 (啟用了或暫停了版本控制的儲存貯體)，`Transition` 動作會套用至目前的物件版本。若要管理非最新的版本，Amazon S3 會定義 `NoncurrentVersionTransition` 動作 (如本主題稍後的內容所述)。

- **`Expiration`動作元素** — `Expiration` 動作會過期規則中識別的物件，並套用至任何 Amazon S3 儲存類別中的合格物件。如需儲存體方案的詳細資訊，請參閱「使用 Amazon S3 儲存體方案」。Amazon S3 會使所有過期的物件無法使用。物件是否要永久移除，取決於儲存貯體的版本控制狀態。
  - 未版本控制的儲存貯體 — 此`Expiration`動作會導致 Amazon S3 永久移除物件。
  - 使用版本控制的儲存貯體 - 對於使用版本控制的儲存貯體 (亦即，啟用了或暫停了版本控制功能的儲存貯體)，Amazon S3 在處理 `Expiration` 動作時，會考量幾項重點。對於已啟用版本控制或暫停版本控制的儲存貯體，適用下列情況：
    - `Expiration` 動作只會套用到目前的版本 (其對非目前的物件版本沒有影響)。
    - 當有一或多個物件版本，且刪除標記為目前的版本時，Amazon S3 不會採取任何動作。
    - 若目前的物件版本是唯一的物件版本，同時也是刪除標記 (亦稱為過期物件刪除標記，這會刪除所有的物件版本，只留下刪除標記)，Amazon S3 會移除過期物件刪除標記。您也可以使用過期動作，指示 Amazon S3 移除任何過期物件刪除標記。如需範例，請參閱 範例 7：移除過期物件刪除標記。

    如需詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。

    設定 Amazon S3 管理過期時也請考量以下事項：

- 啟用版本控制的儲存貯體

  如果目前的物件版本不是刪除標記，則 Amazon S3 會新增具有唯一版本 ID 的刪除標記。如此會讓目前的版本成為非目前的版本，而刪除標記成為目前版本。

- 暫停版本控制的儲存貯體

  在暫停版本控制的儲存貯體中，到期動作會導致 Amazon S3 建立刪除標記作`null`為版本 ID。此刪除標記會以 null 版本 ID 取代版本階層中所有的物件版本。這是刪除物件最有效的方法。

此外，Amazon S3 提供下列動作讓您用於管理使用版本控制之儲存貯體 (即已啟用或已暫停版本控制的儲存貯體) 中非最新的物件版本。

- **`NoncurrentVersionTransition`動作元素** — 使用此動作可指定 Amazon S3 何時將物件轉換到指定的儲存類別。您可以根據物件變成非目前的特定天數來建立此到期日。除了天數之外，您還可以提供要保留的最大封存版本數目 (介於 1 到 100 之間)。此值決定 Amazon S3 才能在指定版本上執行相關動作之前，必須存在多少個較新的非最新版本。Amazon S3 會將超出指定數目的任何其他非最新版本轉換為保留。

  若要指定封存版本的最大數目，您還必須提供`Filter`元素。如果您未指定`Filter`元素，Amazon S3 會在您提供最大數量的非最新版本時產生`InvalidRequest`錯誤。

  如需轉換物件的詳細資訊，請參閱「支援的轉換及相關限制」。如需 Amazon S3 如何計算您在 `NoncurrentVersionTransition` 動作中指定天數之日期的詳細資訊，請參閱 生命週期規則：依據物件的存在時間。

- **`NoncurrentVersionExpiration`動作元素** — 使用此動作可指示 Amazon S3 永久刪除物件的非最新版本。無法復原這些已刪除的物件。您可以根據物件變成非目前的特定天數來建立此到期日。除了天數之外，您還可以提供要保留的最大封存版本數目 (介於 1 到 100 之間)。此值會指定必須有多少個較新的非目前版本存在，Amazon S3 才可對指定的版本執行相關聯的動作。Amazon S3 將永久刪除超出指定數目的任何其他非最新版本以保留。

  若要指定封存版本的最大數目，您還必須提供`Filter`元素。如果您未指定`Filter`元素，Amazon S3 會在您提供最大數量的非最新版本時產生`InvalidRequest`錯誤。

  當您需要更正任何意外的刪除或覆寫時，延後移除非最新物件的方法十分實用。例如，您可以設定過期規則，在物件變成非目前的版本後五天再予刪除。例如，假設您在 2014 年 1 月 1 日上午 10 點 30 分建立一個名為 `photo.gif` (版本識別碼 111111) 的物件。在 2014 年 2 月 1 日上午 11 時 30 分，您不小心刪除了 `photo.gif` (版本識別碼 111111)，這會建立具有新版本識別碼 (例如版本識別碼 4857693) 的刪除標記。現在在永久刪除之前，您有五天的時間可以復原原始版本的 `photo.gif`

(版本 ID 111111)。在 2014 年 8 月 1 日 (世界標準時間 00:00) 上，到期的生命週期規則會在成為非最新版本的五天後執行並永久刪除 photo.gif (版本識別碼 111111)。

如需 Amazon S3 如何計算您在 NoncurrentVersionExpiration 動作中指定天數之日期的詳細資訊，請參閱 生命週期規則：依據物件的存在時間。

> **Note**
>
> 物件到期生命週期設定不會移除不完整的分段上傳。若要移除不完整的分段上傳，您必須使用本節稍後所述的AbortIncompleteMultipartUpload生命週期設定動作。

除了轉換和到期動作之外，您還可以使用下列生命週期組態動作來指示 Amazon S3 停止不完整的分段上傳，或移除過期的物件刪除標記：

- **AbortIncompleteMultipartUpload**action 元素 — 使用此元素可設定允許多部分上傳保持進行中的最長時間 (以天為單位)。如果適用的分段上傳 (由生命週期規則中prefix指定的金鑰名稱決定) 未在預先定義的期間內成功完成，Amazon S3 會停止不完整的分段上傳。如需詳細資訊，請參閱 中止分段上傳。

  > **Note**
  >
  > 您無法在具有使用物件標籤之篩選器的規則中指定此生命週期動作。

- **ExpiredObjectDeleteMarker**動作元素 — 在啟用版本控制的值區中，包含零個非目前版本的刪除標記稱為已過期物件刪除標記。您可以使用此生命週期動作來指示 Amazon S3 移除過期的物件刪除標記。如需範例，請參閱範例 7：移除過期物件刪除標記。

  > **Note**
  >
  > 您無法在具有使用物件標籤之篩選器的規則中指定此生命週期動作。

Amazon S3 如何計算物件變成非最新版本的時間

在啟用版本控制的儲存貯體中，同一個物件可能會有多個版本。永遠只有一個最新版本，以及零或多個非最新版本。當您每次上傳物件時，目前的版本都會保留為非目前的版本，而新增的版本 (即後來項目) 則會變成目前的版本。為判斷物件成為非最新版本的天數，Amazon S3 會查看後來項目的建立時間。Amazon S3 會使用其後來項目建立時間開始起算後的天數，計算物件變成非最新的天數。

在使用 S3 生命週期組態時還原物件的舊版本

如中所述還原舊版本，您可以使用下列兩種方法之一來擷取物件的先前版本：

- 方法 1 — 將物件的非目前版本複製到同一個值區。複製的物件會變成該物件目前的版本，並保留所有的物件版本。
- 方法 2 — 永久刪除物件的目前版本。當您在刪除目前的物件版本時，實際上是將非目前的版本轉換成該物件目前的版本。

當您將 S3 生命週期組態規則與啟用版本控制的儲存貯體搭配使用時，建議您使用方法 1 做為最佳實務。
S3 生命週期在最終一致的模式下運作。在變更傳播到所有 Amazon S3 系統之前，您永久刪除的目前版本可能不會消失。因此，Amazon S3 可能暫時不知道這項刪除項目。) 同時，您設定要讓非最新物件過期的生命週期規則，可能會永久移除非最新物件，其中也包括了您想要還原的非最新物件。因此，按照方法 1 中的建議複製舊版本是更安全的替代方法。

生命週期動作與儲存貯體版本控制的狀態

生命週期規則：依據物件的存在時間

您可以指定從建立 (或修改) 物件開始的天數，Amazon S3 可以採取指定動作的時段。

當您在 S3 生命週期組態的 `Transition` 與 `Expiration` 動作中指定天數時，請注意下列事項：

- 您指定的值是自物件建立之後的天數，動作將會發生。
- Amazon S3 會將規則中指定的天數新增至物件建立時間，並將結果時間四捨五入至 UTC 午夜的次日來計算時間。例如，如果物件是在世界標準時間上午 10:30 時於 2014 年 1 月 15 日建立，而您在轉變規則中指定了 3 天，則物件的轉換日期會計算為世界標準時間 1/19/2014 00:00。

Note

Amazon S3 只會維護每個物件的上次修改日期。例如，Amazon S3 主控台會在物件的「內容」窗格中顯示上次修改日期。當您最初建立新物件時，此日期會反映物件的建立日期。當您取代此物件之後，其日期也會隨之變更。因此，建立日期與上次修改日期同義。

當在生命週期組態的 NoncurrentVersionTransition 與 NoncurrentVersionExpiration 動作中指定天數時，請注意下列事項：

- 您指定的值是 Amazon S3 將在指定的一或多個物件上執行動作的物件版本變為非最新版本 (亦即，物件被覆寫或刪除時) 的天數。
- Amazon S3 會將規則中指定的天數加到建立物件的新後續任務版本的時間，並將結果時間四捨五入至 UTC 午夜的次日來計算時間。例如，在您的值區中，假設您有物件的目前版本是在世界標準時間上午 10:30 時於 2014 年 1 月 1 日建立的。如果取代目前版本的物件的新版本是在世界標準時間上午 10:30 於 2014 年 1 月 15 日建立，而您在轉移規則中指定了 3 天，則物件的轉換日期會計算為世界標準時間 1/19/2014 00:00。

生命週期規則：依據特定日期

在 S3 生命週期規則中指定動作時，可以指定 Amazon S3 採取動作的日期。當達到指定的日期時，Amazon S3 會將該動作套用到所有符合條件的物件 (依據篩選條件)。

如果您指定具有過去日期的 S3 生命週期動作，則所有合格物件都會立即符合該生命週期動作的資格。

> **Important**
>
> 日期型動作不是單次動作。只要規則狀態為 Enabled，Amazon S3 就會繼續套用日期型動作，即使為過去的日期亦是如此。
> 例如，假設您指定以日期為基礎的Expiration動作來刪除所有物件 (假設規則中未指定篩選器)。當指定日期一到，Amazon S3 便會使儲存貯體中所有的物件過期。Amazon S3 也會繼續使您在儲存貯體中建立的任何新物件過期。若要停止生命週期動作，您必須從生命週期規則中移除動作、停用規則，或從生命週期組態中刪除規則。

日期值必須符合 ISO 8601 格式。時間一律是午夜 UTC。

> **Note**
>
> 您無法使用 Amazon S3 主控台建立以日期為基礎的生命週期規則，但可以檢視、停用或刪除此類規則。

S3 生命週期組態範例

本節提供 S3 生命週期組態的範例。每個範例都會說明在各個範例情境下指定 XML 的方式。

主題

範例 1：指定篩選條件

每個 S3 生命週期規則都包含篩選條件，您可用於找出儲存貯體中將套用 S3 生命週期規則的一組物件。下列 S3 生命週期組態說明如何指定篩選條件的範例。

- 在此 S3 生命週期組態規則中，篩選條件指定了一個金鑰字首 (key prefix) (`tax/`)。因此，規則將會套用至其金鑰名稱字首為 `tax/` 的物件，例如 `tax/doc1.txt` 與 `tax/doc2.txt`。

  該項規則指定了兩個動作，指揮 Amazon S3 執行下列操作：

  - 在建立物件的 365 天 (一年) 後將物件轉換為 S3 Glacier Flexible Retrieval 儲存類別。
  - 在建立物件的 3,650 天 (10 年) 後將物件刪除 (`Expiration` 動作)。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Transition and Expiration Rule</ID>
    <Filter>
       <Prefix>tax/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
```

```
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

您可以為每個動作指定日期，而不是以建立後的天數來指定物件存留時間。但在相同的規則中，不可同時使用 Date 與 Days。

- 若想要將 S3 生命週期規則套用到儲存貯體中的所有物件，請指定空白的字首。在下列組態中，規則會指定 Transition 動作，指示 Amazon S3 在建立物件的 0 天後，將其轉換為 S3 Glacier Flexible Retrieval 儲存類別。此規則表示物件有資格在建立之後 UTC 午夜封存至 S3 Glacier 彈性擷取。如需生命週期限制的詳細資訊，請參閱限制。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Archive all object same-day upon creation</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

- 您可以在篩選條件中指定零或一個金鑰名稱字首，以及零或多個物件標籤。下列程式碼範例會將 S3 生命週期規則套用到金鑰字首 (key prefix) 為 tax/，以及有兩個具有指定金鑰及數值之標籤的一組物件。當指定超過一個篩選條件時，您必須如所示地納入 <And> 元素 (Amazon S3 會套用邏輯 AND 來合併指定的篩選條件)。

```
...
<Filter>
   <And>
      <Prefix>tax/</Prefix>
      <Tag>
         <Key>key1</Key>
         <Value>value1</Value>
      </Tag>
      <Tag>
         <Key>key2</Key>
         <Value>value2</Value>
```

```
      </Tag>
    </And>
</Filter>
...
```

- 您可以僅根據標籤來篩選物件。例如，下列 S3 生命週期規則會套用至具有兩個指定標籤的物件 (其並未指定任何字首)。

```
...
<Filter>
   <And>
      <Tag>
         <Key>key1</Key>
         <Value>value1</Value>
      </Tag>
      <Tag>
         <Key>key2</Key>
         <Value>value2</Value>
      </Tag>
    </And>
</Filter>
...
```

Important

當您在 S3 生命週期組態中有多個規則時，一個物件可能會在同一天符合多個 S3 生命週期動作的資格。在這種情況下，Amazon S3 遵循以下一般規則：

- 永久刪除優先於轉換。
- 轉移優先於刪除標記的建立。
- 當物件符合 S3 冰川彈性擷取和 S3 標準 — IA (或 S3 單區域 — IA) 轉換的資格時，Amazon S3 會選擇 S3 冰川彈性擷取轉換。

如需範例，請參閱「範例 5：篩選條件重疊、生命週期動作相衝突，以及 Amazon S3 對為進行版本控制的儲存貯體所採取的動作」。

範例 2：停用生命週期規則

您可以暫時停用 S3 生命週期規則。下列 S3 生命週期組態指定兩項規則：

- 規則 1 指示 Amazon S3 在建立物件之後不久，即將具備 `logs/` 字首的物件轉換為 S3 Glacier Flexible Retrieval 儲存類別。
- 規則 2 指示 Amazon S3 在建立物件之後不久，即將具備 `documents/` 字首的物件轉換為 S3 Glacier Flexible Retrieval 儲存類別。

在組態中，規則 1 為啟用狀態，規則 2 則為停用狀態。Amazon S3 會忽略已停用的規則。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule2</ID>
    <Filter>
      <Prefix>documents/</Prefix>
    </Filter>
    <Status>Disabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

示例 3：在對象的生命週期內分層存儲類

在此範例中，將使用 S3 生命週期組態，在物件的生命週期內降級其儲存體方案。降級動作有助於降低儲存體費用。如需定價的詳細資訊，請參閱 Amazon S3 定價。

下列 S3 生命週期組態指定了一則會套用至其金鑰名稱字首為 `logs/` 之物件的規則。該規則指定下列動作：

- 兩個轉換動作：
  - 在建立物件的 30 天後將物件轉換為 S3 標準 – IA 儲存類別。
  - 在建立物件的 90 天後將物件轉換為 S3 Glacier Flexible Retrieval 儲存類別。
- 一個過期動作，指示 Amazon S3 在建立物件的一年之後將其刪除。

```
<LifecycleConfiguration>
  <Rule>
    <ID>example-id</ID>
    <Filter>
       <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <Transition>
      <Days>90</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Note

若所有的動作皆會套用至相同的一組物件，可以使用單一規則描述所有 S3 生命週期動作 (依篩選條件識別)。否則，您可以新增多個規則，然後每個規則指定不同的篩選條件。

Important

當您在 S3 生命週期組態中有多個規則時，一個物件可能會在同一天符合多個 S3 生命週期動作的資格。在這種情況下，Amazon S3 遵循以下一般規則：

- 永久刪除優先於轉換。
- 轉移優先於刪除標記的建立。
- 當物件符合 S3 冰川彈性擷取和 S3 標準 — IA (或 S3 單區域 — IA) 轉換的資格時，Amazon S3 會選擇 S3 冰川彈性擷取轉換。

如需範例，請參閱 範例 5：篩選條件重疊、生命週期動作相衝突，以及 Amazon S3 對為進行版本控制的儲存貯體所採取的動作。

範例 4：指定多項規則

若希望為不同的物件套用不同的 S3 生命週期動作，可以指定多項規則。下列 S3 生命週期組態有兩項規則：

- 規則 1 會套用到金鑰名稱字首為 `classA/` 的物件。該規則指示 Amazon S3 在建立物件的一年後，將其轉換為 S3 Glacier Flexible Retrieval 儲存類別，並在建立物件的 10 年後將其刪除。
- 規則 2 會套用到金鑰名稱字首為 `classB/` 的物件。其指示 Amazon S3 在建立物件的 90 天後，將其轉換為 S3 標準 – IA 儲存類別，並在建立物件的一年後將其刪除。

```
<LifecycleConfiguration>
    <Rule>
        <ID>ClassADocRule</ID>
        <Filter>
           <Prefix>classA/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
             <Days>3650</Days>
        </Expiration>
    </Rule>
    <Rule>
        <ID>ClassBDocRule</ID>
        <Filter>
```

```
            <Prefix>classB/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>90</Days>
           <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <Expiration>
             <Days>365</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

Important

當您在 S3 生命週期組態中有多個規則時，一個物件可能會在同一天符合多個 S3 生命週期動作的資格。在這種情況下，Amazon S3 遵循以下一般規則：

- 永久刪除優先於轉換。
- 轉移優先於刪除標記的建立。
- 當物件符合 S3 冰川彈性擷取和 S3 標準 — IA (或 S3 單區域 — IA) 轉換的資格時，Amazon S3 會選擇 S3 冰川彈性擷取轉換。

如需範例，請參閱 範例 5：篩選條件重疊、生命週期動作相衝突，以及 Amazon S3 對為進行版本控制的儲存貯體所採取的動作。

範例 5：篩選條件重疊、生命週期動作相衝突，以及 Amazon S3 對為進行版本控制的儲存貯體所採取的動作

您可能在指定的 S3 生命週期組態內，指定了重疊的字首或動作。

通常，S3 生命週期會針對成本進行最佳化。例如，如果兩個過期政策重疊，較短的過期政策會優先被接受，因此資料的存放週期會較預期為短。同樣的，如果兩個轉換原則重疊，S3 生命週期會將您的物件轉換至成本較低的儲存等級。

在這兩種情況下，S3 生命週期會嘗試為您選擇花費較少的途徑。此一般性規則的例外為 S3 Intelligent-Tiering 儲存類別。相較於其他儲存類別，S3 Intelligent-Tiering 較適用於 S3 生命週期，除了 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 儲存類別之外。

下列範例說明 Amazon S3 解決潛在衝突的方式。

Example 1：字首重疊 (無任何衝突)

下列範例組態有兩項規則指定的字首重疊，如下所示：

- 第一項規則指定的篩選條件為空白，代表所有位於儲存貯體中的物件。
- 第二項規則指定的金鑰名稱字首為 `logs/`，代表只有一部份的物件。

規則 1 要求 Amazon S3 在建立物件的一年後，刪除所有物件。規則 2 要求 Amazon S3 在建立 30 天後將物件子集轉換為 S3 標準 – IA 儲存類別。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>30</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

由於在這種情況下沒有衝突，因此 Amazon S3 會在建立 30 天後將具有 `logs/` 字首的物件轉換至 S3 標準 – IA 儲存類別。系統會刪除任何建立達到一年的物件。

Example 2：生命週期動作相衝突

在此範例組態中有兩個規則，分別指示 Amazon S3 對同一組的部分物件，在物件生命週期內的同一時間執行兩個不同的動作：

- 兩項規則都指定了相同的金鑰名稱字首，因此兩項規則都會套用到相同的一組部分物件。
- 套用規則時，兩項規則都相同地指定在建立物件的 365 天後。
- 一項規則指示 Amazon S3 將物件轉換為 S3 標準 – IA 儲存類別，另一項規則則指示 Amazon S3 在相同的時間將物件設為過期。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>365</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

在此情況下，因為您想要將物件設定為過期 (即將移除)，所以將其轉換為其他儲存類別不具任何意義，Amazon S3 因而會對這些物件選擇過期動作。

Example 3：字首重疊導致生命週期動作相衝突

在此範例中，組態有兩項規則，指定的重疊字首如下所示：

- 規則 1 指定的字首為空白 (代表所有物件)。
- 規則 2 指定了金鑰名稱字首 (logs/)，代表所有物件其中一部分。

對於金鑰名稱字首為 `logs/` 的一部分物件，將會套用兩項規則中的 S3 生命週期動作。其中一項規則指示 Amazon S3 在建立物件的 10 天後轉換物件，另一項規則則指示 Amazon S3 在建立物件的 365 天後轉換物件。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>10</Days>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>365</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

在此情況下，Amazon S3 會選擇在建立物件的 10 天進行轉換。

Example 4：使用標籤設定篩選條件以及所引發的生命週期動作衝突

假設您有下列 S3 生命週期組態，其中有兩項規則，每一項都指定了標籤篩選條件：

- 規則 1 指定了使用標籤來設定篩選條件 (`tag1/value1`)。此項規則指示 Amazon S3 在建立物件的 365 天後，將其轉換為 S3 Glacier Flexible Retrieval 儲存類別。
- 規則 2 指定了使用標籤來設定篩選條件 (`tag2/value2`)。此項規則指示 Amazon S3 在建立物件的 14 天後將物件設為過期。

S3 生命週期組態如以下範例所示。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Tag>
         <Key>tag1</Key>
         <Value>value1</Value>
      </Tag>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>GLACIER<StorageClass>
      <Days>365</Days>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Tag>
         <Key>tag2</Key>
         <Value>value2</Value>
      </Tag>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>14</Days>
    </Expiration>
   </Rule>
</LifecycleConfiguration>
```

如果物件擁有兩個標籤，則 Amazon S3 必須決定要遵循哪個規則。在這種情況下，Amazon S3 會選擇在建立物件的 14 天後將其設為過期。該物件將會移除，因此不會套用轉換動作。

Important

當您在 S3 生命週期組態中有多個規則時，一個物件可能會在同一天符合多個 S3 生命週期動作的資格。在這種情況下，Amazon S3 遵循以下一般規則：

- 永久刪除優先於轉換。
- 轉移優先於刪除標記的建立。

- 當物件符合 S3 冰川彈性擷取和 S3 標準 — IA (或 S3 單區域 — IA) 轉換的資格時，Amazon S3 會選擇 S3 冰川彈性擷取轉換。

如需範例，請參閱 範例 5：篩選條件重疊、生命週期動作相衝突，以及 Amazon S3 對為進行版本控制的儲存貯體所採取的動作。

範例 6：為已啟用版本控制的儲存貯體指定生命週期規則

假設您有一個已啟用版本控制的儲存貯體，就表示對於每個物件來說，您都有一個最新版本及零或多個非最新版本。(如需 S3 版本控制的詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。) 在此範例中，您希望維持為期一年的歷史記錄，然後刪除非最新版本。S3 生命週期組態支援保留 1 到 100 個版本的任何物件。

若要節省儲存成本，您會希望在非最新版本脫離最新版本 30 天後，將其移至 S3 Glacier Flexible Retrieval (假設這些非最新物件是您不需要即時存取的原始資料)。此外，您預期目前版本的存取頻率會在建立後 90 天減少，因此您可以選擇將這些物件移至 S3 標準 — IA 儲存類別。

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Filter>
           <Prefix></Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>90</Days>
           <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <NoncurrentVersionTransition>
            <NoncurrentDays>30</NoncurrentDays>
            <StorageClass>GLACIER</StorageClass>
        </NoncurrentVersionTransition>
       <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>5</NewerNoncurrentVersions>
            <NoncurrentDays>365</NoncurrentDays>
       </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

範例 7：移除過期物件刪除標記

對於每個物件來說，已啟用版本控制的儲存貯體會有一個目前的版本，以及零或多個非目前的版本。當您刪除物件時，請注意下列事項：

- 若在刪除請求中並未指定版本 ID，則 Amazon S3 會為物件新增刪除標記，而非直接刪除該物件。最新物件版本會變為非最新版本，且刪除標記將會成為最新版本。
- 若在刪除請求中指定了版本 ID，則 Amazon S3 會永久刪除物件版本 (不會建立刪除標記)。
- 沒有任何非最新版本的刪除標記，稱為過期物件刪除標記。

此範例說明在儲存貯體中建立過期物件刪除標記，以及使用 S3 生命週期組態來指示 Amazon S3 移除過期物件刪除標記的情況。

假設您撰寫 S3 生命週期組態，該組態使用該`NoncurrentVersionExpiration`動作在非最新版本變成非最新版本 30 天後移除，並保留最多 10 個非最新版本，如下列範例所示。

```
<LifecycleConfiguration>
    <Rule>
        ...
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

`NoncurrentVersionExpiration` 動作不適用最新版本的物件。只會移除非最新版本。

對於最新物件版本來說，您有下列選項可管理其生命週期，完全取決於最新物件版本是否妥善定義了生命週期：

- 最新物件版本遵循妥善定義的生命週期。

  在此情況下，可以使用 S3 生命週期組態搭配 `Expiration` 動作，指示 Amazon S3 移除最新版本，如以下範例所示。

  ```
  <LifecycleConfiguration>
      <Rule>
          ...
          <Expiration>
  ```

```
            <Days>60</Days>
        </Expiration>
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

在此範例中，Amazon S3 會在建立最新版本的 60 天後，藉由為每個最新物件版本新增一個刪除標記，將其移除。此程序會讓最新版本成為非最新版本，而刪除標記則會成為最新版本。如需詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。

> **Note**
>
> 您無法在同一個規則上同時指定 Days 和 ExpiredObjectDeleteMarker 標籤。在您指定 Days 標籤的情況下，一旦刪除標記足夠舊，符合留存時間標準，Amazon S3 將自動執行 ExpiredObjectDeleteMarker 清除。若要在刪除標記成為唯一版本後立即清除刪除標記，請建立僅包含 ExpiredObjectDeleteMarker 標籤的個別規則。

相同 S3 生命週期組態中的 NoncurrentVersionExpiration 動作，會在物件成為非最新版本的 30 天後，移除非最新版本物件。因此，在此範例中，所有物件版本會在建立物件 90 天後永久移除。雖然在此過程中會建立過期的物件刪除標記，但 Amazon S3 會為您偵測並移除已過期的物件刪除標記。

- 最新物件版本沒有妥善定義的生命週期。

  在此情況下，您可以在不需要物件的時候手動將其移除，透過一或多個非最新版本建立刪除標記。如果具有 NoncurrentVersionExpiration 動作的 S3 生命週期組態，移除了所有非最新版本，則您即會有已過期的物件刪除標記。

  S3 生命週期組態特別針對此案例提供 Expiration 動作，您可以使用此動作移除已過期的物件刪除標記。

```
<LifecycleConfiguration>
    <Rule>
       <ID>Rule 1</ID>
        <Filter>
          <Prefix>logs/</Prefix>
```

```
        </Filter>
        <Status>Enabled</Status>
        <Expiration>
           <ExpiredObjectDeleteMarker>true</ExpiredObjectDeleteMarker>
        </Expiration>
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

藉由在 `Expiration` 動作中將 `ExpiredObjectDeleteMarker` 元素設定為 `true`，您可以指示 Amazon S3 移除已過期的物件刪除標記。

> **Note**
>
> 使用 `ExpiredObjectDeleteMarker` S3 生命週期動作時，該規則無法指定使用標籤設定的篩選條件。

範例 8：中止分段上傳的生命週期組態

您可以使用 Amazon S3 分段上傳 REST API 操作，分段上傳大型物件。如需分段上傳的詳細資訊，請參閱「使用分段上傳來上傳和複製物件」。

透過使用 S3 生命週期組態，如果 Amazon S3 在啟動後的指定天數內未完成，您可以指示 Amazon S3 停止不完整的多部分上傳 (由規則中指定的金鑰名稱前置詞識別)。Amazon S3 中止分段上傳後，其會刪除所有與該分段上傳相關聯的部分。此過程可確保您不會有未完成之分段上傳的部分存放在 Amazon S3 中，從而協助您控制儲存成本。

> **Note**
>
> 使用 `AbortIncompleteMultipartUpload` S3 生命週期動作時，該規則無法指定使用標籤設定的篩選條件。

下列 S3 生命週期組態範例指定了一項規則，其會採取 `AbortIncompleteMultipartUpload` 動作。此動作會指示 Amazon S3 在分段上傳開始的七天後，停止未完成的分段上傳。

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Filter>
           <Prefix>SomeKeyPrefix/</Prefix>
        </Filter>
        <Status>rule-status</Status>
        <AbortIncompleteMultipartUpload>
          <DaysAfterInitiation>7</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

範例 9：使用以大小為基礎之規則的生命週期組態

您可以建立僅根據物件大小轉換物件的規則。您可以指定最小大小 (ObjectSizeGreaterThan) 或最大大小 (ObjectSizeLessThan)，或者您可以指定物件大小 (以位元組計) 的範圍。使用多個篩選條件時，例如字首和大小規則，您必須將篩選條件包裝在 <And> 元素中。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Transition with a prefix and based on size</ID>
    <Filter>
       <And>
          <Prefix>tax/</Prefix>
          <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
       </And>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

如果您同時使用 ObjectSizeGreaterThan 和 ObjectSizeLessThan 元素指定範圍，則物件大小上限必須大於物件大小下限。使用多個篩選條件時，您必須將篩選條件包裝在 <And> 元素中。下列範例會示範如何指定範圍介於 500 位元組和 64,000 個位元組之間的物件。當您指定範圍時，ObjectSizeGreaterThan和ObjectSizeLessThan篩選器會排除指定的值。如需詳細資訊，請參閱 the section called “Filter 元素”。

```
<LifecycleConfiguration>
    <Rule>
        ...
          <And>
             <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
             <ObjectSizeLessThan>64000</ObjectSizeLessThan>
          </And>
    </Rule>
</LifecycleConfiguration>
```

您也可以建立規則專門讓沒有任何資料的非最新物件過期，包括啟用版本控制的儲存貯體中建立的非最新刪除標記物件。下列範例使用 NoncurrentVersionExpiration 動作，在版本變成非最新版本後經過 30 天即將其移除，並保留最多 10 個非最新的物件版本。此外還會使用 ObjectSizeLessThan 元素只篩選出沒有資料的物件。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Expire noncurrent with size less than 1 byte</ID>
    <Filter>
       <ObjectSizeLessThan>1</ObjectSizeLessThan>
    </Filter>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
       <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
       <NoncurrentDays>30</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

Amazon S3 清查

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

您可以使用 Amazon S3 庫存清單來協助您管理儲存。例如，您可以用它來稽核及回報物件的複寫與加密狀態，以滿足業務、合規及法規需求。您也可以使用 Amazon S3 庫存清單來簡化及加速商業工作流程與大數據任務，該庫存清單提供了 Amazon S3 同步 `List` API 操作的排程替代方式。Amazon S3 庫存清單不會使用 `List` API 操作來稽核您的物件，也不會影響儲存貯體的請求率。

Amazon S3 庫存清單提供的逗號分隔值 (CSV)、Apache 最佳化行列式 (ORC)，或 Apache Parquet 輸出檔，每天或每週為 S3 儲存貯體或共用字首的物件 (也就是名稱開頭為共同字串的物件) 列出物件及其相對應的中繼資料。若您設定每週列出庫存清單，則會在初次報告後，於每個星期日 (UTC 時區) 產生一份報告。如需有關 Amazon S3 清查定價的資訊，請參閱 Amazon S3 定價。

您可以為儲存貯體設定多份清查清單。當您設定庫存清單時，可以指定下列項目：

- 要包括在庫存清單中的物件中繼資料
- 要列出所有物件版本或是只列出目前版本
- 儲存庫存清單檔案輸出的位置
- 要每天或每週產生庫存清單
- 是否要加密庫存清單檔案

您可以使用 Amazon Athena、Amazon Redshift Spectrum、Amazon Redshift Spectrum，以及 Presto、Apache Hive 和 Apache Spark 等其他工具，利用標準 SQL 來查詢 Amazon S3 庫存清單。如需使用 Athena 查詢庫存清單檔案的詳細資訊，請參閱 the section called “使用 Athena 查詢清查”。

來源與目的地儲存貯體

庫存清單列出其物件的儲存貯體稱為來源儲存貯體。而存放清查清單檔案的儲存貯體稱為目的地儲存貯體。

來源儲存貯體

清查會列出存放在來源儲存貯體中的物件。您可取得整個儲存貯體的庫存清單，或是依物件金鑰名稱字首篩選清單。

來源儲存貯體：

- 包含庫存清單中列出的物件
- 包含庫存清單的組態

Destination bucket (目標儲存貯體)

Amazon S3 清查清單檔案會寫入目的地儲存貯體。若要將所有庫存清單檔案集合在目的地儲存貯體中的共同位置，您可以在庫存清單組態中指定目的地字首。

目的地儲存貯體：

- 包含清查檔案清單。
- 包含的清單檔案會列出存放在目的地儲存貯體內之所有庫存清單檔案。如需詳細資訊，請參閱 清查資訊清單。
- 必須有儲存貯體政策，才能對 Amazon S3 授予許可來驗證儲存貯體擁有權，以及授予許可將檔案寫入儲存貯體。
- 必須與來源值區位於 AWS 區域 相同的位置。
- 可和來源儲存貯體相同。
- 可以由不同 AWS 帳戶 於擁有來源值區的帳戶所擁有。

Amazon S3 清查清單

清查清單檔案包含來源儲存貯體的物件清單，以及各物件的中繼資料。庫存清單檔案會以下列其中一種格式存放在目的地儲存貯體中：

- 使用 GZIP 壓縮的 CSV 檔案
- 使用 ZLIB 壓縮的 Apache 最佳化行列式 (ORC) 檔案
- 使用 Snappy 壓縮的 Apache Parquet 檔案

> **Note**
>
> 不保證 Amazon S3 庫存清單報告中的物件會依任何順序進行排序。

庫存清單檔案包含來源儲存貯體中物件的清單，以及所列出的每個物件的中繼資料。

- 儲存貯體名稱 – 要清查的儲存貯體名稱。
- 金鑰名稱 – 物件金鑰名稱 (或稱金鑰)，可唯一識別儲存貯體中的物件。若您使用 CSV 檔案格式，則金鑰名稱為 URL 編碼，且必須先解碼才能夠使用。

- 版本 ID – 物件版本 ID。當您對儲存貯體啟用版本控制時，Amazon S3 會將版本號碼指派給已新增至儲存貯體的物件。如需詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。(如果清單僅設定為物件的目前版本，則不包含此欄位。)
- IsLatest— True 如果物件是物件的目前版本，則設定為。(如果清單僅設定為物件的目前版本，則不包含此欄位。)
- 刪除標記 – 如果物件是刪除標記，則設為 True。如需詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。(若您尚未設定報告以納入所有您的物件版本，則此欄位將自動新增至您的報告)。
- 大小 - 物件大小 (以位元組為單位)，不包括未完成的分段上傳、物件中繼資料和刪除標記的大小。
- 上次修改日期 – 物件建立日期或上次修改日期，以最近者為準。
- ETag – 實體標籤 (ETag) 是物件的雜湊值。ETag 只會反映物件內容的變更，而非其中繼資料的變更。ETag 可以是物件資料的 MD5 Digest。取決於建立物件的方式，及其加密的方式。
- 儲存類別 - 用於存放物件的儲存類別。設定為 STANDARD、REDUCED_REDUNDANCY、STANDARD_IA、ONEZONE_IA、INTELLIGENT_TIERING、GL 如需詳細資訊，請參閱 使用 Amazon S3 儲存體方案。
- 分段上傳標記 – 如果物件是使用分段上傳方式上傳，則設為 True。如需詳細資訊，請參閱 使用分段上傳來上傳和複製物件。
- 複寫狀態 - 設定為 PENDING、COMPLETED、FAILED 或 REPLICA。如需詳細資訊，請參閱 取得複寫狀態資訊。
- 加密狀態 — 伺服器端加密狀態，視使用的加密金鑰類型而定 — Amazon S3 受管 (SSE-S3) 金鑰、() 金鑰 AWS Key Management Service (SSE-KMS AWS KMS) 或客戶提供的金鑰 (SSE-C)。設定為 SSE-S3、SSE-C、SSE-KMS 或 NOT-SSE。NOT-SSE 狀態表示物件未以伺服器端加密進行加密。如需詳細資訊，請參閱「使用加密來保護資料」。
- S3 物件鎖定保留截止日期 - 此日期之後才能刪除鎖定的物件。如需詳細資訊，請參閱 使用 S3 物件鎖定。
- S3 物件鎖定保留模式 - 針對鎖定的物件設為 Governance 或 Compliance。如需詳細資訊，請參閱 使用 S3 物件鎖定。
- S3 物件鎖定法務保存措施狀態 - 如果已將法務保存措施套用到物件，則設為 On。否則會設為 Off。如需詳細資訊，請參閱 使用 S3 物件鎖定。
- S3 Intelligent-Tiering 存取方案 - 物件的存取方案 (經常或不常) (如果儲存在 S3 Intelligent-Tiering 儲存類別中)。設定為 FREQUENT、INFREQUENT、ARCHIVE_INSTANT_ACCESS、ARCHIVE 或 DEEP_ARCHIVE。如需詳細資訊，請參閱 存取模式會變更或不明的自動最佳化資料的儲存體方案。
- S3 儲存貯體金鑰狀態 - 設定為 ENABLED 或 DISABLED。指出物件是否針對 SSE-KMS 使用 S3 儲存貯體金鑰。如需詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰。

- 檢查總和演算法 - 指出用來為物件建立檢查總和的演算法。
- 物件存取控制清單 — 每個物件的存取控制清單 (ACL)，可定義哪些 AWS 帳戶 或群組被授與此物件的存取權，以及授與的存取類型。「物件 ACL」欄位是以 JSON 格式定義。S3 庫存報告包含與來源儲存貯體中的物件相關聯的 ACL，即使儲存貯體的 ACL 已停用也一樣。如需詳細資訊，請參閱 使用物件 ACL 欄位 及 存取控制清單 (ACL) 概觀。

  > **Note**
  >
  > 「物件 ACL」欄位是以 JSON 格式定義。庫存清單報告會以 base64 編碼字串顯示「物件 ACL」欄位的值。
  > 例如，假設您有下列 JSON 格式的「物件 ACL」欄位：
  >
  > ```
  > {
  >         "version": "2022-11-10",
  >         "status": "AVAILABLE",
  >         "grants": [{
  >             "canonicalId": "example-canonical-user-ID",
  >             "type": "CanonicalUser",
  >             "permission": "READ"
  >         }]
  > }
  > ```
  >
  > 「物件 ACL」欄位會經過編碼，並顯示為下列 base64 編碼字串：
  >
  > ```
  > eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3siY2Fub25pY2Fs
  > ```
  >
  > 若要取得「物件 ACL」欄位的 JSON 解碼值，您可以在 Amazon Athena 中查詢此欄位。如需查詢範例，請參閱 使用 Amazon Athena 查詢 Amazon S3 庫存。

- 物件擁有者 - 物件的擁有者。

> **Note**
>
> 依據物件的生命週期組態，當物件的生命週期接近結尾時，Amazon S3 會將此物件排入佇列等待移除，並會以非同步方式進行移除物件。因此，過期日期與 Amazon S3 移除物件的日期之間，可能會有所延遲。清查報告包含已過期但尚未移除的物件。如需 S3 生命週期中過期動作的詳細資訊，請參閱 即將到期的物件。

建議建立刪除舊清查清單的生命週期政策。如需詳細資訊，請參閱 管理儲存生命週期。

`s3:PutInventoryConfiguration` 許可允許使用者在設定庫存清單時選取先前針對每個物件列出的所有中繼資料欄位，並指定目的地儲存貯體來存放庫存。對目的地儲存貯體中的物件具有讀取權限的使用者可以存取庫存清單中所有可用的物件中繼資料欄位。若要限制庫存報告的存取，請參閱 授予 S3 清查與 S3 分析的許可。

清查一致性

所有物件可能不會出現在每份清查清單中。庫存清單為新物件與覆寫項目的 PUT 請求及 DELETE 請求提供了最終一致性。儲存貯體的每一份庫存清單都是儲存貯體項目的快照。這些清單最終會趨於一致 (也就是說，清單可能不包括最近新增或刪除的物件)。

若要先驗證物件狀態再對物件採取動作，建議您執行 `HeadObject` REST API 請求，以擷取物件的中繼資料，或在 Amazon S3 主控台中檢查物件屬性。您也可以使用 AWS CLI 或 AWS SDK 檢查物件中繼資料。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 `HeadObject`。

如需有關使用 Amazon S3 清查的詳細資訊，請參閱下列主題。

主題

- 設定 Amazon S3 清查
- 設定清查完成的 Amazon S3 事件通知
- 尋找您的清查清單
- 使用 Amazon Athena 查詢 Amazon S3 庫存
- 將 Amazon S3 庫存報告中的空白版本 ID 字串轉換為空字串
- 使用物件 ACL 欄位

設定 Amazon S3 清查

Amazon S3 清查在您定義的排程上，提供物件及中繼資料的一般檔案清單。您可以將 S3 清查做為 Amazon S3 同步 `List` API 操作的另一種排程選擇。S3 清查提供以逗號分隔的值 (CSV)、Apache 最佳化資料列單欄式 (ORC) 或 Apache Parquet (Parquet) 輸出檔，其中列出您的物件及相應中繼資料。

您可以設定 S3 清查，為具有共同字首的 S3 儲存貯體和物件 (名稱以相同字串開頭的物件)，每天或每週建立清查清單。如需詳細資訊，請參閱 Amazon S3 清查。

本節說明如何設定清查，包括清查來源與目的地儲存貯體的詳細資訊。

主題

概觀

Amazon S3 清查可依定義的排程，建立 S3 儲存貯體中物件的清單，協助您管理儲存體。您可以為儲存貯體設定多份清查清單。會在目的地儲存貯體中將清查清單發佈為 CSV、ORC 或 Parquet 檔案。

設定庫存最簡單的方法是使用 Amazon S3 主控台，但您也可以使用 Amazon S3 REST API、 AWS Command Line Interface (AWS CLI) 或 AWS 開發套件。主控台會為您執行以下程序的第一項步驟：在目的地儲存貯體中新增儲存貯體政策。

設定 S3 儲存貯體的 Amazon S3 清查

1. 新增目標儲存貯體的儲存貯體政策。

   您必須在目的地儲存貯體上建立儲存貯體政策，以授予 Amazon S3 許可，以將物件寫入定義位置的儲存貯體。如需政策範例，請參閱「授予 S3 清查與 S3 分析的許可」。
2. 設定清查以列出來源儲存貯體的物件，並將清單發佈至目標儲存貯體。

   當您設定來源儲存貯體的清查清單時，要指定存放清單的目的地儲存貯體，以及每日或每週產生清單。您也可以設定是列出所有物件版本還是僅列出目前版本，以及要包含的物件中繼資料。

   S3 庫存報告組態中的某些物件中繼資料欄位是選用的，這表示預設情況下可以使用，但是當您授與使用者`s3:PutInventoryConfiguration`權限時，這些欄位可能會受到限制。您可以使用`s3:InventoryAccessibleOptionalFields`條件索引鍵，控制使用者是否可以在報表中包含這些選擇性的中繼資料欄位。

   如需 S3 庫存中可用的選用中繼資料欄位的詳細資訊，請參閱 Amazon 簡單儲存服務 API 參考OptionalFields中的。如需有關限制存取詳細目錄組態中某些選用中繼資料欄位的詳細資訊，請參閱控制 S3 庫存報告組態建立。

   您可以使用伺服器端加密搭配 Amazon S3 受管金鑰 (SSE-S3) 或 () 客戶受管金鑰 (SSE-KMS AWS KMS)，指定將庫存清單檔案加密。 AWS Key Management Service

> **Note**
>
> 不支援使用 S3 庫存進行 SSE-KMS 加密的 AWS 受管金鑰 (aws/s3)。

如需 SSE-S3 與 SSE-KMS 的詳細資訊，請參閱 使用伺服器端加密保護資料。若預計使用 SSE-KMS 加密，請參閱步驟 3。

- 如需如何使用主控台設定清查清單的資訊，請參閱「使用 S3 主控台設定清查」。
- 若要使用 Amazon S3 API 設定庫存清單，請使用 PutBucketInventoryConfigurationREST API 操作或 AWS CLI 或 AWS 開發套件中的對等項目。

3. 若要使用 SSE-KMS 加密清查清單檔案，請對 Amazon S3 授予許可使用 AWS KMS key。

   您可以使用 Amazon S3 主控台、Amazon S3 REST API 或 AWS 開發套件為庫存清單檔案設定加密。 AWS CLI無論選擇哪種方式，您必須對 Amazon S3 授予許可使用客戶受管金鑰來加密清查檔案。對 Amazon S3 授予許可時，請針對要用於加密清查檔案的客戶受管金鑰，修改金鑰政策。如需詳細資訊，請參閱 對 Amazon S3 授予許可使用您的客戶受管金鑰進行加密。

   儲存庫存清單檔案的目的地儲存貯體可由不同的 AWS 帳戶 擁有，不需與擁有來源儲存貯體的帳戶相同。如果您對 Amazon S3 庫存的跨帳戶操作使用 SSE-KMS 加密，建議您在設定 S3 庫存時使用完全合格的 KMS 金鑰 ARN。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 針對跨帳戶操作使用 SSE-KMS 加密 或 ServerSideEncryptionByDefault。

建立目的地儲存貯體政策

如果透過 S3 主控台建立庫存清單組態，Amazon S3 會自動在目的地儲存貯體上建立儲存貯體政策，將儲存貯體的寫入許可授予 Amazon S3。但是，如果您透過 AWS CLI、 AWS 開發套件或 Amazon S3 REST API 建立庫存組態，則必須在目標儲存貯體上手動新增儲存貯體政策。如需詳細資訊，請參閱 授予 S3 清查與 S3 分析的許可。S3 庫存目的地儲存貯體政策允許 Amazon S3 將庫存報告的資料寫入儲存貯體。

若在嘗試建立儲存貯體政策的時候發生錯誤，會為您提供修正方式的說明。例如，如果您選擇另一 AWS 帳戶 個目標值區中的目標值區，但沒有讀取和寫入值區政策的權限，您會看到錯誤訊息。

在這種情況下，目的地值區擁有者必須將值區政策新增至目的地值區。如果未將政策新增至目的地儲存貯體，您將不會取得清查報告，因為 Amazon S3 沒有目的地儲存貯體的寫入許可。若來源儲存貯體由不同的帳戶擁有，而非由目前的使用者擁有，則必須為政策中的來源儲存貯體擁有者替換正確的帳戶 ID。

對 Amazon S3 授予許可使用您的客戶受管金鑰進行加密

若要授與 Amazon S3 使用您的 AWS Key Management Service (AWS KMS) 客戶受管金鑰進行伺服器端加密的權限，您必須使用金鑰政策。若要更新您的金鑰政策，以便使用您的客戶受管金鑰，請依照下列步驟執行。

使用您的客戶受管金鑰授予 Amazon S3 加密許可

1. 使用擁 AWS 帳戶 有客戶管理金鑰的，登入 AWS Management Console.
2. 請在以下位置開啟 AWS KMS 主控台。 https://console.aws.amazon.com/kms
3. 若要變更 AWS 區域，請使用頁面右上角的「地區」選取器。
4. 在左側導覽窗格中，選擇 Customer managed keys (客戶受管金鑰)。
5. 在客戶受管金鑰下，選擇要用於加密清查檔案的客戶受管金鑰。
6. 在 Key policy (金鑰政策) 區段中，選擇 Switch to policy view (切換至政策檢視)。
7. 若要更新金鑰政策，請選擇 Edit (編輯)。
8. 在編輯金鑰政策頁面上，將下列幾行新增至現有的金鑰政策。針對 *source-account-id* 和 *example-s3-source-bucket*，提供您的使用案例適用的值。

```
{
    "Sid": "Allow Amazon S3 use of the customer managed key",
    "Effect": "Allow",
    "Principal": {
        "Service": "s3.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition":{
      "StringEquals":{
         "aws:SourceAccount":"source-account-id"
     },
      "ArnLike":{
        "aws:SourceARN": "arn:aws:s3:::example-s3-source-bucket"
     }
   }
}
```

9. 選擇儲存變更。

如需建立客戶受管金鑰和使用金鑰政策的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的下列連結：

- 管理金鑰
- 中的主要政策 AWS KMS

使用 S3 主控台設定清查

使用這些說明來設定使用 S3 主控台的清查。

> **Note**
>
> Amazon S3 最長可能需要 48 小時的時間才能提供第一份庫存清單報告。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。在儲存貯體清單中，選擇您要設定其 Amazon S3 庫存清單的儲存貯體名稱。
3. 選擇 Management (管理) 標籤，
4. 在 Inventory configurations (清查組態) 下，選擇 Create inventory configuration (建立清查組態)。
5. 在清查組態名稱中，輸入名稱。
6. 針對庫存清單範圍，執行下列操作：
   - 輸入選擇性字首。
   - 選擇要包含的物件版本：僅目前版本或包含所有版本。
7. 在 Report details (報告詳細資訊) 下，選擇要儲存報告的 AWS 帳戶 位置：This account (此帳戶) 或 A different account (不同帳戶)。
8. 在目的地下，選擇要儲存庫存清單報告的目的地儲存貯體。

   目的地值區必須與您 AWS 區域 要設定庫存的值區相同。目的地儲存貯體可位於不同的 AWS 帳戶中。指定目的地儲存貯體時，您也可以加入選用的字首，以將庫存清單報告集中在一起。

   在目的地儲存貯體欄位下，您會看到目的地儲存貯體許可陳述式，這會新增至目的地儲存貯體政策，以允許 Amazon S3 將資料放入該儲存貯體中。如需詳細資訊，請參閱 建立目的地儲存貯體政策。

9. 在頻率下，選擇產生報告的頻率：每日或每週。
10. 針對輸出格式，選擇下列其中一種報告格式：
    - CSV - 如果您打算使用此庫存清單報告進行 S3 批次操作，或是您想要在其他工具 (例如 Microsoft Excel) 中分析此報告，請選擇 CSV。
    - Apache ORC
    - Apache Parquet
11. 在 Status (狀態) 下，選擇 Enable (啟用) 或 Disable (停用)。
12. 若要設定伺服器端加密，請在清查報告加密之下,遵循下列步驟：
    a. 在 [伺服器端加密] 下，選擇 [不指定加密金鑰] 或 [指定要加密資料的加密金鑰]。
       - 若要在將物件存放到 Amazon S3 時，保留預設伺服器端加密的儲存貯體設定，請選擇不指定加密金鑰。只要儲存貯體目的地已啟用 S3 儲存貯體金鑰，複製操作便會在目的地儲存貯體中套用 S3 儲存貯體金鑰。

         > **Note**
         > 如果指定目標的儲存貯體政策要求物件先加密，然後再將物件存放到 Amazon S3，您必須選擇指定加密金鑰。否則，便無法將物件複製到目的地。

       - 若要先加密物件再存放到 Amazon S3，請選擇指定加密金鑰。

    b. 如果您選擇「指定加密金鑰」，則在「加密類型」下，您必須選擇 Amazon S3 受管金鑰 (SSE-S3) 或金AWS Key Management Service 鑰 (SSE-KMS)。

       SSE-S3 使用目前最強大的其中一種區塊加密法，也就是 256 位元進階加密標準 (AES-256)，來加密每個物件。SSE-KMS 可以讓您更完善地控制金鑰。如需 SSE-S3 的詳細資訊，請參閱使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密。如需 SSE-KMS 的詳細資訊，請參閱「使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)」。

       > **Note**
       > 若要使用 SSE-KMS 加密清查清單檔案，您必須對 Amazon S3 授予許可使用客戶受管金鑰。如需說明，請參閱對 Amazon S3 授予許可使用 KMS 金鑰進行加密。

    c. 如果您選擇金AWS Key Management Service 鑰 (SSE-KMS)，在下 AWS KMS key，您可以透過下列其中一個選項指定金 AWS KMS 鑰。

> **Note**
>
> 如果儲存詳細目錄清單檔案的目的地儲存貯體屬於不同的 AWS 帳戶，請確定您使用完整的 KMS 金鑰 ARN 來指定 KMS 金鑰。

- 若要從可用 KMS 金鑰清單中選擇，請選擇 [從您的金 AWS KMS 鑰中選擇]，然後從可用金鑰清單中選擇對稱加密 KMS 金鑰。確定 KMS 金鑰與儲存貯體位於相同的區域。

> **Note**
>
> AWS 受管金鑰 (aws/s3) 和客戶管理的金鑰都會出現在清單中。不過， AWS 受管金鑰 (aws/s3) 不支援使用 S3 庫存進行 SSE-KMS 加密。

- 若要輸入 KMS 金鑰 ARN，請選擇 [輸入金 AWS KMS 鑰 ARN]，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。
- 若要在 AWS KMS 主控台中建立新的客戶管理金鑰，請選擇 [建立 KMS 金鑰]。

13. 針對其他欄位，選取下列其中一項或多項以新增至庫存清單報告中：

- 大小 - 物件大小 (以位元組為單位)，不包括未完成的分段上傳、物件中繼資料和刪除標記的大小。
- 上次修改日期 – 物件建立日期或上次修改日期，以最近者為準。
- Multipart upload (分段上傳) – 指出物件的上傳方式為分段上傳。如需詳細資訊，請參閱「使用分段上傳來上傳和複製物件」。
- Replication status (複寫狀態) – 物件的複寫狀態。如需詳細資訊，請參閱 取得複寫狀態資訊。
- 加密狀態 - 用來加密物件的伺服器端加密類型。如需詳細資訊，請參閱 使用伺服器端加密保護資料。
- 值區索引鍵狀態 — 指出由所產生的儲存貯體層級金鑰是否 AWS KMS 套用至物件。如需詳細資訊，請參閱 使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本。
- 物件存取控制清單 — 每個物件的存取控制清單 (ACL)，可定義哪些 AWS 帳戶 或群組被授與此物件的存取權，以及授與的存取類型。如需此欄位的詳細資訊，請參閱 使用物件 ACL 欄位。如需 ACL 的詳細資訊，請參閱「存取控制清單 (ACL) 概觀」。
- 物件擁有者 - 物件的擁有者。
- 儲存類別 - 用於存放物件的儲存類別。

- Intelligent-Tiering：存取方案 - 指出物件的存取方案 (經常或不常) (如果儲存在 Intelligent-Tiering 儲存類別中)。如需詳細資訊，請參閱 存取模式會變更或不明的自動最佳化資料的儲存體方案。
- ETag – 實體標籤 (ETag) 是物件的雜湊值。ETag 只會反映物件內容的變更，而非其中繼資料的變更。ETag 可能是 (也可能不是) 物件資料的 MD5 摘要。取決於建立物件的方式，及其加密的方式。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 Object。
- 檢查總和演算法 - 說明用於建立物件的檢查總和演算法。
- 所有物件鎖定組態 - 物件的物件鎖定狀態，包括下列設定：
  - 物件鎖定：保留模式 - 套用於物件的保護層級：控管或合規。
  - 物件鎖定：保留截止日期 - 此日期之後才能刪除鎖定的物件。
  - 物件鎖定：法務保存措施狀態 - 鎖定物件的法務保存措施狀態。

  如需 S3 物件鎖定的詳細資訊，請參閱 S3 物件鎖定的運作方式。

如需清查報告內容的詳細資訊，請參閱 Amazon S3 清查清單。

如需有關限制存取詳細目錄組態中某些選用中繼資料欄位的詳細資訊，請參閱控制 S3 庫存報告組態建立。

14. 選擇建立。

發佈清查清單時，您可以使用 Amazon S3 Select 查詢清查清單檔案。如需如何使用 Amazon S3 Select 尋找清查清單並查詢清查清單檔案的詳細資訊，請參閱 尋找您的清查清單。

使用 REST API 搭配 S3 庫存清單

以下是您可以用來與 Amazon S3 庫存搭配使用的其餘作業。

- DeleteBucketInventoryConfiguration
- GetBucketInventoryConfiguration
- ListBucketInventoryConfigurations
- PutBucketInventoryConfiguration

設定清查完成的 Amazon S3 事件通知

您可以設定在建立資訊清單檢查總和時收到 Amazon S3 事件通知，其中指出清查清單已新增至目的地儲存貯體。資訊清單是 up-to-date 目的地位置上所有清單清單的清單。

Amazon S3 可以將事件發佈到 Amazon Simple Notification Service (Amazon SNS) 主題、Amazon Simple Queue Service (Amazon SQS) 佇列或 AWS Lambda 函數。如需詳細資訊，請參閱 Amazon S3 事件通知。

下列通知組態會定義所有最近新增至目標儲存貯體的 `manifest.checksum` 檔案，都由 AWS Lambda `cloud-function-list-write` 進行處理。

```
<NotificationConfiguration>
  <QueueConfiguration>
      <Id>1</Id>
      <Filter>
          <S3Key>
              <FilterRule>
                  <Name>prefix</Name>
                  <Value>destination-prefix/source-bucket</Value>
              </FilterRule>
              <FilterRule>
                  <Name>suffix</Name>
                  <Value>checksum</Value>
              </FilterRule>
          </S3Key>
      </Filter>
      <Cloudcode>arn:aws:lambda:us-west-2:222233334444:cloud-function-list-write</
Cloudcode>
      <Event>s3:ObjectCreated:*</Event>
  </QueueConfiguration>
  </NotificationConfiguration>
```

如需詳細資訊，請參閱 AWS Lambda 開AWS Lambda 發人員指南中的搭配 Amazon S3 使用。

尋找您的清查清單

發佈清查清單時，資訊清單檔案會發佈到目的地儲存貯體的以下位置。

```
destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.json
 destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.checksum
 destination-prefix/source-bucket/config-ID/hive/dt=YYYY-MM-DD-HH-MM/symlink.txt
```

- *destination-prefix* 是物件金鑰名稱字首，可在庫存清單組態中選擇性地指定。您可以使用此字首將所有庫存清單檔案集合到目的地儲存貯體內的共同位置。
- *source-bucket* 是庫存清單所適用的來源儲存貯體。當來自不同來源儲存貯體的多份庫存清單報告傳送至相同目的地儲存貯體時，會加入來源儲存貯體名稱以避免衝突。
- 當來自相同來源儲存貯體的多份庫存清單報告傳送至相同目的地儲存貯體時，會加入 *config-ID* 以避免衝突。*config-ID* 來自庫存清單報告組態，並且是設定時定義的報告名稱。
- *YYYY-MM-DDTHH-MMZ* 是時間戳記，由庫存清單報告產生程序開始掃描儲存貯體的開始時間與日期所組成，例如 2016-11-06T21-32Z。
- manifest.json 是資訊清單檔案。
- manifest.checksum 是 manifest.json 檔案內容的 MD5 雜湊。
- symlink.txt 是與 Apache Hive 相容的清單檔案。

清查清單會每日或每週發佈到目的地儲存貯體的以下位置。

```
destination-prefix/source-bucket/config-ID/data/example-file-name.csv.gz
...
destination-prefix/source-bucket/config-ID/data/example-file-name-1.csv.gz
```

- *destination-prefix* 是物件金鑰名稱字首，可在庫存清單組態中選擇性地指定。您可以使用此字首將所有庫存清單檔案集合到目的地儲存貯體中的共同位置。
- *source-bucket* 是庫存清單所適用的來源儲存貯體。當來自不同來源儲存貯體的多份庫存清單報告傳送至相同目的地儲存貯體時，會加入來源儲存貯體名稱以避免衝突。
- *example-file-name*.csv.gz 是其中一個 CSV 庫存檔案。ORC 清查名稱的結尾是副檔名 .orc，而 Parquet 清查名稱的結尾則是副檔名 .parquet。

  您可以使用 Amazon S3 Select 查詢庫存清單檔案。*# Amazon S3 ############## (##### ####/######/##### /## /.csv.gz)#example-file-name*然後，選擇物件動作和使用 S3 Select 查詢。如需如何使用 S3 Select 彙總函數查詢庫存清單檔案的範例，請參閱 SUM 範例。

清查資訊清單

資訊清單檔案 manifest.json 與 symlink.txt 能描述清查檔案的所在位置。每次交付新的清查清單時，都會伴隨一組新的資訊清單檔案。這些檔案可能會相互覆寫。在啟用版本控制的儲存貯體中，Amazon S3 會建立新版本的資訊清單檔案。

manifest.json 檔案內所包含的每個資訊清單檔案，都會提供清查中繼資料與其他基本資訊。此資訊包含下列項目：

- 來源儲存貯體名稱
- 目的地儲存貯體名稱
- 庫存清單的版本
- 採用 epoch 日期格式的建立時間戳記，內容是由庫存清單報告產生程序開始掃描儲存貯體的開始時間與日期所組成。
- 庫存清單檔案的格式與結構描述
- 目的地儲存貯體中的庫存清單檔案清單

每次寫入 manifest.json 檔案都會伴隨 manifest.checksum 檔案，該檔案為 manifest.json 檔案內容的 MD5 雜湊。

Example **manifest.json** 檔案中的清查資訊清單

以下範例顯示 manifest.json 檔案中 CSV、ORC 和 Parquet 格式庫存清單的庫存清單檔案。

CSV

以下清單檔案範例是 CSV 格式清查的 manifest.json 檔案。

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-inventory-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "CSV",
    "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker,
 Size, LastModifiedDate, ETag, StorageClass, IsMultipartUploaded,
 ReplicationStatus, EncryptionStatus, ObjectLockRetainUntilDate, ObjectLockMode,
 ObjectLockLegalHoldStatus, IntelligentTieringAccessTier, BucketKeyStatus,
 ChecksumAlgorithm, ObjectAccessControlList, ObjectOwner",
    "files": [
        {
            "key": "Inventory/example-source-bucket/2016-11-06T21-32Z/
files/939c6d46-85a9-4ba8-87bd-9db705a579ce.csv.gz",
            "size": 2147483647,
            "MD5checksum": "f11166069f1990abeb9c97ace9cdfabc"
        }
```

```
    ]
}
```

ORC

以下清單檔案範例是 ORC 格式清查的 `manifest.json` 檔案。

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "ORC",
    "fileSchema":
 "struct<bucket:string,key:string,version_id:string,is_latest:boolean,is_delete_marker:boole
    "files": [
        {
            "key": "inventory/example-source-bucket/data/
d794c570-95bb-4271-9128-26023c8b4900.orc",
            "size": 56291,
            "MD5checksum": "5925f4e78e1695c2d020b9f6eexample"
        }
    ]
}
```

Parquet

以下清單檔案範例是 Parquet 格式清查的 `manifest.json` 檔案。

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "Parquet",
    "fileSchema": "message s3.inventory { required binary bucket (UTF8);
 required binary key (UTF8); optional binary version_id (UTF8); optional boolean
 is_latest; optional boolean is_delete_marker; optional int64 size; optional
 int64 last_modified_date (TIMESTAMP_MILLIS); optional binary e_tag (UTF8);
 optional binary storage_class (UTF8); optional boolean is_multipart_uploaded;
 optional binary replication_status (UTF8); optional binary encryption_status
 (UTF8); optional int64 object_lock_retain_until_date (TIMESTAMP_MILLIS); optional
 binary object_lock_mode (UTF8); optional binary object_lock_legal_hold_status
```

```
 (UTF8); optional binary intelligent_tiering_access_tier (UTF8); optional binary
 bucket_key_status (UTF8); optional binary checksum_algorithm (UTF8); optional
 binary object_access_control_list (UTF8); optional binary object_owner (UTF8);}",
    "files": [
        {
           "key": "inventory/example-source-bucket/data/
d754c470-85bb-4255-9218-47023c8b4910.parquet",
            "size": 56291,
            "MD5checksum": "5825f2e18e1695c2d030b9f6eexample"
        }
    ]
}
```

`symlink.txt` 檔案是與 Apache Hive 相容的清單檔案，可讓 Hive 自動探索庫存清單檔案及其相關資料檔案。Hive 相容的清單檔案適用於 Hive 相容的服務：Athena 和 Amazon Redshift Spectrum。它也可以搭配與 Hive 相容的應用程式使用，包括 Presto、Apache Hive、Apache Spark 及其他許多應用程式。

> **Important**
>
> `symlink.txt` Apache Hive 相容的清單檔案目前不適用於 AWS Glue。
> 不支援針對 ORC 和 Parquet 格式的庫存清單檔案使用 Apache Hive 和 Apache Spark 讀取 `symlink.txt` 檔案。

使用 Amazon Athena 查詢 Amazon S3 庫存

您可以在所有提供 Athena 的區域中，使用 Amazon Athena 透過標準 SQL 查詢來查詢 Amazon S3 庫存清單檔案。若要檢查 AWS 區域 可用性，請參閱《AWS 區域 資料表》。

Athena 可查詢採用 Apache 最佳化行列式 (ORC)、Apache Parquet 或逗號分隔值 (CSV) 格式的 Amazon S3 庫存清單檔案。當您使用 Athena 查詢庫存清單檔案時，建議您使用 ORC 格式或 Parquet 格式的庫存清單檔案。ORC 和 Parquet 格式提供更快的查詢效能及較低的查詢成本。ORC 和 Parquet 都是自我描述且具類型感知功能的單欄式檔案格式，專為 Apache Hadoop 所設計。此分欄式格式可讓閱讀的人只讀取、解壓縮及處理目前查詢所需要的欄。所有 AWS 區域皆提供 ORC 和 Parquet 格式的 Amazon S3 庫存清單。

使用 Athena 查詢 Amazon S3 庫存清單檔案

1. 建立 Athena 資料表。如需有關建立資料表的資訊，請參閱《Amazon Athena 使用者指南》中的在 Amazon Athena 中建立資料表。
2. 根據您要查詢的是 ORC 格式、Parquet 格式或 CSV 格式的庫存清單報告而定，使用下列其中一種範例查詢範本來建立查詢。

   - 若您使用 Athena 查詢 ORC 格式的庫存清單報告，請使用以下範例查詢作為範本。

     下列範例查詢包含 ORC 格式庫存清單報告中的所有選用欄位。

     若要使用此範例查詢，請執行下列操作：

     - 將 *your_table_name* 替換成您建立的 Athena 表格名稱。
     - 將您未針對庫存清單選擇的任何選用欄位移除，如此就能讓查詢對應您為庫存清單選擇的欄位。
     - 將下列儲存貯體名稱和庫存清單位置 (組態 ID) 替換為適合您組態的值。

       s3://*DOC-EXAMPLE-BUCKET*/*config-ID*/hive/

     - 將 projection.dt.range 底下的 *2022-01-01-00-00* 日期替換為您在 Athena 中分割資料的時間範圍內的第一天。如需詳細資訊，請參閱在 Athena 中分割資料。

```
CREATE EXTERNAL TABLE your_table_name(
         bucket string,
         key string,
         version_id string,
         is_latest boolean,
         is_delete_marker boolean,
         size bigint,
         last_modified_date timestamp,
         e_tag string,
         storage_class string,
         is_multipart_uploaded boolean,
         replication_status string,
         encryption_status string,
         object_lock_retain_until_date bigint,
         object_lock_mode string,
         object_lock_legal_hold_status string,
         intelligent_tiering_access_tier string,
         bucket_key_status string,
         checksum_algorithm string,
```

```
         object_access_control_list string,
         object_owner string
) PARTITIONED BY (
        dt string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
  STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
  LOCATION 's3://source-bucket/config-ID/hive/'
  TBLPROPERTIES (
    "projection.enabled" = "true",
    "projection.dt.type" = "date",
    "projection.dt.format" = "yyyy-MM-dd-HH-mm",
    "projection.dt.range" = "2022-01-01-00-00,NOW",
    "projection.dt.interval" = "1",
    "projection.dt.interval.unit" = "HOURS"
  );
```

- 若您使用 Athena 查詢 Parquet 格式的庫存清單報告，請使用 ORC 格式報告的範例查詢。不過，請使用下列 Parquet SerDe 取代 `ROW FORMAT SERDE` 陳述式中的 ORC SerDe。

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
```

- 若您使用 Athena 查詢 CSV 格式的庫存清單報告，請使用以下範例查詢作為範本。

  下列範例查詢包含 CSV 格式庫存清單報告中的所有選用欄位。

  若要使用此範例查詢，請執行下列操作：

  - 將 *your_table_name* 替換成您建立的 Athena 表格名稱。
  - 將您未針對庫存清單選擇的任何選用欄位移除，如此就能讓查詢對應您為庫存清單選擇的欄位。
  - 將下列儲存貯體名稱和庫存清單位置 (組態 ID) 替換為適合您組態的值。

    s3://*DOC-EXAMPLE-BUCKET*/*config-ID*/hive/

  - 將 `projection.dt.range` 底下的 *2022-01-01-00-00* 日期替換為您在 Athena 中分割資料的時間範圍內的第一天。如需詳細資訊，請參閱在 Athena 中分割資料。

```
CREATE EXTERNAL TABLE your_table_name(
         bucket string,
         key string,
         version_id string,
```

```
         is_latest boolean,
         is_delete_marker boolean,
         size string,
         last_modified_date string,
         e_tag string,
         storage_class string,
         is_multipart_uploaded boolean,
         replication_status string,
         encryption_status string,
         object_lock_retain_until_date string,
         object_lock_mode string,
         object_lock_legal_hold_status string,
         intelligent_tiering_access_tier string,
         bucket_key_status string,
         checksum_algorithm string,
         object_access_control_list string,
         object_owner string
) PARTITIONED BY (
        dt string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
  STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
  LOCATION 's3://source-bucket/config-ID/hive/'
  TBLPROPERTIES (
    "projection.enabled" = "true",
    "projection.dt.type" = "date",
    "projection.dt.format" = "yyyy-MM-dd-HH-mm",
    "projection.dt.range" = "2022-01-01-00-00,NOW",
    "projection.dt.interval" = "1",
    "projection.dt.interval.unit" = "HOURS"
  );
```

3. 現在您可以對庫存清單執行各種不同的查詢，如下列範例中所示。將每個 *user input placeholder* 替換成您自己的資訊。

```
# Get a list of the latest inventory report dates available.
SELECT DISTINCT dt FROM your_table_name ORDER BY 1 DESC limit 10;

# Get the encryption status for a provided report date.
SELECT encryption_status, count(*) FROM your_table_name WHERE dt = 'YYYY-MM-DD-HH-
MM' GROUP BY encryption_status;
```

```
# Get the encryption status for inventory report dates in the provided range.
SELECT dt, encryption_status, count(*) FROM your_table_name
WHERE dt > 'YYYY-MM-DD-HH-MM' AND dt < 'YYYY-MM-DD-HH-MM' GROUP BY dt,
 encryption_status;
```

若您設定 S3 庫存清單將「物件存取控制清單 (物件 ACL)」欄位新增至庫存清單報告中，則報告會以 base64 編碼字串顯示「物件 ACL」欄位的值。若要取得「物件 ACL」欄位的 JSON 解碼值，您可以使用 Athena 查詢此欄位。請參閱以下查詢範例。如需「物件 ACL」欄位的詳細資訊，請參閱 使用物件 ACL 欄位。

```
# Get the S3 keys that have Object ACL grants with public access.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'uri') = 'http://acs.amazonaws.com/groups/global/AllUsers'
```

```
# Get the S3 keys that have Object ACL grantees in addition to the object owner.
WITH grants AS
    (SELECT key,
    from_utf8(from_base64(object_access_control_list)) AS
 object_access_control_list,
         object_owner,
         CAST(json_extract(from_utf8(from_base64(object_access_control_list)),
         '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))) AS grants_array
    FROM your_table_name)
SELECT key,
       grant,
       objectowner
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE cardinality(grants_array) > 1 AND element_at(grant, 'canonicalId') !=
 object_owner;
```

```
# Get the S3 keys with READ permission that is granted in the Object ACL.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'permission') = 'READ';
```

```
# Get the S3 keys that have Object ACL grants to a specific canonical user ID.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'canonicalId') = 'user-canonical-id';
```

```
# Get the number of grantees on the Object ACL.
SELECT key,
       object_access_control_list,
       json_array_length(json_extract(object_access_control_list,'$.grants')) AS
 grants_count
FROM your_table_name;
```

如需有關使用 Athena 的詳細資訊，請參閱《Amazon Athena 使用者指南》https://docs.aws.amazon.com/athena/latest/ug/。

將 Amazon S3 庫存報告中的空白版本 ID 字串轉換為空字串

> **Note**
>
> 下列處理程序僅適用於包含所有版本的 Amazon S3 庫存報告，且僅當「所有版本」報告用作已啟用 S3 版本控制之儲存貯體上 S3 批次操作的資訊清單時。您不需要為僅指定最新版本的 S3 庫存報告轉換字串。

您可以使用 S3 庫存報告作為 S3 批次操作的資訊清單。不過，在儲存貯體上啟用 S3 版本控制時，包含所有版本的 S3 庫存報告會在版本 ID 欄位中以空字串標示任何未進行版本控制的物件。當庫存報告包含所有物件版本 ID 時，批次操作會辨識 `null` 字串作為版本 ID，但不能為空字串。

當 S3 批次操作任務使用「所有版本」S3 庫存報告作為資訊清單時，在版本 ID 欄位中具有空字串之物件上的所有任務都會失敗。若要將 S3 庫存報告的版本 ID 欄位中的空字串轉換為批次操作的 `null` 字串，請使用以下處理程序。

更新 Amazon S3 庫存報告以搭配批次操作使用

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 導覽至您的 S3 庫存報告。庫存報告位於您在設定庫存報告時所指定的目的地儲存貯體。如需有關尋找庫存報告的詳細資訊，請參閱 尋找您的清查清單。
    a. 選擇目的地儲存貯體。
    b. 選擇 資料夾。以原始來源儲存貯體命名資料夾。
    c. 選擇以庫存組態命名的資料夾。
    d. 選取名為 hive 資料夾旁邊的核取方塊。在頁面頂端，選擇 Copy S3 URI (複製 S3 URI)，複製資料夾的 S3 URI。
3. 前往 https://console.aws.amazon.com/athena/ 開啟 Amazon Athena 主控台。
4. 在查詢編輯器中，選擇 Settings (設定)，然後選擇 Manage (管理)。在 Manage settings (管理設定) 頁面，對於 Location of query result (查詢結果的位置)，請選擇 S3 儲存貯體來存放查詢結果。

5. 在查詢編輯器中，建立 Athena 資料表，以使用下列命令保留庫存報告中的資料。使用您選擇的名稱取代 *table_name*，並在 LOCATION 子句中，插入您先前複製的 S3 URI。然後選擇 Run (執行) 來執行查詢。

```
CREATE EXTERNAL TABLE table_name(bucket string, key string,
 version_id string) PARTITIONED BY (dt string)ROW FORMAT SERDE
 'org.apache.hadoop.hive.serde2.OpenCSVSerde'STORED AS INPUTFORMAT
 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat' OUTPUTFORMAT
 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat' LOCATION 'Copied S3 URI';
```

6. 若要清除查詢編輯器，請選擇 Clear (清除)。之後，使用下列命令將庫存報告載入資料表。使用您在上一個步驟中選擇的項目取代 *table_name*。然後選擇 Run (執行) 來執行查詢。

```
MSCK REPAIR TABLE table_name;
```

7. 若要清除查詢編輯器，請選擇 Clear (清除)。執行下列 SELECT 查詢來擷取原始庫存報告中的所有項目，並將任何空白的版本 ID 取代為 null 字串。使用您之前選擇的項目取代 *table_name*，並使用您想要此工具在庫存報告上執行的日期取代 WHERE 子句中的 *YYYY-MM-DD-HH-MM*。然後選擇 Run (執行) 來執行查詢。

```
SELECT bucket as Bucket, key as Key, CASE WHEN version_id = '' THEN 'null' ELSE
 version_id END as VersionId FROM table_name WHERE dt = 'YYYY-MM-DD-HH-MM';
```

8. 退回 Amazon S3 主控台 (https://console.aws.amazon.com/s3/)，並導覽至您之前為 Location of query result (查詢結果位置) 選擇的 S3 儲存貯體。其中應該有一系列以日期結尾的資料夾。

   例如，您應該會看到類似 s3://***DOC-EXAMPLE-BUCKET***/***query-result-location***/Unsaved/2021/10/07/。您應該會看到包含您所執行之 SELECT 查詢結果的 .csv 檔案。

   選擇具有最新修改日期的 CSV 檔案。將此檔案下載到您的本機電腦，以進行下一個步驟。

9. 產生的 CSV 檔案包含標頭列。若要使用此 CSV 檔案作為 S3 批次操作任務的輸入，您必須移除標頭列，因為批次操作不支援 CSV 資訊清單上的標頭列。

   若要移除標頭列，您可以對檔案執行下列其中一個命令。用您的 CSV 檔案名稱取代 *file.csv*。

   對於 macOS 和 Linux 電腦，請在終端機視窗中執行 tail 命令。

```
tail -n +2 file.csv > tmp.csv && mv tmp.csv file.csv
```

對於 Windows 電腦，請在 Windows PowerShell 視窗中執行下列指令碼。將 *File-location* 替換為您的檔案路徑，並將 *file.csv* 替換為檔案名稱。

```
$ins = New-Object System.IO.StreamReader File-location\file.csv
$outs = New-Object System.IO.StreamWriter File-location\temp.csv
try {
    $skip = 0
    while ( !$ins.EndOfStream ) {
        $line = $ins.ReadLine();
        if ( $skip -ne 0 ) {
            $outs.WriteLine($line);
        } else {
            $skip = 1
        }
    }
} finally {
    $outs.Close();
    $ins.Close();
}
Move-Item File-location\temp.csv File-location\file.csv -Force
```

10. 從 CSV 檔案中移除標頭列之後，您就可以在 S3 批次操作任務中將其用作資訊清單。將 CSV 檔案上傳到 S3 儲存貯體或您選擇的位置，然後使用 CSV 檔案作為資訊清單建立批次操作任務。

    如需有關建立 S3 批次操作任務的詳細資訊，請參閱 建立 S3 批次操作任務。

使用物件 ACL 欄位

Amazon S3 庫存清單報告中包含 S3 來源儲存貯體的物件清單，以及各物件的中繼資料。「物件存取控制清單 (ACL)」欄位是 Amazon S3 庫存清單中提供的中繼資料欄位。具體而言，「物件 ACL」欄位包含每個物件的存取控制清單 (ACL)。物件的 ACL 會定義哪些 AWS 帳戶 或群組被授與此物件的存取權，以及授與的存取權類型。如需詳細資訊，請參閱 存取控制清單 (ACL) 概觀 及 Amazon S3 清查清單。

Amazon S3 庫存清單報告中的「物件 ACL」欄位是以 JSON 格式定義。JSON 資料包括下列欄位：

- `version` - 庫存清單報告中「物件 ACL」欄位格式的版本。其為日期格式 `yyyy-mm-dd`。
- `status` - 可能的值為 `AVAILABLE` 或 `UNAVAILABLE`，以指出物件 ACL 是否適用於物件。若物件 ACL 的狀態為 `UNAVAILABLE`，庫存清單報告中「物件擁有者」欄位的值也會是 `UNAVAILABLE`。

- `grants` - 承授者-許可配對，這裡會列出物件 ACL 授與的每個承授者的狀態。承授者可用的值為 `CanonicalUser` 和 `Group`。如需承授者的詳細資訊，請參閱存取控制清單中的承授者。

  若是 `Group` 類型的承授者，承授者-許可配對會包括下列屬性：

  - `uri` - 預先定義的 Amazon S3 群組。
  - `permission` - 物件獲得授與的 ACL 許可。如需詳細資訊，請參閱物件的 ACL 許可。
  - `type` - `Group` 類型，表示承授者為群組。

  若是 `CanonicalUser` 類型的承授者，承授者-許可配對會包括下列屬性：

  - `canonicalId` - 混淆形式的 AWS 帳戶 ID。的標準使用者 ID 專屬 AWS 帳戶 於該帳戶。您可以檢索規範用戶 ID。如需詳細資訊，請參閱《AWS 帳戶管理參考指南》中的尋找您 AWS 帳戶的標準使用者 ID。

    > **Note**
    >
    > 如果 ACL 中的受權者是的電子郵件地址 AWS 帳戶，則 S3 庫存會使用該項目`canonicalId`的 AWS 帳戶 和`CanonicalUser`類型來指定此受權者。如需詳細資訊，請參閱存取控制清單中的承授者。

  - `permission` - 物件獲得授與的 ACL 許可。如需詳細資訊，請參閱物件的 ACL 許可。
  - `type`— 類型`CanonicalUser`，表示受權者為. AWS 帳戶

下列範例顯示 JSON 格式的「物件 ACL」欄位的可能值：

```
{
    "version": "2022-11-10",
    "status": "AVAILABLE",
    "grants": [{
        "uri": "http://acs.amazonaws.com/groups/global/AllUsers",
        "permission": "READ",
        "type": "Group"
    }, {
        "canonicalId": "example-canonical-id",
        "permission": "FULL_CONTROL",
        "type": "CanonicalUser"
    }]
}
```

> **Note**
>
> 「物件 ACL」欄位是以 JSON 格式定義。庫存清單報告會以 base64 編碼字串顯示「物件 ACL」欄位的值。
> 例如，假設您有下列 JSON 格式的「物件 ACL」欄位：
>
> ```
> {
>         "version": "2022-11-10",
>         "status": "AVAILABLE",
>         "grants": [{
>             "canonicalId": "example-canonical-user-ID",
>             "type": "CanonicalUser",
>             "permission": "READ"
>         }]
> }
> ```
>
> 「物件 ACL」欄位會經過編碼，並顯示為下列 base64 編碼字串：
>
> ```
> eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3siY2Fub25pY2FsSW
> ```
>
> 若要取得「物件 ACL」欄位的 JSON 解碼值，您可以在 Amazon Athena 中查詢此欄位。如需查詢範例，請參閱 使用 Amazon Athena 查詢 Amazon S3 庫存。

複製物件概觀

您可以使用複寫來啟用跨 Amazon S3 儲存貯體的物件自動異步複製。設定用於物件複寫的儲存貯體可由相同 AWS 帳戶 或不同帳戶擁有。您可將物件複寫到單一目的地儲存貯體或多個目的地儲存貯體。目的地值區可以位於與來源值區不同 AWS 區域 的區域內，也可以位於相同的「區域」中。

複寫有兩種類型：即時複寫和隨選複寫。

- 即時複製 — 若要在新物件和更新物件寫入來源值區時自動複製這些物件，請使用即時複製。即時複製在您設定複寫之前，不會複製值區中存在的任何物件。若要複製在設定複製之前已存在的物件，請使用隨選複寫。
- 隨需複寫 — 若要視需求將現有物件從來源儲存貯體複寫到一或多個目的地儲存貯體，請使用 S3 Batch 複寫。如需複寫現有物件的詳細資訊，請參閱何時使用 S3 批次複寫。

有兩種形式的即時複寫：跨區域複寫 (CRR) 和單一區域複寫 (SRR)。

- 跨區域複寫 (CRR) — 您可以使用 CRR 在不同的 Amazon S3 儲存貯體之間複寫物件。 AWS 區域如需 CRR 的詳細資訊，請參閱the section called “何時使用跨區域複寫”。
- 單一區域複寫 (SRR) — 您可以使用 SRR 在相同的 Amazon S3 儲存貯體之間複製物件。 AWS 區域如需 SRR 的詳細資訊，請參閱the section called “何時使用相同區域複寫”。

主題

為什麼要使用複寫？

複寫可協助您執行以下項目：

- 保留中繼資料的同時複寫物件 – 您可以使用複寫，來製作可保留所有中繼資料 (例如原始物件建立時間和版本 ID) 的物件複本。如果您需要確保複本與來源物件相同，則此功能非常重要。
- 將物件複寫到不同的儲存類別 – 您可以使用複寫直接將物件放置在目的地儲存貯體中的 S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive 或其他儲存類別。您也可以複寫資料到相同的儲存體類別，並在目的地儲存貯體上使用生命週期組態，來隨著存在時間移動物件到較冷的儲存體。
- 在不同擁有權下維護物件副本 — 無論誰擁有來源物件，您都可以告訴 Amazon S3 將複本擁有權變更為擁 AWS 帳戶 有目標儲存貯體的擁有權。這稱為擁有者覆寫選項。您可以使用這個選項來限制物件複本的存取。

- 將物件儲存在多個位置 AWS 區域— 為了確保資料保存位置的地理差異，您可以跨不同的目標值區設定多個目標值區 AWS 區域。此功能可以協助您滿足特定的合規要求。
- 在 15 分鐘內複寫物件 — 若要在可預測的時間範圍內在相同 AWS 區域 或不同區域複寫資料，您可以使用 S3 複寫時間控制 (S3 RTC)。S3 RTC 會在 15 分鐘內，複寫 99.99% 在 Amazon S3 中存放的新物件 (由服務水準協議支援)。如需詳細資訊，請參閱 the section called “使用 S3 複寫時間控制”。

  > **Note**
  >
  > S3 RTC 不適用於批次複寫。批次複寫是隨需複寫任務，可以使用 S3 批次操作來追蹤。如需詳細資訊，請參閱 追蹤任務狀態和完成報告。

- 同步儲存貯體、複寫現有物件，以及複寫先前失敗或已複寫的物件 – 若要同步儲存貯體和複寫現有物件，請使用批次複寫做為隨需複寫動作。如需何時使用批次複寫的詳細資訊，請參閱何時使用 S3 批次複寫。
- 複寫物件並容錯移轉到另一個 AWS 區域中的儲存貯體 - 若要在資料複寫期間跨儲存貯體保持所有中繼資料和物件同步，請在設定 Amazon S3 多區域存取點容錯移轉控制之前，使用雙向複寫規則。雙向複寫規則有助於確保將資料寫入流量容錯移轉至其中的 S3 儲存貯體時，該資料隨後會複寫回來源儲存貯體。

何時使用跨區域複寫

S3 跨區域複寫 (CRR) 用於在不同 AWS 區域中跨 Amazon S3 儲存貯體複製物件。CRR 可協助您執行以下作業：

- 達到合規要求 – 雖然 Amazon S3 預設會跨多個地理位置遙遠的可用區域存放您的資料，但合規要求可能讓您必須將資料存放在更遠的距離。若要滿足這些要求，請使用跨區域複寫來在相距遙遠的 AWS 區域之間複寫資料。
- 最小化延遲 — 如果您的客戶位於兩個地理位置，您可以將存取物件的延遲降到最低，方法是在 AWS 區域 地理位置上更接近使用者的物件複本。
- 提高作業效率 — 如果您在兩個不同 AWS 區域 的運算叢集中分析同一組物件，則可以選擇在這些區域中維護物件複本。

何時使用相同區域複寫

相同區域複寫 (SRR) 用於在相同 AWS 區域中跨 Amazon S3 儲存貯體複製物件。SRR 可協助您執行以下作業：

- 將日誌匯總到單一儲存貯體 – 如果您在多個儲存貯體或跨多個帳戶存放日誌，您可以輕鬆將日誌複寫到單一、區域內的儲存貯體。這麼做可讓您更容易處理單一位置中的日誌。
- 設定生產和測試帳戶間的即時複寫 – 如果您或客戶的生產和測試帳戶使用相同資料，您可以在這些多個帳戶間複寫物件，同時維護物件中繼資料。
- 遵守資料主權法律 — 您可能需要在特定區域 AWS 帳戶 內單獨儲存多份資料副本。當合規法規不允許資料離開您的國家時，相同區域內的複寫可協助您自動複寫重要資料。

使用雙向複寫的時機

- 跨多個建立共用資料集 AWS 區域 — 使用複本修改同步功能，您可以輕鬆地在複寫物件上複製中繼資料變更，例如物件存取控制清單 (ACL)、物件標籤或物件鎖定。如果您想要讓所有物件和物件中繼資料變更保持同步，則此雙向複寫很重要。在相同或不同 AWS 區域的兩個以上儲存貯體之間執行雙向複寫時，您可以在新的或現有的複寫規則上啟用複本修改同步。
- 在容錯移轉期間 AWS 區域 保持跨區域的資料同步 — 您可以直接從多區域存取點設定雙向複寫規則，使用 S3 跨區域複寫 (CRR) 設定雙向複寫規則，同步儲存貯體中的資料。若要在何時啟動容錯移轉做出明智的決定，您也可以啟用 S3 複寫指標，以便在 Amazon CloudWatch、S3 複寫時間控制 (S3 RTC) 或從多區域存取點監控複寫。
- 讓您的應用程式高度可用 - 即使在區域流量中斷的情況下，您也可以使用雙向複寫規則，在資料複寫期間，跨儲存貯體讓所有中繼資料和物件保持同步。

何時使用 S3 批次複寫

批次複寫做為隨需選項，將現有物件複寫到不同的儲存貯體。與即時複寫不同，這些任務可以視需要執行。批次複寫可協助您執行以下項目：

- 複寫現有物件 – 您可以使用批次複寫來複寫在設定相同區域複寫或跨區域複寫之前，即已新增到儲存貯體中的物件。
- 複寫先前複寫失敗的物件 – 您可以篩選批次複寫任務，以嘗試複寫其複寫狀態為 FAILED 的物件。
- 複寫已複寫過的物件 – 您可能需要將資料的多個副本存放在不同的 AWS 帳戶 或 AWS 區域。批次複寫可以將現有物件複寫到新增的目的地。

- 複寫根據複寫規則建立之物件的複本 – 複寫組態會在目的地儲存貯體中建立物件的複本。物件的複本只能使用批次複寫來複寫。

工作負載需求和即時複製

視您的工作負載需求而定，某些複製類型會比其他使用案例更適合您的使用案例。使用下表決定要針對您的情況使用哪種複寫類型，以及是否針對您的工作負載使用 S3 複寫時間控制 (S3 RTC)。S3 RTC 會在 15 分鐘內複寫存放在 Amazon S3 中 99.99% 的新物件 (由服務層級協議或 SLA 支援)。如需詳細資訊，請參閱 the section called “使用 S3 複寫時間控制”。

複製比較的工作負載需求

工作量要求	中三 RTC (15 分鐘服務水準協定)	跨區域複寫 (CRR)	單一區域複寫 (SRR)
在不同之間複製物件 AWS 帳戶	是	是	是
在 24-48 小時內複製相同 AWS 區域 物件 (不支援 SLA)	否	否	是
AWS 區域 在 24-48 小時內複製不同物件 (不支援 SLA)	否	是	否
可預測的複寫時間：由 SLA 支援，可在 15 分鐘內複寫 99.9% 的物件	是	否	否

Amazon S3 複寫的內容？

Amazon S3 僅會複寫儲存貯體中已設定複寫的特定項目。

主題

- 使用複寫組態會複寫什麼項目？
- 使用複寫組態不會複寫什麼項目？

- 預設儲存貯體加密如何影響複寫

使用複寫組態會複寫什麼項目？

依預設，Amazon S3 會複寫下列項目：

- 在您新增複寫組態之後建立的物件。
- 未加密的物件。
- 使用客戶提供的金鑰 (SSE-C) 加密的物件、Amazon S3 受管金鑰 (SSE-S3) 下的靜態物件加密，或存放於 AWS Key Management Service (SSE-KMS) 的 KMS 金鑰。如需詳細資訊，請參閱 the section called “複寫加密的物件”。
- 從來源物件到複本的物件中繼資料。如需將中繼資料從複本複寫至來源物件的相關資訊，請參閱 使用 Amazon S3 複本修改同步複寫中繼資料變更。
- 僅限來源儲存貯體中的物件 (且儲存貯體擁有者具備讀取物件與讀取存取控制清單 (ACL) 的許可)。

  如需資源擁有權的詳細資訊，請參閱「Amazon S3 儲存貯體和物件擁有權」。
- 除非您指示 Amazon S3 在來源與目的地儲存貯體不屬於相同帳戶時變更複本擁有權，否則物件 ACL 都會更新。

  如需詳細資訊，請參閱「變更複本擁有者」。

  Amazon S3 可能需要一些時間才能讓兩個 ACL 保持同步。此項擁有權變更只適用於將複寫組態新增至儲存貯體之後所建立的物件。
- 物件標籤 (如果有)。
- S3 物件鎖定保留資訊 (如果有)。

  當 Amazon S3 複寫的物件已套用保留資訊時，就會將這些相同的保留控制套用至複本，進而覆寫目的地儲存貯體上設定的預設保留期間。如果您沒有將保留控制套用到來源儲存貯體中的物件，而且您複寫至已設定預設保留期間的目的地儲存貯體，則系統會將目的地儲存貯體預設保留期套用至物件複本。如需詳細資訊，請參閱「使用 S3 物件鎖定」。

刪除操作對複寫的影響

如果您從來源儲存貯體中刪除物件，預設會執行下列動作：

- 如果您發出 DELETE 請求但未指定物件版本 ID，則 Amazon S3 會新增刪除標記。Amazon S3 會如下處理刪除標記：

  - 如果您使用最新版本的複寫組態 (亦即您在複寫組態規則中指定 `Filter` 元素)，根據預設，Amazon S3 就不會複寫刪除標記。但是，您可以將刪除標記複製新增至 non-tag-based 規則。如需詳細資訊，請參閱 複寫儲存貯體之間的刪除標記。
  - 如果您沒有指定 `Filter` 元素，Amazon S3 會假設複寫組態為版本 V1，並複寫使用者動作產生的刪除標記。但是，如果 Amazon S3 因生命週期動作而刪除物件，則刪除標記不會複寫至目的地儲存貯體。
- 如果您在 `DELETE` 請求中指定要刪除的物件版本 ID，Amazon S3 會刪除來源儲存貯體中的該物件版本。但不會在目的地儲存貯體中進行刪除。換句話說，Amazon S3 不會從目的地儲存貯體中刪除相同的物件版本。這可防止資料遭到惡意刪除。

使用複寫組態不會複寫什麼項目？

依預設，Amazon S3 不會複寫下列項目：

- 來源儲存貯體中由其他複寫規則所建立的物件複本。例如，若您設定複寫，其中儲存貯體 A 是來源，而儲存貯體 B 是目的地。現在，假設您新增另一個複寫組態，其中儲存貯體 B 是來源，而儲存貯體 C 是目的地。在此情況下，如果儲存貯體 B 中的物件是儲存貯體 A 中的物件複本，則不會複寫至儲存貯體 C。

  若要複寫做為複本的物件，請使用批次複寫。若要進一步了解設定批次複寫，請參閱 複寫現有物件。
- 來源儲存貯體中已複寫至不同目的地的物件。例如，如果您變更現有複寫組態中的目的地儲存貯體，Amazon S3 就不會再次將它複寫。

  若要複寫以前複寫過的物件，請使用批次複寫。若要進一步了解設定批次複寫，請參閱 複寫現有物件。
- 批次複寫不支援重新複寫具有來自目的地儲存貯體之物件的版本 ID 的已刪除物件。若要重新複寫這些物件，您可以使用批次複製 (Batch Copy) 任務將來源物件複製到位。將這些物件複製到位會在來源儲存貯體中建立物件的新版本，並自動將複寫啟動到目的地。如需有關如何使用批次複製 (Batch Copy) 的詳細資訊，請參閱 使用批次操作複製物件的範例。
- 根據預設，從不同的複製時 AWS 帳戶，不會複寫新增至來源值區的刪除標記。

  如需有關如何複寫刪除標記的資訊，請參閱 複寫儲存貯體之間的刪除標記。
- 存放在 S3 Glacier 彈性擷取、S3 Glacier Deep Archive、S3 智慧型分層封存存取或 S3 智慧型分層深度存檔存取儲存類別或層中的物件。您需先還原這些物件並複製到不同的儲存類別，才能複製這些物件。

若要進一步了解 S3 Glacier 彈性擷取和 S3 Glacier Deep Archive，請參閱很少存取物件的儲存區類別。

若要進一步了解 S3 智慧型分層，請參閱。Amazon S3 Intelligent Tiering

- 來源儲存貯體中，儲存貯體擁有者沒有足夠複寫許可的物件。

  如需物件擁有者如何將許可授予儲存貯體擁有者的資訊，請參閱授予跨帳戶許可，以在確保儲存貯體擁有者具有完全控制時上傳物件。

- 儲存貯體層級子資源的更新。

  例如，如果您變更來源儲存貯體上的生命週期組態，或將通知組態新增至來源儲存貯體，這些變更並不會套用至目的地儲存貯體。此功能可讓來源儲存貯體與目的地儲存貯體的組態不同。

- 生命週期組態執行的動作。

  例如，如果只啟用來源儲存貯體上的生命週期組態，Amazon S3 會建立過期物件的刪除標記，但不會複寫這些標記。如果您想要將相同的生命週期組態套用至來源與目的地儲存貯體，請在這兩個儲存貯體上啟用相同的生命週期組態。如需生命週期組態的詳細資訊，請參閱「管理儲存生命週期」。

- 當您將以標籤為基礎的複寫規則與即時複寫搭配使用時，必須在`PutObject`作業中使用相符的複寫規則標籤來標記新物件。否則，將不會複寫物件。如果物件在`PutObject`作業之後加上標籤，也不會複製這些物件。

  若要複寫`PutObject`作業後已標記的物件，您必須使用 S3 Batch 複寫。如需批次複寫的詳細資訊，請參閱 複寫現有物件。

預設儲存貯體加密如何影響複寫

在您啟用域複寫目的地儲存貯體的預設加密之後，適用下列加密行為：

- 如果未加密來源儲存貯體中的物件，則會使用目的地儲存貯體的預設加密設定來加密目的地儲存貯體中的複本物件。因此，來源物件的實體標籤 (ETag) 與複本物件的 ETag 不同。如果您有使用 ETag 的應用程式，則必須更新這些應用程式以解決此差異。
- 如果來源儲存貯體中的物件使用伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3)、使用 () 金鑰 (SSE-KMS AWS KMS) 進行伺服器端加密，或使用金鑰的雙層伺服器端加密 (DSSE-KMS) 加密，目的地儲存貯體中的複本物件會使用與 AWS KMS 來源物件相同的加密類型。 AWS Key Management Service 不會使用目的地儲存貯體的預設加密設定。

複寫的需求和考量

Amazon S3 複寫需要下列事項：

- 來源值區擁有者必須 AWS 區域 啟用其帳戶的來源和目的地。目的地儲存貯體擁有者必須為該帳戶啟用目的地區域。

  如需啟用或停用的詳細資訊 AWS 區域，請參閱 AWS 區域中的管理AWS 一般參考。
- 來源與目的地儲存貯體都必須啟用版本控制。如需版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。
- Amazon S3 必須具備許可，才能代您將物件從來源儲存貯體複寫至目的地儲存貯體。如需這些許可的詳細資訊，請參閱「設定即時複製的權限」。
- 如果來源儲存貯體的擁有者並未擁有儲存貯體中的物件，則物件擁有者必須先使用物件存取控制清單 (ACL)，將 READ 和 READ_ACP 許可授予儲存貯體擁有者。如需詳細資訊，請參閱「存取控制清單 (ACL) 概觀」。
- 如果來源儲存貯體已啟用 S3 物件鎖定，目的地儲存貯體必須也啟用 S3 物件鎖定。

  若要在已啟用物件鎖定的值區上啟用複寫 AWS Command Line Interface，您必須使用 REST API 或 AWS SDK。如需一般詳細資訊，請參閱使用 S3 物件鎖定。

  > **Note**
  >
  > 您必須在用來設定複寫的 AWS Identity and Access Management (IAM) 角色中，對來源 S3 儲存貯體授予兩個新許可。這兩個新許可為 `s3:GetObjectRetention` 和 `s3:GetObjectLegalHold`。若該角色有 `s3:Get*` 許可，即滿足要求。如需詳細資訊，請參閱 設定即時複製的權限。

如需詳細資訊，請參閱 設定即時複製。

如果您在「跨帳戶案例」中設定複寫組態，其中來源與目的地儲存貯體的擁有者是不同的 AWS 帳戶，則適用下列其他需求：

- 目的地儲存貯體擁有者必須使用儲存貯體原則，來授予來源儲存貯體擁有者複寫物件的許可。如需詳細資訊，請參閱「當來源值區和目的地值區屬於不同時授與權限 AWS 帳戶」。
- 此儲存貯體無法設定為「申請者付款」的儲存貯體。如需詳細資訊，請參閱 使用儲存體傳輸和用量的申請者付款儲存貯體。

複寫的考量

建立複製組態之前，請注意下列考量事項。

主題

生命週期組態與物件複本

Amazon S3 複寫物件所需的時間取決於物件大小。若是大型物件，則可能需要數小時的時間。雖然目的地中的複本在可使用前需要一些時間，但建立複本與在來源儲存貯體中建立相對應物件所需的時間相同。如果目的地儲存貯體上啟用了生命週期組態，生命週期規則即是根據原始物件的建立時間，而不是目的地儲存貯體中的複本成為可用的時間。

複寫組態要求儲存貯體必須已啟用版本控制。當您啟用儲存貯體上的版本控制時，請記住下列事項：

- 如果您擁有物件過期生命週期組態，在您啟用版本控制之後，請新增 `NonCurrentVersionExpiration` 政策，以保持如啟用版本控制之前的相同永久刪除行為。
- 如果您擁有轉移生命週期組態，在您啟用版本控制之後，請考慮新增 `NonCurrentVersionTransition` 政策。

版本控制組態與複寫組態

當您在儲存貯體上設定複寫時，來源與目的地儲存貯體都必須已啟用版本控制。當您同時啟用來源與目的地儲存貯體上的版本控制，並在來源儲存貯體上設定複寫之後，您會遇到下列問題：

- 如果您嘗試停用來源儲存貯體上的版本控制，Amazon S3 會傳回錯誤。您必須移除複寫組態，才能停用來源儲存貯體上的版本控制。
- 如果您停用目的地儲存貯體上的版本控制，複寫會失敗。來源物件的複寫狀態為 `FAILED`。

搭配 S3 Intelligent-Tiering 使用 S3 複寫

S3 Intelligent-Tiering 是一種儲存類別，其設計在於自動將資料移至最具成本效益的存取層，藉此最佳化儲存成本。只需每月支付少許的物件監控和自動化費用，S3 Intelligent-Tiering 即可監控存取模式，並自動將未存取的物件移至低成本層。

使用 S3 批次複寫來複寫儲存在 S3 Intelligent-Tiering 中的物件，或是調用 CopyObject 或 UploadPartCopy，都會構成存取。在這些情況下，複製或複寫操作的來源物件會分層。

如需 S3 Intelligent-Tiering 的詳細資訊，請參閱 Amazon S3 Intelligent Tiering。

記錄日誌組態與複寫組態

如果 Amazon S3 將日誌提供給已啟用複寫的儲存貯體，即會複寫日誌物件。

如果已啟用來源或目的地儲存貯體上的伺服器存取日誌 (使用伺服器存取記錄記錄要求) 或 AWS CloudTrail 日誌 (使用記錄 Amazon S3 API 呼叫 AWS CloudTrail)，Amazon S3 會在日誌中包含複寫相關請求。例如，Amazon S3 會記錄其所複寫的每個物件。

CRR 與目的地區域

Amazon S3 跨區域複寫 (CRR) 可用來跨不同 S3 儲存貯體複製物件。 AWS 區域您可以根據商業需求或成本考量來選擇目的地儲存貯體的區域。例如，區域之間的資料傳輸費會依所選區域而異。

假設您選擇美國東部 (維吉尼亞北部) (us-east-1) 作為您的來源儲存貯體區域。如果您選擇美國西部 (奧勒岡) (us-west-2) 作為目的地儲存貯體區域，您支付的費用會比選擇美國東部 (俄亥俄) (us-east-2) 區域更多。如需定價資訊，請參閱 Amazon S3 定價中的「資料傳輸定價」。

相同區域複寫 (SRR) 沒有相關的資料傳輸費。

S3 Batch 複寫

如需有關 Batch 複寫考量的資訊，請參閱S3 批次複寫注意事項。

S3 複寫時間控制

如需 S3 複寫時間控制 (S3 RTC) 的最佳實務和考量事項的相關資訊，請參閱S3 RTC 的最佳實務和指導方針。

設定即時複製

> **Note**
>
> 您設定複寫之前就已存在的物件不會自動複寫。換句話說，Amazon S3 不會追溯複寫物件。若要複寫在複寫組態之前建立的物件，請使用 S3 批次複寫。若要進一步了解設定批次複寫，請參閱 複寫現有物件。

若要啟用即時複製 (相同區域複寫 (SRR) 或跨區域複寫 (CRR)，請將複寫組態新增至來源儲存貯體。此組態會告訴 Amazon S3 按照指定的方式複寫物件。在複寫組態中，您必須提供以下項目：

- 目的地儲存貯體 – 您希望 Amazon S3 在其中複寫物件的儲存貯體。
- 您要複寫的物件 – 您可以複寫來源儲存貯體中的所有物件或物件子集。您可以在組態中提供金鑰名稱前綴、一或多個物件標籤或兩者，來識別子集。

  例如，如果您將複寫規則設定為僅複寫具有金鑰名稱前綴為 `Tax/` 的物件，則 Amazon S3 會複寫具有 `Tax/doc1` 或 `Tax/doc2` 等金鑰的物件。但不會複寫具有 `Legal/doc3` 金鑰的物件。如果您指定字首以及一或多個標籤，則 Amazon S3 僅會複寫具備特定索引鍵字首和標籤的物件。
- AWS Identity and Access Management (IAM) 角色 — Amazon S3 會假設此 IAM 角色代表您複寫物件。

除了這些最低需求以外，您可以選擇下列選項：

- 複本儲存體方案 – 根據預設，Amazon S3 會使用與來源物件相同的儲存體方案來存放物件複本。您可以為複本指定不同的儲存體方案。
- 複本擁有權 – Amazon S3 會假設物件複本的擁有者持續是來源物件擁有者。所以，當複寫物件時，也會複寫對應的物件存取控制清單 (ACL) 或 S3 物件擁有權設定。如果來源與目的地儲存貯體的擁有者是不同的 AWS 帳戶，您可以設定複寫，以將複本擁有者變更為擁有目標儲存貯體的 AWS 帳戶 。

您可以使用 REST API、 AWS 開發套件 AWS Command Line Interface (AWS CLI) 或 Amazon S3 主控台來設定複寫。

Amazon S3 也提供 API 操作來支援設定複寫規則。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的下列主題：

- PutBucketReplication

- GetBucketReplication
- DeleteBucketReplication

主題

- 複寫組態
- 設定即時複製的權限
- 設定即時複製的範例

複寫組態

Amazon S3 會以 XML 的形式存放複寫組態。在複寫組態 XML 檔案中，您可以指定 AWS Identity and Access Management (IAM) 角色和一或多個規則。

```
<ReplicationConfiguration>
    <Role>IAM-role-ARN</Role>
    <Rule>
        ...
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
</ReplicationConfiguration>
```

Amazon S3 必須先獲得您的許可，才可以複寫物件。您可以使用複寫組態中指定的 IAM 角色來授予許可。Amazon S3 會擔任 IAM 角色以代您複寫物件。您必須先將所需的許可授予 IAM 角色。如需如何管理許可的詳細資訊，請參閱 設定即時複製的權限。

針對下列情況，您可以在複寫組態中新增一個規則：

- 您想要複寫所有物件。
- 您想要複寫物件子集。您在規則中新增篩選條件，以識別物件子集。您可以在篩選條件中指定物件金鑰前綴、標籤，或這兩項的組合，以識別要套用規則的物件子集。篩選條件的目標是確切符合您指定值的物件。

若您想複寫不同的物件子集，可以在複寫組態中新增多項規則。在每個規則中，您可以指定篩選條件以選取不同的物件子集。例如，您可以選擇複寫含有 `tax/` 或 `document/` 索引鍵字首的物件。要做到這

一點，您需要新增兩個規則，一個指定 tax/ 索引鍵字首篩選條件，另一個指定 document/ 索引鍵字首。如需物件金鑰字首的詳細資訊，請參閱 使用字首整理物件。

下列各節提供了額外的資訊。

主題

- 基本規則組態
- 選用：指定篩選條件
- 其他目標組態
- 範例複寫組態
- 回溯相容性

基本規則組態

每個規則都必須包括規則的狀態和優先順序。規則還必須指示是否複寫刪除標記。

- Status 指出是使用 Enabled 還是 Disabled 值來啟用或停用規則。當規則停用時，Amazon S3 即不會執行規則中指定的動作。
- Priority 指出當兩個或多個複寫規則發生衝突時，哪些規則具有優先權。Amazon S3 會嘗試根據所有複寫規則來複寫物件。不過，如果有兩個或多個規則具有相同目的地儲存貯體，則會根據具有最高優先順序的規則來複寫物件。數字愈高，優先順序愈高。
- DeleteMarkerReplication 使用 Enabled 或 Disabled 值，指示是否複寫刪除標記。

在目的地組態中，您必須提供希望 Amazon S3 複寫其中物件的儲存貯體名稱。

以下範例顯示 V2 規則所需的最低需求。為了回溯相容性，Amazon S3 繼續支援 XML V1 格式。如需詳細資訊，請參閱 回溯相容性。

```
...
    <Rule>
        <ID>Rule-1</ID>
        <Status>Enabled-or-Disabled</Status>
        <Filter>
            <Prefix></Prefix>
        </Filter>
        <Priority>integer</Priority>
        <DeleteMarkerReplication>
           <Status>Enabled-or-Disabled</Status>
```

```
        </DeleteMarkerReplication>
        <Destination>
           <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
        </Destination>
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
...
```

您也可以指定其他組態選項。例如，若物件複本的儲存體方案與來源物件的方案不同，您可以選擇為物件複本使用儲存體方案。

選用：指定篩選條件

若要選擇物件子集以套用規則，請新增選用的篩選條件。您可以依物件索引鍵字首、物件標籤或兩者的組合來進行篩選。如果您依索引鍵字首和物件標籤兩者來篩選，Amazon S3 會使用邏輯 AND 運算子來合併篩選條件。換句話說，規則會套用到具有特定金鑰前綴與特定標籤的物件子集。

根據物件金鑰前綴進行篩選

若要指定含物件金鑰前綴篩選條件的規則，請使用下列程式碼。您只能指定一個前綴。

```
<Rule>
    ...
    <Filter>
        <Prefix>key-prefix</Prefix>
    </Filter>
    ...
</Rule>
...
```

根據物件標籤篩選

若要指定含物件標籤篩選條件的規則，請使用下列程式碼。您可以指定一或多個物件標籤。

```
<Rule>
    ...
    <Filter>
        <And>
            <Tag>
```

```
                <Key>key1</Key>
                <Value>value1</Value>
            </Tag>
            <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
            </Tag>
             ...
        </And>
    </Filter>
    ...
</Rule>
...
```

使用金鑰前綴和物件標籤篩選

若要指定含物件金鑰字首和物件標籤組合的規則篩選條件，請使用下列程式碼。您可以將這些篩選條件包裝在 <And> 父元素中。Amazon S3 會執行邏輯 AND 操作來結合這些篩選條件。換句話說，規則會套用到具有特定金鑰前綴與特定標籤的物件子集。

```
<Rule>
    ...
    <Filter>
        <And>
            <Prefix>key-prefix</Prefix>
            <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
            </Tag>
            <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
            </Tag>
             ...
    </Filter>
    ...
</Rule>
...
```

> **Note**
>
> - 如果您使用空白<Filter>元素指定規則，則規則會套用至值區中的所有物件。

- 當您將以標籤為基礎的複寫規則與即時複寫搭配使用時，必須在PutObject作業中使用相符的複寫規則標籤來標記新物件。否則，將不會複寫物件。如果物件在PutObject作業之後加上標籤，也不會複製這些物件。

  若要複寫PutObject作業後已標記的物件，您必須使用 S3 Batch 複寫。如需批次複寫的詳細資訊，請參閱 複寫現有物件。

其他目標組態

在目的地組態中，您可以指定希望 Amazon S3 複寫其中物件的儲存貯體。您可以設定組態以將一個來源儲存貯體中的物件複寫至一個或多個目的地儲存貯體。

```
...
<Destination>
    <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
</Destination>
...
```

您可以在 <Destination> 元素中新增下列選項。

主題

- 指定儲存類別
- 新增多個目的地儲存貯體
- 為具有多個目的地儲存貯體的每個複寫規則指定不同的參數
- 變更複本擁有權
- 啟用 S3 複寫時間控制
- 複製使用伺服器端加密建立的物件 AWS KMS

指定儲存類別

您可以為物件複本指定儲存體方案。根據預設，Amazon S3 會使用來源物件的儲存體方案來建立物件複本，如下列範例所示。

```
...
<Destination>
       <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
       <StorageClass>storage-class</StorageClass>
```

```
</Destination>
...
```

新增多個目的地儲存貯體

您可以在單一複寫組態中新增多個目的地儲存貯體，如下所示。

```
...
<Rule>
    <ID>Rule-1</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled-or-Disabled</Status>
    </DeleteMarkerReplication>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
</Rule>
<Rule>
    <ID>Rule-2</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled-or-Disabled</Status>
    </DeleteMarkerReplication>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET2</Bucket>
    </Destination>
</Rule>
...
```

為具有多個目的地儲存貯體的每個複寫規則指定不同的參數

在單一複寫組態中新增多個目的地儲存貯體時，您可以為每個複寫規則指定不同的參數，如下所示。

```
...
<Rule>
    <ID>Rule-1</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Disabled</Status>
```

```
    </DeleteMarkerReplication>
      <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
</Rule>
<Rule>
    <ID>Rule-2</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled</Status>
    </DeleteMarkerReplication>
      <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
  <ReplicationTime>
    <Status>Enabled</Status>
    <Time>
      <Minutes>15</Minutes>
    </Time>
  </ReplicationTime>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET2</Bucket>
    </Destination>
</Rule>
...
```

變更複本擁有權

當來源和目的地值區不屬於相同帳戶時，您可以將複本的擁有權變更為擁有目的 AWS 帳戶 地值區的擁有權。要做到這一點，請新增 `AccessControlTranslation` 元素。此元素採用 `Destination` 的值。

```
...
```

```
<Destination>
   <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
   <Account>destination-bucket-owner-account-id</Account>
   <AccessControlTranslation>
       <Owner>Destination</Owner>
   </AccessControlTranslation>
</Destination>
...
```

如果您未將AccessControlTranslation元素新增至複寫組態，則複本將由擁有來源物件的複本所擁有。 AWS 帳戶 如需詳細資訊，請參閱 變更複本擁有者。

啟用 S3 複寫時間控制

您可以在複寫組態中啟用 S3 複寫時間控制 (S3 RTC)。S3 RTC 會在數秒內複寫多數物件，以及在 15 分鐘內複寫 99.99% 的物件 (由服務水準協議支援)。

> **Note**
>
> EventThreshold 與 Time 可接受的唯一一個有效值為 <Minutes>15</Minutes>。

```
...
<Destination>
  <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
  <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
  <ReplicationTime>
    <Status>Enabled</Status>
    <Time>
      <Minutes>15</Minutes>
    </Time>
  </ReplicationTime>
</Destination>
...
```

如需詳細資訊，請參閱 使用 S3 複寫時間控制 (S3 RTC) 來達到合規要求。有關 API 示例，請參閱 Amazon 簡單存儲服務 API 參考PutBucketReplication中的。

複製使用伺服器端加密建立的物件 AWS KMS

您的來源儲存貯體可能包含使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 以伺服器端加密建立的物件。依預設，Amazon S3 不會複寫這些物件。您可以選擇性地指示 Amazon S3 複寫這些物件。要做到這一點，請先新增 `SourceSelectionCriteria` 元素，明確選擇使用此功能。然後提供用於加密物件複本 AWS 區域 的 AWS KMS key (用於目的地值區的)。下列範例顯示如何指定這些元素。

```
...
<SourceSelectionCriteria>
  <SseKmsEncryptedObjects>
    <Status>Enabled</Status>
  </SseKmsEncryptedObjects>
</SourceSelectionCriteria>
<Destination>
  <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
  <EncryptionConfiguration>
    <ReplicaKmsKeyID>AWS KMS key ID to use for encrypting object replicas</
ReplicaKmsKeyID>
  </EncryptionConfiguration>
</Destination>
...
```

如需詳細資訊，請參閱 [複寫加密的物件 (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)]。

範例複寫組態

若要開始使用，您可以視需要將下列範例複寫組態新增至您的儲存貯體。

> **Important**
>
> 若要將複寫組態新增至儲存貯體，您必須具有 `iam:PassRole` 許可。這項許可允許您傳遞授予 Amazon S3 複寫許可的 IAM 角色。您可以提供 Amazon Resource Name (ARN) 以指定 IAM 角色，其用於複寫組態 XML 中的 `Role` 元素。如需詳細資訊，請參閱 IAM 使用者指南中[授予使用者將角色傳遞至 AWS 服務的許可]。

Example 1：具有一項規則的複寫組態

下列基本複寫組態可指定一個規則。此規則可指定 Amazon S3 可擔任的 IAM 角色，以及物件複本的單一目的地儲存貯體。`Status` 為 `Enabled` 值表示規則為作用中。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>

    <Destination><Bucket>arn:aws:s3:::example-s3-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

若要選擇物件子集以供複寫，您可以新增篩選條件。在下列組態中，篩選條件指定了一個物件金鑰前綴。此規則會套用至金鑰名稱前綴為 *Tax/* 的物件。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>

    <Filter>
       <Prefix>Tax/</Prefix>
    </Filter>

    <Destination><Bucket>arn:aws:s3:::example-s3-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

如果您指定 Filter 元素，則必須也包含 Priority 與 DeleteMarkerReplication 元素。在此範例中，由於只有一個規則，因此 Priority 並無關聯。

在下列組態中，篩選條件指定了一個前綴和兩個標籤。此規則會套用至具有指定之金鑰前綴和標籤的物件子集。具體來說，它會套用至金鑰名稱有 *Tax/* 具字首和兩個指定物件標籤的物件。由於只有一個規則，因此不會套用 Priority。

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>

    <Filter>
        <And>
          <Prefix>Tax/</Prefix>
          <Tag>
             <Tag>
                <Key>tagA</Key>
                <Value>valueA</Value>
             </Tag>
          </Tag>
          <Tag>
             <Tag>
                <Key>tagB</Key>
                <Value>valueB</Value>
             </Tag>
          </Tag>
       </And>

    </Filter>

    <Destination><Bucket>arn:aws:s3:::example-s3-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

您可以指定物件複本的儲存體方案，如下所示。

```
<?xml version="1.0" encoding="UTF-8"?>

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Destination>
       <Bucket>arn:aws:s3:::example-s3-bucket</Bucket>
```

```
        <StorageClass>storage-class</StorageClass>
      </Destination>
    </Rule>
</ReplicationConfiguration>
```

您可以指定 Amazon S3 支援的任何儲存體方案。

Example 2：具有兩項規則的複寫組態

Example

在下列複寫組態中：

- 每個規則會篩選不同的金鑰前綴，每個規則會套用至不同的物件子集。在此例中，Amazon S3 會複寫金鑰名稱為 *Tax/doc1.pdf* 和 *Project/project1.txt* 的物件，但不會複寫金鑰名稱為 *PersonalDoc/documentA* 的物件。
- 由於規則是套用至兩個不同的物件子集，因此規則優先順序並無關聯。下一個範例說明套用規則優先順序時會發生的情況。
- 第二個規則指定物件複本的 S3 標準 – IA 儲存體方案。Amazon S3 會針對那些物件複本使用指定的儲存體方案。

```
<?xml version="1.0" encoding="UTF-8"?>

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>Tax</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
     ...
  </Rule>
 <Rule>
```

```
    <Status>Enabled</Status>
    <Priority>2</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>Project</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
     <StorageClass>STANDARD_IA</StorageClass>
    </Destination>
     ...
  </Rule>


</ReplicationConfiguration>
```

Example 3：具有兩項前綴重疊之規則的複寫組態

在此組態中，有兩項規則指定了含 *star/* 和 *starship/* 重疊索引鍵字首的篩選條件。這兩個規則都會套用到金鑰名稱為 *starship-x* 的物件。在此情況下，Amazon S3 會使用規則優先順序來判斷要套用哪一個規則。數字愈高，優先順序愈高。

```
<ReplicationConfiguration>

  <Role>arn:aws:iam::account-id:role/role-name</Role>

  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>star</Prefix>
    </Filter>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
  </Rule>
  <Rule>
```

```
    <Status>Enabled</Status>
    <Priority>2</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>starship</Prefix>
    </Filter>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

Example 4：範例演練

如需範例逐步解說，請參閱 設定即時複製的範例。

如需有關複寫組態之 XML 結構的詳細資訊，請參閱 Amazon 簡單儲存服務 API 參考PutBucketReplication中的。

回溯相容性

複寫組態 XML 的最新版本為 V2。XML V2 複寫組態是指包含規則 Filter 元素和指定 S3 複寫時間控制 (S3 RTC) 規則的組態。

若要查看您的複寫組態版本，您可以使用 GetBucketReplication API 操作。如需詳細資訊，請參閱 Amazon 簡易儲存服務 API 參考GetBucketReplication中的。

為了回溯相容性，請 Amazon S3 繼續支援 XML V1 複寫組態。如果您已使用 XML V1 複寫組態，請注意以下會影響回溯相容性的問題：

- 複寫組態 XML V2 包含規則的 Filter 元素。使用 Filter 元素時，您可以依物件金鑰前綴、標籤，或這兩項的組合，指定物件篩選條件，以限定要套用規則的物件範圍。複寫組態 XML V1 支援僅根據金鑰前綴來進行篩選。在這個情況下，您會新增 Prefix，並將其直接作為 Rule 元素的子元素，如下範例所示。

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Role>arn:aws:iam::account-id:role/role-name</Role>
    <Rule>
      <Status>Enabled</Status>
      <Prefix>key-prefix</Prefix>
  ```

```
    <Destination><Bucket>arn:aws:s3:::example-s3-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

Amazon S3 會繼續支援 V1 組態，以提供回溯相容性。

- 當您從來源儲存貯體刪除未指定版本 ID 的物件，Amazon S3 會在此物件上新增刪除標記。如果您使用 V1 的複寫組態 XML，Amazon S3 就會複寫由使用者動作產生的刪除標記。換言之，Amazon S3 僅在使用者刪除物件時才會複寫刪除標記。如果 Amazon S3 移除了過期的物件 (做為生命週期操作的一部分)，Amazon S3 不會複寫刪除標記。

  在 V2 複寫組態中，您可以啟用 non-tag-based 規則的刪除標記複寫。如需詳細資訊，請參閱 複寫儲存貯體之間的刪除標記。

設定即時複製的權限

設定即時複製時，您必須依照下列方式取得必要的權限：

- Amazon S3 需要許可，才能代您複寫物件。您可建立 IAM 角色並在您的複寫組態中指定此角色以授予這些許可。
- 當來源與目的地儲存貯體的擁有者是不同的帳戶時，目的地儲存貯體的擁有者必須將存放複本的許可授予來源儲存貯體擁有者。

主題

- 建立 IAM 角色
- 當來源值區和目的地值區屬於不同時授與權限 AWS 帳戶
- 授予 S3 批次操作的許可
- 變更複本擁有權
- 允許從來源儲存貯體接收複寫的物件

建立 IAM 角色

根據預設，所有 Amazon S3 資源 (儲存貯體、物件與相關子資源) 皆為私有，且只有資源擁有者才可存取該資源。Amazon S3 需要從來源儲存貯體讀取和複寫物件的許可。您可建立 IAM 角色並在您的複寫組態中指定此角色以授予這些許可。

本節說明信任政策與最低必要許可政策。範例逐步解 step-by-step 說提供建立 IAM 角色的指示。如需詳細資訊，請參閱 設定即時複製的範例。

- 以下範例顯示信任政策，您可以在其中將 Amazon S3 識別為可擔任該角色的服務主體。

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

- 以下範例顯示「信任政策」，您可以在其中將 Amazon S3 和 S3 批次操作識別為服務主體。如果您要建立批次複寫任務，此操作很有用。如需詳細資訊，請參閱 為第一個複寫規則或新目的地建立批次複寫任務。

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

如需 IAM 角色的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 角色。

- 以下範例顯示存取政策，您可以在其中授予角色許可來代您執行複寫作業。當 Amazon S3 擔任該角色時，即具備您在此政策中指定的許可。在這項政策中，*example-s3-bucket1* 是來源儲存貯體，*example-s3-bucket2* 是目的地儲存貯體。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::example-s3-bucket1"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3:::example-s3-bucket1/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket2/*"
      }
   ]
}
```

存取政策會授予下列動作的許可：

- `s3:GetReplicationConfiguration` 和 `s3:ListBucket` – *`example-s3-bucket1`* 儲存貯體 (來源儲存貯體) 上這些動作的許可允許 Amazon S3 擷取複寫組態並列出儲存貯體內容。(目前的許可模型需要 `s3:ListBucket` 許可以存取刪除標記。)
- `s3:GetObjectVersionForReplication` 與 `s3:GetObjectVersionAcl` – 所有物件上授予的這些動作許可，其允許 Amazon S3 取得特定物件版本以及與物件相關聯的存取控制清單 (ACL)。
- `s3:ReplicateObject` 與 `s3:ReplicateDelete` – *`example-s3-bucket2`* 儲存貯體 (目的地儲存貯體) 中所有物件上這些動作的許可，其允許 Amazon S3 將物件或刪除標記複寫至目的地儲存貯體。如需刪除標記的資訊，請參閱 刪除操作對複寫的影響。

  > **Note**
  >
  > *`example-s3-bucket2`* 儲存貯體 (目的地儲存貯體) 上 `s3:ReplicateObject` 動作的許可也允許複寫物件標籤和 ACL 等中繼資料。因此，您不需要明確授予 `s3:ReplicateTags` 動作的許可。

- `s3:GetObjectVersionTagging` – *`example-s3-bucket1`* 儲存貯體 (來源儲存貯體) 中物件上這個動作的許可，其允許 Amazon S3 讀取要複寫的物件標籤。如需詳細資訊，請參閱 使用標籤分類儲存空間。如果 Amazon S3 不具備這些許可，則會複寫物件，但不會複寫物件標籤。

如需 Amazon S3 動作的清單，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

> **Important**
>
> 擁 AWS 帳戶 有 IAM 角色的人必須具有授與 IAM 角色之動作的許可。
> 例如，假設來源儲存貯體包含另一個 AWS 帳戶所擁有的物件。物件的擁有者必須透過物件 ACL 明確授與擁有 IAM 角色的必要權限。 AWS 帳戶 否則，Amazon S3 就無法存取這些物件，而導致物件的複寫失敗。如需 ACL 許可的資訊，請參閱「存取控制清單 (ACL) 概觀」。
> 此處描述的許可與基本複寫組態相關。如果您選擇新增額外的複寫組態，則必須將額外許可授予給 Amazon S3。

當來源值區和目的地值區屬於不同時授與權限 AWS 帳戶

當來源與目的地儲存貯體的擁有者是不同的帳戶時，目的地儲存貯體的擁有者也必須新增儲存貯體政策，以將執行複寫動作的許可授予來源儲存貯體擁有者，如下所示。在此政策中，*example-s3-bucket2* 是目的地儲存貯體。

> **Note**
>
> 角色的 ARN 格式可能會有所不同。如果角色是使用主控台建立的，則 ARN 格式為arn:aws:iam::*account-ID*:role/service-role/*role-name*。如果角色是使用建立的 AWS CLI，ARN 格式為arn:aws:iam::*account-ID*:role/*role-name*。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 角色。

```
{
   "Version":"2012-10-17",
   "Id":"PolicyForDestinationBucket",
   "Statement":[
      {
         "Sid":"Permissions on objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":[
            "s3:ReplicateDelete",
            "s3:ReplicateObject"
         ],
         "Resource":"arn:aws:s3:::example-s3-bucket2/*"
      },
      {
         "Sid":"Permissions on bucket",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action": [
            "s3:List*",
            "s3:GetBucketVersioning",
            "s3:PutBucketVersioning"
```

```
         ],
         "Resource":"arn:aws:s3::example-s3-bucket2"
      }
   ]
}
```

如需範例，請參閱當不同帳戶擁有來源與目的地儲存貯體時設定複寫。

如果來源儲存貯體中的物件已標記，請注意下列情況：

- 如果來源儲存貯體擁有者將 s3:GetObjectVersionTagging 與 s3:ReplicateTags 動作的許可授予 Amazon S3 來複寫物件標籤 (透過 IAM 角色)，Amazon S3 會連同物件一起複寫標籤。如需 IAM 角色的資訊，請參閱 建立 IAM 角色。
- 如果目的地儲存貯體擁有者不想要複寫標籤，擁有者可以將下列陳述式新增至目的地儲存貯體政策，以明確拒絕 s3:ReplicateTags 動作的許可。在此政策中，*example-s3-bucket2* 是目的地儲存貯體。

```
...
   "Statement":[
      {
         "Effect":"Deny",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-id:role/service-role/source-account-IAM-role"
         },
         "Action":"s3:ReplicateTags",
         "Resource":"arn:aws:s3:::example-s3-bucket2/*"
      }
   ]
...
```

授予 S3 批次操作的許可

S3 批次複寫讓您能夠複寫在複寫組態就位之前就存在的物件、以前已複寫過的物件以及複寫失敗的物件。在新複寫組態中建立第一項規則或透過 AWS Management Console 將新目的地新增到現有組態時，您可以建立一次性的批次複寫任務。您也可以藉由建立批次操作任務，為現有複寫組態啟動批次複寫。

如需批次複寫 IAM 角色和政策範例，請參閱 設定批次複寫的 IAM 政策。

變更複本擁有權

如果來源和目的地儲存貯體 AWS 帳戶 擁有不同，您可以告訴 Amazon S3 將複本的擁有權變更為擁有目的 AWS 帳戶 地儲存貯體的擁有權。如需擁有者覆寫的詳細資訊，請參閱 變更複本擁有者。

允許從來源儲存貯體接收複寫的物件

您可以透過 AWS Management Console快速產生允許接收來自來源儲存貯體的複寫物件所需的政策。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇您要作為目的地儲存貯體的儲存貯體。
4. 選擇 Management (管理) 標籤，然後向下捲動至 Replication rules (複寫規則)。
5. 針對 Actions (動作)，選擇 Receive replicated objects (接收複寫物件)。

   依照提示輸入來源值區帳戶的 AWS 帳戶 ID，然後選擇「產生政策」。這將產生 Amazon S3 儲存貯體政策和 KMS 金鑰政策。
6. 若要將此政策新增到現有儲存貯體政策，請選擇 Apply settings (套用設定)，或選擇 Copy (複製) 手動複製變更。
7. (選擇性) 將 AWS KMS 原則複製到 AWS Key Management Service 主控台上所需的 KMS 金鑰原則。

設定即時複製的範例

以下範例顯示如何設定常用案例的即時複寫。

> **Note**
>
> 即時複寫是指相同區域複寫 (SRR) 和跨區域複寫 (CRR)。即時複製在您設定複寫之前，不會複製值區中存在的任何物件。若要複製在設定複製之前已存在的物件，請使用隨選複寫。若要視需求同步值區並複製現有物件，請參閱複寫現有物件。

這些範例示範如何使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI) 和 AWS 開發套件建立複寫組態 (以AWS SDK for Java 及示 AWS SDK for .NET 例)。

若要取得有關安裝和配置的資訊 AWS CLI，請參閱《AWS Command Line Interface 使用指南》中的以下主題。

- 安裝 AWS Command Line Interface
- 配置 AWS CLI — 您至少必須設定一個設定檔。如果您要探索跨帳戶案例，請設定兩個描述檔。

如需有關AWS 開 AWS 發套件的詳細資訊，請參閱適用於 .NET 的 Java 和AWS SDK 的開發套件。

> **Tip**
>
> 如需示範如何使用即時複寫來複寫資料的 step-by-step 教學課程，請參閱教學課程： AWS 區域 使用 S3 複寫在內部和之間複寫資料。

主題

- 為相同帳戶擁有的來源和目的地儲存貯體設定複寫
- 當不同帳戶擁有來源與目的地儲存貯體時設定複寫
- 使用 S3 複寫時間控制 (S3 RTC) 來達到合規要求
- 複寫加密的物件 (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)
- 使用 Amazon S3 複本修改同步複寫中繼資料變更
- 複寫儲存貯體之間的刪除標記

為相同帳戶擁有的來源和目的地儲存貯體設定複寫

複製是在相同或不同值區之間對物件進行自動、非同步複製 AWS 區域。複寫會將來源儲存貯體中新建立的物件和物件更新複製至目的地儲存貯體。如需詳細資訊，請參閱 複製物件概觀。

設定複寫時，會將複寫規則新增至來源儲存貯體。複寫規則會定義要複寫的來源儲存貯體物件，以及存放已複寫物件的目的地儲存貯體。您可以建立規則，以特定的金鑰名稱前綴、一或多個物件標籤、或兩種都用，複寫儲存貯體中的所有物件，或一部分的物件。目的地值區可以位於與來源值區 AWS 帳戶相同，也可以位於不同的帳戶中。

如果您指定要刪除的物件版本 ID，Amazon S3 會刪除來源儲存貯體中的該物件版本。但不會在目的地儲存貯體中進行刪除。換句話說，它不會從目的地儲存貯體中刪除相同的物件版本。這可防止資料遭到惡意刪除。

當您將複寫規則新增至儲存貯體時，預設會啟用此規則，讓它在您儲存它之後立即運作。

在此範例中，您會設定來源與目的地儲存貯體為同一 AWS 帳戶所擁有的複寫。提供了使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 和的範例 AWS SDK for .NET。

使用 S3 主控台

若要在目的地值區與來源值區位於相同 AWS 帳戶 時設定複製規則，請依照下列步驟執行。

如果目的地儲存貯體位在與來源儲存貯體不同的帳戶中，您必須將儲存貯體政策新增至目的地儲存貯體，以將複寫目的地儲存貯體中物件的許可授予來源儲存貯體帳戶擁有者。如需詳細資訊，請參閱 當來源值區和目的地值區屬於不同時授與權限 AWS 帳戶。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇您所需的儲存貯體名稱。
4. 選擇管理標籤，向下捲動至複寫規則，然後選擇建立複寫規則。
5. 在複寫規則設定區段的複寫規則名稱下，輸入規則名稱，以便之後識別規則。此名稱為必要，且在儲存貯體內必須是唯一的。
6. 在 Status (狀態) 下，預設會選取 Enabled (已啟用)。已啟用規則在您儲存它之後就會立即運作。如果希望稍後再啟用此規則，請選擇已停用。
7. 如果儲存貯體有現存的複寫規則，系統會指示您設定規則的優先順序。您必須設定規則優先順序，以免多項規則範圍內的物件引發衝突。如果發生規則重疊的情況，Amazon S3 會使用規則優先順序來判斷要套用哪一個規則。數字愈高，優先順序愈高。如需有關規則優先順序的詳細資訊，請參閱 複寫組態。
8. 在來源儲存貯體之下，您有下列選項可用來設定複寫來源：
   - 若要複寫整個儲存貯體，請選擇 Apply to all objects in the bucket (套用至儲存貯體中的所有物件)。
   - 若要複寫具有相同字首的所有物件，請選擇Limit the scope of this rule using one or more filters (使用一或多個篩選器限制此規則的範圍)。這會將複寫限制為具有以您指定字首開頭的所有物件 (例如 `pictures`) 名稱。在字首方塊中輸入字首。

> **Note**
>
> 如果您輸入的字首是資料夾名稱，最後一個字元必須使用 / (正斜線) (例如，`pictures/`)。

- 若要複寫具有一個或多個物件標籤的所有物件，請選擇新增標籤，然後在方塊中輸入關鍵值比對。重複此過程，添加另一個標籤。您可以合併字首與標籤。如需物件標籤的詳細資訊，請參閱使用標籤分類儲存空間。

新的複寫組態 XML 結構描述支援字首和標籤篩選條件以及規則的優先順序。如需有關新結構描述的資訊，請參閱 回溯相容性。如需與在使用者介面後工作之 Amazon S3 API 搭配使用的 XML 的相關資訊，請參閱 複寫組態。新的結構描述會被描述為複寫組態 XML V2。

9. 在目的地下，選擇要 Amazon S3 複寫物件的儲存貯體。

> **Note**
>
> 目的地儲存貯體的數目限制為指定分割區 AWS 區域 中的數目。分割區是區域的群組。AWS 目前有三個分區：`aws`(標準區域)、`aws-cn` (中國地區) 和 `aws-us-gov` (AWS GovCloud (US) 區域)。您可以使用服務配額來要求增加目的地儲存貯體配額。

- 若要複寫到帳戶中的儲存貯體，請選擇選擇此帳戶中的儲存貯體，然後輸入或瀏覽目的地儲存貯體名稱。
- 若要複製到另一個或多個值區中的值區 AWS 帳戶，請選擇「在其他帳戶中指定值區」，然後輸入目的地時段科目 ID 與值區名稱。

  如果目的地儲存貯體位在與來源儲存貯體不同的帳戶中，您必須將儲存貯體原則新增至目的地儲存貯體，以將複寫物件的許可授予來源儲存貯體帳戶擁有者。如需詳細資訊，請參閱 當來源值區和目的地值區屬於不同時授與權限 AWS 帳戶。

  或者，如果您想要啟用協助標準化目的地儲存貯體中新物件的擁有權，請選擇將物件擁有權變更為目的地儲存貯體擁有者。如需有關此選項的詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

> **Note**
>
> 如果未在目的地儲存貯體上啟用版本控制，您會收到包含啟用版本控制按鈕的警告訊息。選擇此按鈕，以在儲存貯體上啟用版本控制。

10. 設定 Amazon S3 可假設為代表您複寫物件的 AWS Identity and Access Management (IAM) 角色。

    若要設定 IAM 角色，請在 IAM 角色區段中，從 IAM 角色下拉式清單選取下列其中一項：

    - 強烈建議您選擇建立新角色，讓 Amazon S3 為您建立新 IAM 角色。當您儲存規則時，系統會為符合所選擇來源與目的地儲存貯體的 IAM 角色產生新原則。
    - 您可以選擇使用現有 IAM 角色。如果這麼做，則必須選擇將必要複寫許可授予 Amazon S3 的角色。如果此角色未將遵循您複寫規則的足夠許可授予 Amazon S3，則複寫會失敗。

    > **Important**
    >
    > 當您新增複寫規則至儲存貯體時，必須擁有 `iam:PassRole` 許可，方能透過 IAM 角色授予 Amazon S3 複寫許可。如需詳細資訊，請參閱《IAM 使用者指南》中授予使用者將角色傳遞至 AWS 服務的許可。

11. 若要複製使用 () 金鑰 AWS Key Management Service (SSE-KMS AWS KMS) 進行伺服器端加密的來源儲存貯體中的物件，請在「加密」下選取「複製使用加密的物件」。 AWS KMS在用於加密目的地物件的AWS KMS 金鑰下，是您允許複寫使用的來源金鑰。根據預設，會包含所有來源 KMS 金鑰。您可以選擇別名或 ID，縮小 KMS 金鑰的選取範圍。

    不會複製由 AWS KMS keys 您未選取的加密物件。系統會為您選擇一個 KMS 金鑰或一組 KMS 金鑰，但您也可以自行選擇 KMS 金鑰。如需 AWS KMS 與複製搭配使用的資訊，請參閱複寫加密的物件 (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)。

    > **Important**
    >
    > 當您複寫使用加密的物件時 AWS KMS，來源區域中的 AWS KMS 要求率會加倍，而且目的地區域的要求率會增加相同的數量。這些增加的通話費率 AWS KMS 是因為使用您為複寫目的地區域定義的 KMS 金鑰重新加密資料的方式所致。 AWS KMS 具有每個區域

每個呼叫帳戶的請求率配額。如需配額預設值的資訊，請參閱《AWS Key Management Service 開發人員指南》中的 AWS KMS 配額 – 每秒請求數：各有不同。
如果複寫期間目前的 Amazon S3 PUT 物件請求率超過帳戶預設 AWS KMS 速率限制的一半以上，建議您請求提高 AWS KMS 請求率配額。若要請求提高，請在 AWS Support 中心的聯絡我們中內建立案例。例如，假設您目前的PUT物件要求率為每秒 1,000 個要求，而您用 AWS KMS 來加密物件。在這種情況下，我們建議您要求 AWS Support 將來源區域和目的地區域 (如果不同) 的 AWS KMS 速率限制提高到每秒 2,500 個請求，以確保沒有限制。 AWS KMS
若要在來源儲存貯體中查看PUT物件請求率，請`PutRequests`在 Amazon Amazon S3 的 Amazon CloudWatch 請求指標中查看。如需檢視 CloudWatch 測量結果的資訊，請參閱使用 S3 主控台。

如果您選擇複製使用加密的物件 AWS KMS，請執行下列動作：

- 在用於加密目的地物件的AWS KMS key 下，使用下列其中一種方式指定 KMS 金鑰：
  - 若要從可用的 KMS 金鑰清單中選擇，請選擇從 AWS KMS keys中選擇，然後從可用金鑰清單中選擇您的 KMS 金鑰。

    AWS 受管金鑰 (`aws/s3`) 和您的客戶管理金鑰都會出現在此清單中。如需詳細了解客戶受管金鑰，請參閱《AWS Key Management Service 開發人員指南》中的客戶金鑰和 AWS 金鑰。
  - 若要輸入 KMS 金鑰 Amazon Resource Name (ARN)，請選擇輸入 AWS KMS key ARN，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。此動作會加密目的地儲存貯體中的複本。您可以在 IAM 主控台的 [加密金鑰] 底下找到 KMS 金鑰的 ARN。
  - 若要在 AWS KMS 主控台中建立新的客戶管理金鑰，請選擇 [建立 KMS 金鑰]。

    如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。

> **Important**
>
> 您只能使用與儲存貯體相 AWS 區域 同的 KMS 金鑰。如果選擇的是從您的 KMS 金鑰選擇，則 S3 主控台只會列出每個區域的其中 100 個 KMS 金鑰。如果您在相同區域中有超過 100 個 KMS 金鑰，您只能看到 S3 主控台中的前 100 個 KMS 金鑰。若

要使用主控台中未列出的 KMS 金鑰，請選擇輸入 AWS KMS key ARN，然後輸入 KMS 金鑰 ARN。
當您在 Amazon S3 中使 AWS KMS key 用伺服器端加密時，您必須選擇對稱加密 KMS 金鑰。Amazon S3 僅支援對稱加密 KMS 金鑰，而不支援非對稱 KMS 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的識別對稱和非對稱 KMS 金鑰。

如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。如需 AWS KMS 搭配 Amazon S3 搭配使用的詳細資訊，請參閱使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)。

12. 在目的地儲存方案之下,若要將資料複寫至目的地中的特定儲存方案，請選取變更已複寫物件的儲存方案。然後選擇您要用於目的地中已複寫物件的儲存體方案。如果您未選擇此選項，則已複寫物件的儲存體方案會與原始物件的類別相同。
13. 設定其他複寫選項時，您具有下列其他選項：

    - 如果您想要在複製組態中啟用 S3 複寫時間控制 (S3 RTC)，請選取 複製時間控制。如需有關此選項的詳細資訊，請參閱 使用 S3 複寫時間控制 (S3 RTC) 來達到合規要求。
    - 如果您想要在複寫組態中啟用 S3 複寫指標，請選取 Replication metrics and events (複寫指標和事件)。如需詳細資訊，請參閱 使用複寫指標和 S3 事件通知監控進度。
    - 如果您想要在複寫組態中啟用刪除標記複寫，請選取 Delete marker replication (刪除標記複寫)。如需詳細資訊，請參閱 複寫儲存貯體之間的刪除標記。
    - 如果您想要在複寫組態中啟用 Amazon S3 複本修改同步，請選取 Replica modification sync (複本修改同步)。如需詳細資訊，請參閱 使用 Amazon S3 複本修改同步複寫中繼資料變更。

    > **Note**
    >
    > 當您使用 S3 RTC 或 S3 複寫指標時，需支付額外費用。

14. 若要完成，請選擇 Save (儲存)。
15. 儲存規則之後，您可以選取規則並選擇編輯規則來編輯、啟用、停用或刪除規則。

使用 AWS CLI

若要在 AWS CLI 來源值區和目的地儲存貯體擁有時使用來設定複製 AWS 帳戶，請執行下列操作：

- 建立來源與目的地儲存貯體
- 在儲存貯體啟用版本控制
- 建立 IAM 角色，授予 Amazon S3 複寫物件的許可。
- 將複寫組態新增至來源儲存貯體

您可以測試以驗證設定。

若要在來源和目的地儲存貯體擁有相同值區時設定複製 AWS 帳戶

1. 設定 AWS CLI的憑證描述檔。在此範例中，我們使用描述檔名稱 `acctA`。如需設定憑證描述檔的相關資訊，請參閱《AWS Command Line Interface 使用者指南》中的具名描述檔。

   > **Important**
   >
   > 用於此練習的描述檔必須有必要的許可。例如，您可以在複寫組態中指定 Amazon S3 可以擔任的 IAM 角色。只有當您所用的描述檔有 `iam:PassRole` 許可時才可執行此作業。如需詳細資訊，請參閱《IAM 使用者指南》中的授予使用者將角色傳遞至 AWS 服務的許可。如果您使用管理員憑證建立具名描述檔，即可執行所有任務。

2. 建立 *`source`* 儲存貯體並對它啟用版本控制。下列程式碼會在美國東部 (維吉尼亞北部) (us-east-1) 區域中建立 *`source`* 儲存貯體。

   ```
   aws s3api create-bucket \
   --bucket source \
   --region us-east-1 \
   --profile acctA
   ```

   ```
   aws s3api put-bucket-versioning \
   --bucket source \
   --versioning-configuration Status=Enabled \
   --profile acctA
   ```

3. 建立 *`destination`* 儲存貯體並對它啟用版本控制。下列程式碼會在美國西部 (奧勒岡) (us-west-2) 區域中建立 *`destination`* 儲存貯體。

> **Note**
>
> 若要在來源儲存貯體和目的地儲存貯體位於相同時設定複製組態 AWS 帳戶，請使用相同的設定檔。此範例使用 `acctA`。若要在不同值區擁有時測試複製組態 AWS 帳戶，請為每個值區指定不同的設定檔。此範例的目標儲存貯體使用 `acctB` 描述檔。

```
aws s3api create-bucket \
--bucket destination \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket destination \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. 建立 IAM 角色。您可以在複寫組態中指定稍後要新增至 *source* 儲存貯體的角色，Amazon S3 就會擔任此角色以代您複寫物件。建立 IAM 角色需要兩個步驟：

   - 建立角色。
   - 將許可政策連接到角色。

   a. 建立 IAM 角色。

      i. 複製下列信任政策，並將它儲存至本機電腦目前目錄下名稱為 `s3-role-trust-policy.json` 的檔案中。此政策會授予 Amazon S3 服務主體擔任該角色的許可。

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

```
      }
   ]
}
```

ii. 執行下列命令以建立角色。

```
$ aws iam create-role \
--role-name replicationRole \
--assume-role-policy-document file://s3-role-trust-policy.json  \
--profile acctA
```

b. 將許可政策連接到角色。

i. 複製下列許可政策，並將它儲存至本機電腦目前目錄中名為 s3-role-permissions-policy.json 的檔案。此政策會授予各種 Amazon S3 儲存貯體與物件動作的許可。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3:::source-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
```

```
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
        ],
        "Resource":"arn:aws:s3:::destination-bucket/*"
      }
   ]
}
```

ii. 執行下列命令以建立政策，並將它連接至角色。

```
$ aws iam put-role-policy \
--role-name replicationRole \
--policy-document file://s3-role-permissions-policy.json \
--policy-name replicationRolePolicy \
--profile acctA
```

5. 將複寫組態新增至 *source* 儲存貯體。

a. 雖然 Amazon S3 API 需要以 XML 格式進行複寫組態，但您 AWS CLI 必須將複寫組態指定為 JSON。將下列 JSON 儲存至您電腦本機目錄下的 `replication.json` 檔案中。

```
{
  "Role": "IAM-role-ARN",
  "Rules": [
    {
      "Status": "Enabled",
      "Priority": 1,
      "DeleteMarkerReplication": { "Status": "Disabled" },
      "Filter" : { "Prefix": "Tax"},
      "Destination": {
        "Bucket": "arn:aws:s3:::destination-bucket"
      }
    }
  ]
}
```

b. 提供 *destination-bucket* 和 *IAM-role-ARN* 的值以更新 JSON。儲存變更。

c. 執行下列命令，將複寫組態新增至您的來源儲存貯體。請務必提供 *source* 儲存貯體名稱。

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket source \
```

```
--profile acctA
```

使用 `get-bucket-replication` 命令來擷取複寫組態。

```
$ aws s3api get-bucket-replication \
--bucket source \
--profile acctA
```

6. 在 Amazon S3 主控台中測試設定：

   a. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

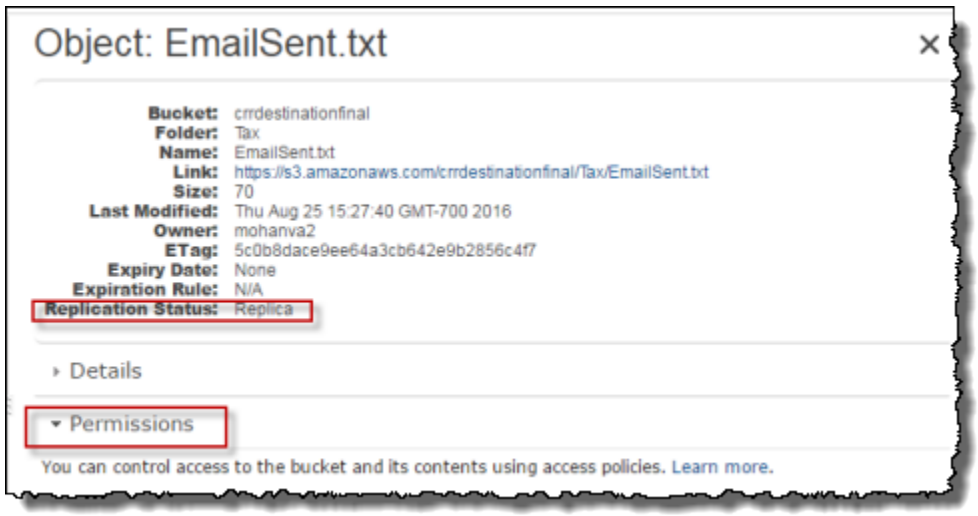
   b. 在 *source* 儲存貯體中，建立名為 `Tax` 的資料夾。

   c. 將範例物件新增至 *source* 儲存貯體的 `Tax` 資料夾。

   > **Note**
   >
   > Amazon S3 複寫物件所需的時間長短取決於物件大小。如需如何查看複寫狀態的相關資訊，請參閱 取得複寫狀態資訊。

   在 *destination* 儲存貯體中驗證下列事項：

   - Amazon S3 已複寫物件。
   - 在物件 properties (屬性) 中，Replication Status (複寫狀態) 已設定為 `Replica` (將此識別為複本物件)。
   - 在物件 properties (屬性) 中，permission (許可) 區段顯示無許可。這表示複本仍為 *source* 儲存貯體的擁有者所有，而 *destination* 儲存貯體擁有者沒有物件複本的任何許可。您可以新增選用組態以指示 Amazon S3 變更複本擁有權。如需範例，請參閱如何變更複本擁有者。



使用 AWS 軟體開發套件

使用下列程式碼範例 AWS SDK for .NET，將複寫組態分別新增至含有 AWS SDK for Java 和的值區。

Java

下列範例會將複寫組態新增至儲存貯體，然後擷取它，並驗證組態。如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import
 com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreateRoleRequest;
import com.amazonaws.services.identitymanagement.model.PutRolePolicyRequest;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.BucketReplicationConfiguration;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.CreateBucketRequest;
import com.amazonaws.services.s3.model.DeleteMarkerReplication;
import com.amazonaws.services.s3.model.DeleteMarkerReplicationStatus;
import com.amazonaws.services.s3.model.ReplicationDestinationConfig;
```

```
import com.amazonaws.services.s3.model.ReplicationRule;
import com.amazonaws.services.s3.model.ReplicationRuleStatus;
import com.amazonaws.services.s3.model.SetBucketVersioningConfigurationRequest;
import com.amazonaws.services.s3.model.StorageClass;
import com.amazonaws.services.s3.model.replication.ReplicationFilter;
import com.amazonaws.services.s3.model.replication.ReplicationFilterPredicate;
import com.amazonaws.services.s3.model.replication.ReplicationPrefixPredicate;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class CrossRegionReplication {

        public static void main(String[] args) throws IOException {
                Regions clientRegion = Regions.DEFAULT_REGION;
                String accountId = "*** Account ID ***";
                String roleName = "*** Role name ***";
                String sourceBucketName = "*** Source bucket name ***";
                String destBucketName = "*** Destination bucket name ***";
                String prefix = "Tax/";

                String roleARN = String.format("arn:aws:iam::%s:%s", accountId,
 roleName);
                String destinationBucketARN = "arn:aws:s3:::" + destBucketName;

                AmazonS3 s3Client = AmazonS3Client.builder()
                                .withCredentials(new ProfileCredentialsProvider())
                                .withRegion(clientRegion)
                                .build();

                createBucket(s3Client, clientRegion, sourceBucketName);
                createBucket(s3Client, clientRegion, destBucketName);
                assignRole(roleName, clientRegion, sourceBucketName,
 destBucketName);

                try {

                        // Create the replication rule.
                        List<ReplicationFilterPredicate> andOperands = new
 ArrayList<ReplicationFilterPredicate>();
                        andOperands.add(new ReplicationPrefixPredicate(prefix));
```

```
                        Map<String, ReplicationRule> replicationRules = new
HashMap<String, ReplicationRule>();
                        replicationRules.put("ReplicationRule1",
                                        new ReplicationRule()
                                                        .withPriority(0)

.withStatus(ReplicationRuleStatus.Enabled)

.withDeleteMarkerReplication(
                                                                        new
DeleteMarkerReplication().withStatus(

   DeleteMarkerReplicationStatus.DISABLED))
                                                        .withFilter(new
ReplicationFilter().withPredicate(
                                                                        new
ReplicationPrefixPredicate(prefix)))
                                                        .withDestinationConfig(new
ReplicationDestinationConfig()

.withBucketARN(destinationBucketARN)

.withStorageClass(StorageClass.Standard)));

                        // Save the replication rule to the source bucket.
                        s3Client.setBucketReplicationConfiguration(sourceBucketName,
                                        new BucketReplicationConfiguration()
                                                        .withRoleARN(roleARN)

.withRules(replicationRules));

                        // Retrieve the replication configuration and verify that
the configuration
                        // matches the rule we just set.
                        BucketReplicationConfiguration replicationConfig = s3Client

.getBucketReplicationConfiguration(sourceBucketName);
                        ReplicationRule rule =
replicationConfig.getRule("ReplicationRule1");
                        System.out.println("Retrieved destination bucket ARN: "
                                        +
rule.getDestinationConfig().getBucketARN());
```

```
                        System.out.println("Retrieved priority: " +
rule.getPriority());
                        System.out.println("Retrieved source-bucket replication rule
status: " + rule.getStatus());
                } catch (AmazonServiceException e) {
                        // The call was transmitted successfully, but Amazon S3
couldn't process
                        // it, so it returned an error response.
                        e.printStackTrace();
                } catch (SdkClientException e) {
                        // Amazon S3 couldn't be contacted for a response, or the
client
                        // couldn't parse the response from Amazon S3.
                        e.printStackTrace();
                }
        }

        private static void createBucket(AmazonS3 s3Client, Regions region, String
bucketName) {
                CreateBucketRequest request = new CreateBucketRequest(bucketName,
region.getName());
                s3Client.createBucket(request);
                BucketVersioningConfiguration configuration = new
BucketVersioningConfiguration()
                                .withStatus(BucketVersioningConfiguration.ENABLED);

                SetBucketVersioningConfigurationRequest enableVersioningRequest =
new SetBucketVersioningConfigurationRequest(
                                bucketName, configuration);
                s3Client.setBucketVersioningConfiguration(enableVersioningRequest);

        }

        private static void assignRole(String roleName, Regions region, String
sourceBucket, String destinationBucket) {
                AmazonIdentityManagement iamClient =
AmazonIdentityManagementClientBuilder.standard()
                                .withRegion(region)
                                .withCredentials(new ProfileCredentialsProvider())
                                .build();
                StringBuilder trustPolicy = new StringBuilder();
                trustPolicy.append("{\\r\\n   ");
                trustPolicy.append("\\\"Version\\\":\\\"2012-10-17\\\",\\r\\n   ");
```

```
                trustPolicy.append("\\\"Statement\\\":[\\r\\n      {\\r\\n
 ");
                trustPolicy.append("\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\
\"Principal\\\":{\\r\\n            ");
                trustPolicy.append("\\\"Service\\\":\\\"s3.amazonaws.com\\\"\\r\\n
      },\\r\\n         ");
                trustPolicy.append("\\\"Action\\\":\\\"sts:AssumeRole\\\"\\r\\n
  }\\r\\n   ]\\r\\n}");

                CreateRoleRequest createRoleRequest = new CreateRoleRequest()
                                .withRoleName(roleName)

 .withAssumeRolePolicyDocument(trustPolicy.toString());

                iamClient.createRole(createRoleRequest);

                StringBuilder permissionPolicy = new StringBuilder();
                permissionPolicy.append(
                                "{\\r\\n   \\\"Version\\\":\\\"2012-10-17\\\",\\r\\n
   \\\"Statement\\\":[\\r\\n      {\\r\\n         ");
                permissionPolicy.append(
                                "\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\
\"Action\\\":[\\r\\n             ");
                permissionPolicy.append("\\\"s3:GetObjectVersionForReplication\\\",\
\r\\n            ");
                permissionPolicy.append(
                                "\\\"s3:GetObjectVersionAcl\\\"\\r\\n         ],\\r\
\n         \\\"Resource\\\":[\\r\\n            ");
                permissionPolicy.append("\\\"arn:aws:s3:::");
                permissionPolicy.append(sourceBucket);
                permissionPolicy.append("/*\\\"\\r\\n         ]\\r\\n      },\\r\\n
     {\\r\\n         ");
                permissionPolicy.append(
                                "\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\
\"Action\\\":[\\r\\n            ");
                permissionPolicy.append(
                                "\\\"s3:ListBucket\\\",\\r\\n            \\
\"s3:GetReplicationConfiguration\\\"\\r\\n         ");
                permissionPolicy.append("],\\r\\n         \\\"Resource\\\":[\\r\\n
          \\\"arn:aws:s3:::");
                permissionPolicy.append(sourceBucket);
                permissionPolicy.append("\\r\\n         ");
                permissionPolicy
```

```
                                .append("]\\r\\n      },\\r\\n      {\\r\\n
 \\\"Effect\\\":\\\"Allow\\\",\\r\\n         ");
                permissionPolicy.append(
                                "\\\"Action\\\":[\\r\\n            \\
\"s3:ReplicateObject\\\",\\r\\n            ");
                permissionPolicy
                                .append("\\\"s3:ReplicateDelete\\\",\\r\\n
  \\\"s3:ReplicateTags\\\",\\r\\n            ");
                permissionPolicy.append("\\\"s3:GetObjectVersionTagging\\\"\\r\\n\\r
\\n         ],\\r\\n         ");
                permissionPolicy.append("\\\"Resource\\\":\\\"arn:aws:s3:::");
                permissionPolicy.append(destinationBucket);
                permissionPolicy.append("/*\\\"\\r\\n      }\\r\\n   ]\\r\\n}");

                PutRolePolicyRequest putRolePolicyRequest = new
 PutRolePolicyRequest()
                                .withRoleName(roleName)
                                .withPolicyDocument(permissionPolicy.toString())
                                .withPolicyName("crrRolePolicy");

                iamClient.putRolePolicy(putRolePolicyRequest);

        }
}
```

C#

下列 AWS SDK for .NET 程式碼範例會將複寫組態新增至值區，然後擷取它。若要使用此程式碼，請提供儲存貯體的名稱和 IAM 角色的 Amazon Resource Name (ARN)。如需有關設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CrossRegionReplicationTest
    {
        private const string sourceBucket = "*** source bucket ***";
```

```
        // Bucket ARN example - arn:aws:s3:::destinationbucket
        private const string destinationBucketArn = "*** destination bucket ARN
***";
        private const string roleArn = "*** IAM Role ARN ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint sourceBucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(sourceBucketRegion);
            EnableReplicationAsync().Wait();
        }
        static async Task EnableReplicationAsync()
        {
            try
            {
                ReplicationConfiguration replConfig = new ReplicationConfiguration
                {
                    Role = roleArn,
                    Rules =
                        {
                            new ReplicationRule
                            {
                                Prefix = "Tax",
                                Status = ReplicationRuleStatus.Enabled,
                                Destination = new ReplicationDestination
                                {
                                    BucketArn = destinationBucketArn
                                }
                            }
                        }
                };

                PutBucketReplicationRequest putRequest = new
PutBucketReplicationRequest
                {
                    BucketName = sourceBucket,
                    Configuration = replConfig
                };

                PutBucketReplicationResponse putResponse = await
s3Client.PutBucketReplicationAsync(putRequest);
```

```
                // Verify configuration by retrieving it.
                await RetrieveReplicationConfigurationAsync(s3Client);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
        private static async Task RetrieveReplicationConfigurationAsync(IAmazonS3
 client)
        {
            // Retrieve the configuration.
            GetBucketReplicationRequest getRequest = new GetBucketReplicationRequest
            {
                BucketName = sourceBucket
            };
            GetBucketReplicationResponse getResponse = await
 client.GetBucketReplicationAsync(getRequest);
            // Print.
            Console.WriteLine("Printing replication configuration information...");
            Console.WriteLine("Role ARN: {0}", getResponse.Configuration.Role);
            foreach (var rule in getResponse.Configuration.Rules)
            {
                Console.WriteLine("ID: {0}", rule.Id);
                Console.WriteLine("Prefix: {0}", rule.Prefix);
                Console.WriteLine("Status: {0}", rule.Status);
            }
        }
    }
}
```

當不同帳戶擁有來源與目的地儲存貯體時設定複寫

當##和###儲存貯體擁有不同時設定複寫，類似 AWS 帳戶 於在兩個值區都屬於同一帳戶時設定複寫。唯一的區別是 *destination* 儲存貯體擁有者必須透過新增儲存貯體政策，授予 *source* 儲存貯體擁有者複寫物件的許可。

如需有關在跨帳戶案例中藉由 AWS Key Management Service 使用伺服器端加密設定複寫的詳細資訊，請參閱 跨帳戶案例之授予其他許可。

在來源和目的地儲存貯體為不同時設定複製 AWS 帳戶

1. 在此範例中，您會建立兩個不同的##值區和#的地值區 AWS 帳戶。您需要為 AWS CLI (在此範例中，我們使用acctA和acctB設定檔名稱) 設定兩個認證設定檔。如需設定憑證描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的具名描述檔。
2. 請按照中的 step-by-step 說明設定相同帳戶內的儲存貯體進行以下變更：
   - 對於與#存儲桶活動相關的所有 AWS CLI 命令（用於創建#存儲桶，啟用版本控制以及創建 IAM 角色），請使用acctA配置文件。使用 acctB 描述檔建立 *destination* 儲存貯體。
   - 請確定許可政策指定您為此範例建立的 *source* 和 *destination* 儲存貯體。
3. 在主控台中，新增 *destination* 儲存貯體上的下列儲存貯體政策，來允許 *source* 儲存貯體擁有者複寫物件。請務必提供來#儲存貯體擁有者的 AWS 帳戶 ID 和#的地值區名稱來編輯政策。

> **Note**
>
> 若要使用下列範例，請以您自己的資訊取代 *user input placeholders*。將 *DOC-EXAMPLE-BUCKET* 取代為您的目的地儲存貯體名稱。將 *source-bucket-acct-ID###/####/## ACT-IAM-####*為您用於此複製組態的角色。
> 如果您手動建立了 IAM 服務角色，請將角色路徑設定為 role/service-role/，如下列政策範例所示。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM ARN。

```
{
   "Version":"2012-10-17",
   "Id":"",
   "Statement":[
      {
         "Sid":"Set-permissions-for-objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-acct-ID:role/service-role/source-acct-IAM-role"
         },
         "Action":["s3:ReplicateObject", "s3:ReplicateDelete"],
         "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      },
```

```
      {
         "Sid":"Set permissions on bucket",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-acct-ID:role/service-role/source-
acct-IAM-role"
         },
         "Action":["s3:GetBucketVersioning", "s3:PutBucketVersioning"],
         "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET"
      }
   ]
}
```

選擇儲存貯體及新增儲存貯體政策。如需說明，請參閱使用 Amazon S3 主控台新增儲存貯體政策。

在複寫中，來源物件的擁有者依預設還擁有複本。當來源和目的地儲存貯體擁有不同時 AWS 帳戶，您可以新增選擇性組態設定，將複本擁有權變更為擁有目 AWS 帳戶 的地值區的擁有權。這包含授予 `ObjectOwnerOverrideToBucketOwner` 許可。如需詳細資訊，請參閱 變更複本擁有者。

變更複本擁有者

在複寫中，來源物件的擁有者依預設也擁有複本。當來源和目的地值區屬於不同的值區， AWS 帳戶 而您想要將複本擁有權變更為擁有目的 AWS 帳戶 地值區的擁有權時，您可以新增選擇性組態設定，將複本擁有權變更為擁有目 AWS 帳戶 的地值區的擁有權。例如，您可以這樣做來限制物件複本的存取。這稱為複寫組態的擁有者覆寫選項。如需擁有者覆寫選項的詳細資訊，請參閱 將擁有者覆寫選項新增至複寫組態。如需設定複寫組態的相關資訊，請參閱 複製物件概觀。

若要設定擁有者覆寫，您可以執行下列動作：

- 將擁有者覆寫選項新增至複寫組態，以通知 Amazon S3 變更複本所有權。
- 授予 Amazon S3 變更複本擁有權的許可。
- 在目的地儲存貯體原則中新增許可，以允許變更複本擁有權。這可讓目的地儲存貯體的擁有者接受物件複本的擁有權。

如需詳細資訊，請參閱 將擁有者覆寫選項新增至複寫組態。如需包含 step-by-step 指示的工作範例，請參閱如何變更複本擁有者。

「物件擁有權」的儲存貯體擁有者強制執行設定

當您使用 Amazon S3 複寫且來源和目的地儲存貯體屬於不同時 AWS 帳戶，目的地儲存貯體的擁有者可以停用 ACL (使用儲存貯體擁有者強制執行物件擁有權設定)，將複本擁有權變更為擁有目的地儲存貯體 AWS 帳戶 的擁有權。此設定會模擬現有的擁有者覆寫行為，而無需 `s3:ObjectOwnerOverrideToBucketOwner` 許可。這意味著使用儲存貯體擁有者強制執行設定複寫至目的地儲存貯體的所有物件都由目的地儲存貯體擁有者所擁有。如需「物件擁有權」的詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

將擁有者覆寫選項新增至複寫組態

> **Warning**
>
> 只有在來源值區和目的地值區為不同時，才新增擁有者覆寫選項 AWS 帳戶。Amazon S3 不會檢查儲存貯體的擁有者是相同或不同的帳戶。如果您在兩個儲存貯體都擁有相同時新增擁有者覆寫 AWS 帳戶，Amazon S3 會套用擁有者覆寫。它會授予完整許可給目的地儲存貯體擁有者，而且不會將後續更新複寫至來源物件存取控制清單 (ACL)。複本擁有者可以使用 PUT ACL 要求直接變更與複本相關聯的 ACL，但不是透過複寫。

若要指定擁有者覆寫選項，請將下列項目新增至 `Destination` 元素：

- `AccessControlTranslation` 元素，其可通知 Amazon S3 變更複本擁有權
- 元`Account`素，指定目 AWS 帳戶 的地值區擁有者

```
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    ...
    <Destination>
      ...
      <AccessControlTranslation>
           <Owner>Destination</Owner>
       </AccessControlTranslation>
      <Account>destination-bucket-owner-account-id</Account>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

下列複寫組態範例會通知 Amazon S3，將含 `Tax` 金鑰前綴的物件複寫至目的地儲存貯體，並變更複本的擁有權。

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Role>arn:aws:iam::account-id:role/role-name</Role>
   <Rule>
      <ID>Rule-1</ID>
      <Priority>1</Priority>
      <Status>Enabled</Status>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix>Tax</Prefix>
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::destination-bucket</Bucket>
         <Account>destination-bucket-owner-account-id</Account>
         <AccessControlTranslation>
            <Owner>Destination</Owner>
         </AccessControlTranslation>
      </Destination>
   </Rule>
</ReplicationConfiguration>
```

授予 Amazon S3 變更複本擁有權的許可

在與 IAM 角色相關聯的許可政策中，新增 `s3:ObjectOwnerOverrideToBucketOwner` 動作許可，以授予 Amazon S3 變更複本擁有權的許可。此為您在複寫組態中指定的 IAM 角色，其可讓 Amazon S3 擔任角色並代您複寫物件。

```
...
{
    "Effect":"Allow",
         "Action":[
       "s3:ObjectOwnerOverrideToBucketOwner"
    ],
    "Resource":"arn:aws:s3:::destination-bucket/*"
}
...
```

在目的地儲存貯體政策中新增許可，以允許變更複本所有權

目的地儲存貯體擁有者必須授予來源儲存貯體擁有者變更複本所有權的許可。目的地儲存貯體擁有者可授予來源儲存貯體擁有者 `s3:ObjectOwnerOverrideToBucketOwner` 動作的許可。這可讓目的地儲存貯體擁有者接受物件複本的所有權。下列範例儲存貯體政策陳述式說明如何執行此作業。

```
...
{
    "Sid":"1",
    "Effect":"Allow",
    "Principal":{"AWS":"source-bucket-account-id"},
    "Action":["s3:ObjectOwnerOverrideToBucketOwner"],
    "Resource":"arn:aws:s3:::destination-bucket/*"
}
...
```

其他考量

當您設定擁有者覆寫選項時有下列考量：

- 依預設，來源物件的擁有者也擁有複本。Amazon S3 會複寫物件版本以及與其相關聯的 ACL。

  如果您新增擁有者覆寫，Amazon S3 僅會複寫物件版本，而不會複寫 ACL。此外，Amazon S3 不會將後續變更複寫到來源物件 ACL。Amazon S3 會在複本上設定 ACL，將完整控制權授予目的地儲存貯體擁有者。
- 當您更新複寫組態以啟用或停用擁有者覆寫時，會發生下列情況。

  - 如果您將擁有者覆寫選項新增至複寫組態：

    當 Amazon S3 複寫物件版本時，它會捨棄與來源物件相關聯的 ACL。相反地，它會在複本上設定 ACL，將完整控制權授予目的地儲存貯體的擁有者。它不會複寫來源物件 ACL 的後續變更。不過，如果是在您設定擁有者覆寫選項之前即已複寫的物件版本，就不適用這項 ACL 變更。針對在設定擁有者覆寫之前即已複寫的來源物件，系統會繼續複寫其中的任何 ACL 更新 (因為物件及其複本的擁有者仍然相同)。
  - 如果您移除複寫組態中的擁有者覆寫選項：

    Amazon S3 會將來源儲存貯體中出現的新物件與相關聯的 ACL 複寫至目的地儲存貯體。如果物件是在您移除擁有者覆寫之前即已複寫，Amazon S3 就不會複寫 ACL，因為 Amazon S3 所做的物

件擁有權變更仍然有效。亦即，針對在設定擁有者覆寫時複寫的物件版本，系統仍然不會複寫其上的 ACL。

如何變更複本擁有者

當複寫組態中的##和##儲存貯體擁有不同時 AWS 帳戶，您可以告訴 Amazon S3 將複本擁有權變更為擁有##儲存貯體 AWS 帳戶 的擁有權。此範例說明如何使用 Amazon S3 主控台以及變 AWS CLI 更複本擁有權。如需詳細資訊，請參閱 變更複本擁有者。

> **Note**
>
> 當您使用 S3 複寫且來源和目的地儲存貯體屬於不同時 AWS 帳戶，目的地儲存貯體的擁有者可以停用 ACL (使用儲存貯體擁有者強制執行物件擁有權設定)，將複本擁有權變更為擁有目的 AWS 帳戶 地儲存貯體的擁有權。此設定會模擬現有的擁有者覆寫行為，而無需 `s3:ObjectOwnerOverrideToBucketOwner` 許可。這意味著使用儲存貯體擁有者強制執行設定複寫至目的地儲存貯體的所有物件都由目的地儲存貯體擁有者所擁有。如需「物件擁有權」的詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

如需 AWS Key Management Service 在跨帳戶案例中使用資料夾端加密來設定複寫的詳細資訊，請參閱。跨帳戶案例之授予其他許可

使用 S3 主控台

如需 step-by-step 指示，請參閱為相同帳戶擁有的來源和目的地儲存貯體設定複寫。本主題提供在值區為相同且不同擁有時設定複製組態的指示 AWS 帳戶。

使用 AWS CLI

若要使用變更複本擁有權 AWS CLI，您可以建立儲存貯體、在儲存貯體上啟用版本控制、建立 IAM 角色以授予 Amazon S3 複寫物件的權限，以及將複寫組態新增至來源儲存貯體。請在複寫組態中指示 Amazon S3 變更複本擁有者。您也可以測試設定。

當來源和目的地值區為不同時變更複本擁有權 AWS 帳戶 (AWS CLI)

1. 在此範例中，您會以兩個不同的方式建立##值區和#的地值區 AWS 帳戶。 AWS CLI 使用兩個具名的設定檔進行配置。此範例分別使用名為 `acctA` 和 `acctB` 的描述檔。如需設定憑證描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》https://docs.aws.amazon.com/cli/latest/userguide/cli-multiple-profiles.html中的具名描述檔。

> **Important**
>
> 用於此練習的描述檔必須有必要的許可。例如，您可以在複寫組態中指定 Amazon S3 可以擔任的 IAM 角色。只有當您所用的描述檔有 `iam:PassRole` 許可時才可執行此作業。如果您使用管理員使用者憑證建立具名描述檔，即可執行所有任務。如需詳細資訊，請參閱《IAM 使用者指南》中的授與使用者將角色傳遞給 AWS 服務的權限。

您需要確定這些描述檔有必要的許可。例如，複寫組態包含 Amazon S3 可以擔任的 IAM 角色。用以連接此類組態和儲存貯體的具名描述檔只有在具備 `iam:PassRole` 許可時才可執行此作業。如果您在建立這些具名描述檔時指定管理員使用者憑證，則它們全都具備所有許可。如需詳細資訊，請參閱《IAM 使用者指南》中的授與使用者將角色傳遞給 AWS 服務的權限。

2. 建立 *source* 儲存貯體並啟用版本控制。本範例會在美國東部 (維吉尼亞北部) (us-east-1) 區域中建立 *source* 儲存貯體。

```
aws s3api create-bucket \
--bucket source \
--region us-east-1 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket source \
--versioning-configuration Status=Enabled \
--profile acctA
```

3. 建立 *destination* 儲存貯體並啟用版本控制。本範例會在美國西部 (奧勒岡) (us-west-2) 區域中建立 *destination* 儲存貯體。使用不同於 *source* 儲存貯體所使用的 AWS 帳戶 設定檔。

```
aws s3api create-bucket \
--bucket destination \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctB
```

```
aws s3api put-bucket-versioning \
--bucket destination \
--versioning-configuration Status=Enabled \
```

```
--profile acctB
```

4. 您必須在 *destination* 儲存貯體政策中新增許可，以允許複本擁有權的變更。

   a. 將下列政策儲存至 *destination-bucket-policy*.json。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "destination_bucket_policy_sid",
      "Principal": {
        "AWS": "source-bucket-owner-account-id"
      },
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete",
        "s3:ObjectOwnerOverrideToBucketOwner",
        "s3:ReplicateTags",
        "s3:GetObjectVersionTagging"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::destination/*"
      ]
    }
  ]
}
```

   b. 將以上原則放到 *destination* 儲存貯體：

```
aws s3api put-bucket-policy --region $ {destination_region} --
bucket $ {destination} --policy file://destination_bucket_policy.json
```

5. 建立 IAM 角色。您可以在複寫組態中指定稍後要新增至 *source* 儲存貯體的角色，Amazon S3 就會擔任此角色以代您複寫物件。建立 IAM 角色需要兩個步驟：

   - 建立角色。
   - 將許可政策連接到角色。

   a. 建立 IAM 角色。

i. 複製下列信任政策，並將它儲存至本機電腦目前目錄下名稱為 s3-role-trust-policy.json 的檔案中。此政策會授予 Amazon S3 擔任該角色的許可。

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

ii. 執行下列 AWS CLI 命令以建立角色。

```
$ aws iam create-role \
--role-name replicationRole \
--assume-role-policy-document file://s3-role-trust-policy.json  \
--profile acctA
```

b. 將許可政策連接到角色。

i. 複製下列許可政策，並將它儲存至本機電腦目前目錄中名為 s3-role-perm-pol-changeowner.json 的檔案。此政策會授予各種 Amazon S3 儲存貯體與物件動作的許可。在下列步驟中，您需要建立 IAM 角色，並將此政策連接至該角色。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::source/*"
         ]
      },
```

```
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::source"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ObjectOwnerOverrideToBucketOwner",
            "s3:ReplicateTags",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":"arn:aws:s3:::destination/*"
      }
   ]
}
```

ii. 若要建立政策並將它連接至角色，請執行下列命令。

```
$ aws iam put-role-policy \
--role-name replicationRole \
--policy-document file://s3-role-perm-pol-changeowner.json \
--policy-name replicationRolechangeownerPolicy \
--profile acctA
```

6. 將複寫組態新增到來源儲存貯體。

a. AWS CLI 需要將複寫組態指定為 JSON。將下列 JSON 儲存至您本機電腦目前目錄中名為 `replication.json` 的檔案。在組態中，新增 `AccessControlTranslation` 指出複本擁有權的變更。

```
{
   "Role":"IAM-role-ARN",
   "Rules":[
      {
         "Status":"Enabled",
```

```
            "Priority":1,
            "DeleteMarkerReplication":{
               "Status":"Disabled"
            },
            "Filter":{
            },
            "Status":"Enabled",
            "Destination":{
               "Bucket":"arn:aws:s3:::destination",
               "Account":"destination-bucket-owner-account-id",
               "AccessControlTranslation":{
                  "Owner":"Destination"
               }
            }
         }
      ]
}
```

b. 提供 *destination* 儲存貯體擁有者帳戶 ID 和 *IAM-role-ARN* 的值，以編輯 JSON。儲存變更。

c. 執行下列命令，將複寫組態新增至來源儲存貯體。提供 *source* 儲存貯體名稱。

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket source \
--profile acctA
```

7. 在 Amazon S3 主控台中檢查複本擁有權。

a. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

b. 將物件新增到##儲存貯體。確認###值區包含物件複本，以及複本的擁有權已變更為擁有#的地值區 AWS 帳戶 的擁有權。

使用 AWS 軟體開發套件

如需新增複寫組態的程式碼範例，請參閱 使用 AWS 軟體開發套件。您需要正確修改複寫組態。如需相關概念資訊，請參閱 變更複本擁有者。

使用 S3 複寫時間控制 (S3 RTC) 來達到合規要求

S3 複寫時間控制 (S3 RTC) 可協助滿足資料複寫的合規性或業務要求，讓您清楚掌握 Amazon S3 複寫時間。S3 RTC 會在數秒內複寫您上傳至 Amazon S3 的多數物件，以及在 15 分鐘內複寫 99.99% 的這些物件。

S3 RTC 預設包括 S3 複寫指標和 Amazon S3 事件通知，您可以使用這些指標來監控擱置複寫的 S3 API 操作總數、擱置複寫的物件總大小，以及複寫時間上限。您可以獨立於 S3 RTC 啟用複寫指標。如需詳細資訊，請參閱監控複寫指標的進度。此外，如果物件複寫超過 15 分鐘閾值或之後複寫，S3 RTC 會提供通知儲存貯體擁有者的 `OperationMissedThreshold` 和 `OperationReplicatedAfterThreshold` 事件。

透過 S3 RTC，Amazon S3 事件可在下列罕見的情況下通知您：物件未在 15 分鐘內複寫，以及這些物件在 15 分鐘閾值後複寫。Amazon S3 活動可透過 Amazon SQS、Amazon SNS 或 AWS Lambda。如需詳細資訊，請參閱 the section called “Amazon S3 事件通知”。

主題

- S3 複寫時間控制
- 使用 S3 RTC 的複寫指標
- 使用 Amazon S3 事件通知追蹤複寫物件
- S3 RTC 的最佳實務和指導方針
- 啟用 S3 複寫時間控制 (S3 RTC)

S3 複寫時間控制

您可以開始使用 S3 複寫時間控制 (S3 RTC) 搭配新的或現有的複寫規則。您可以選擇將複寫規則套用至整個 S3 儲存貯體，或套用至具有特定前綴或標籤的 Amazon S3 物件。當您啟用 S3 RTC 時，也會在複寫規則上啟用複寫指標。

如果您使用最新版本的複寫組態 (亦即您在複寫組態規則中指定 `Filter` 元素)，根據預設，Amazon S3 就不會複寫刪除標記。不過，您可以將刪除標記複製新增至 non-tag-based 規則。

> **Note**
>
> 複寫指標的費率與 Amazon CloudWatch 自訂指標相同。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

如需有關使用 S3 RTC 建立規則的詳細資訊，請參閱 啟用 S3 複寫時間控制 (S3 RTC)。

使用 S3 RTC 的複寫指標

啟用 S3 複寫時間控制 (S3 RTC) 的複寫規則會發佈複寫指標。使用複寫指標即可監控待複寫的 S3 API 操作總數、待複寫的物件總大小、目的地區域的複寫時間上限，以及複寫失敗的操作總數。然後，您可以監控個別複寫的每個資料集。

複寫指標可在啟用 S3 RTC 後的 15 分鐘內使用。複寫指標可透過 Amazon S3 主控台、Amazon S3 API、 AWS 開發套件、AWS Command Line Interface (AWS CLI) 和 Amazon CloudWatch 取得。如需詳細資訊，請參閱 使用 Amazon 監控指標 CloudWatch。

如需有關透過 Amazon S3 主控台尋找複寫指標的詳細資訊，請參閱 使用 Amazon S3 主控台檢視複寫指標。

使用 Amazon S3 事件通知追蹤複寫物件

您可以透過監控 S3 複寫時間控制 (S3 RTC) 發行的特定事件通知，來追蹤未在 15 分鐘內複寫之物件的複寫時間。當有資格使用 S3 RTC 複寫的物件未在 15 分鐘內複寫，以及該物件在 15 分鐘閾值後複寫時，會發佈這些事件。

複寫事件可在啟用 S3 RTC 後的 15 分鐘內使用。Amazon S3 活動可透過 Amazon SQS、Amazon SNS 或 AWS Lambda。如需詳細資訊，請參閱 Amazon S3 事件通知。

S3 RTC 的最佳實務和指導方針

在 Amazon S3 中使用 S3 複寫時間控制 (S3 RTC) 複寫資料時，請遵循這些最佳實務指導方針，以最佳化工作負載的複寫效能。

主題

- Amazon S3 複寫與請求率效能的指導方針
- 預估您的複寫請求率
- 超過 S3 RTC 資料傳輸速率限制
- AWS KMS 加密物件複寫要求率

Amazon S3 複寫與請求率效能的指導方針

從 Amazon S3 上傳和擷取儲存時，您的應用程式可以在請求效能中實現每秒數千筆交易。例如，在 S3 儲存貯體中，應用程式每秒至少可達到每個字首每秒 3,500 個 PUT/COPY/POST/DELETE 或

5,500 個 GET/HEAD 請求，包括 S3 複寫代表您進行的請求。在儲存貯體中的字首數不受限制。您可以並行讀取以提升您的讀取或寫入的效能。例如，如果您在 S3 儲存貯體裡建立 10 個字首，平行讀取，您可以調整讀取效能至每秒 55,000 讀取請求。

Amazon S3 會自動調整以回應高於這些指導方針的持續請求率，或與 LIST 請求並行的持續請求率。當 Amazon S3 在內部針對新請求率最佳化時，您將會暫時收到 HTTP 503 請求回應，直到最佳化完成為止。當每秒的請求率增加，或當您第一次啟用 S3 RTC 時，可能會發生此情況。在這些期間，您的複寫延遲可能會增加。S3 RTC 服務水準協議 (SLA) 不適用 Amazon S3 效能指導方針超過每秒請求數的期間。

S3 RTC SLA 也不適用您的複寫資料傳輸率超過預設的 1 Gbps 限制的期間。如果預期複寫傳輸率超過 1 Gbps，您可以聯絡 AWS Support 中心或使用 Service Quotas (服務配額) 來請求增加限制。

預估您的複寫請求率

您的請求率總計 (包括 Amazon S3 代表您進行的複寫請求) 應符合複寫來源和目的地儲存貯體的 Amazon S3 請求率指導方針。對於每個複寫的物件，Amazon S3 複寫最多會對來源儲存貯體形成五個 GET/HEAD 請求和一個 PUT 請求，以及對每個目的地儲存貯體形成一個 PUT 請求。

例如，如果您預期每秒複寫 100 個物件，Amazon S3 複寫可能會代表您執行額外 100 個 PUT 請求，因此，對來源 S3 儲存貯體每秒總計 200 個 PUT 請求。Amazon S3 複寫也可能會執行最多 500 個 GET/HEAD (每個複寫的物件 5 個 GET/HEAD 請求)。

> **Note**
>
> 對於每個複寫的物件，只會產生一個 PUT 請求成本。如需詳細資訊，請參閱有關複寫的 Amazon S3 常見問答集中的定價資訊。

超過 S3 RTC 資料傳輸速率限制

如果您預期 S3 複寫時間控制資料傳輸率超過預設的 1 Gbps 限制，請聯絡 AWS Support 中心或使用 Service Quotas (服務配額) 來請求增加限制。

AWS KMS 加密物件複寫要求率

當您使用 Amazon S3 複寫複寫使用伺服器端加密 (SSE-KMS) 加密的物件時，會套用 AWS Key Management Service (AWS KMS) 每秒請求限制。 AWS KMS 可能會拒絕其他有效的要求，因為您的要求率超過每秒要求數目的限制。當請求被限制時， AWS KMS 返回一個`ThrottlingException`錯誤。請 AWS KMS 求率限制適用於您直接提出的請求，以及 Amazon S3 代表您發出的請求。

例如，如果您希望每秒複寫 1,000 個物件，您可以從請求速率限制中減去 2,000 個 AWS KMS 要求。產生的每秒請求率可用於不包括複製的 AWS KMS 工作負載。您可以使用 Amazon 中的AWS KMS 請求指標 CloudWatch來監控您的總 AWS KMS 請求率 AWS 帳戶。

啟用 S3 複寫時間控制 (S3 RTC)

S3 複寫時間控制 (S3 RTC) 可協助滿足資料複寫的合規性或業務要求，讓您清楚掌握 Amazon S3 複寫時間。S3 RTC 會在數秒內複寫您上傳至 Amazon S3 的多數物件，以及在 15 分鐘內複寫 99.99% 的這些物件。

使用 S3 RTC 時，您可以監控擱置中複寫之物件的總數和大小，以及對目標區域的最長複寫時間。複寫指標可透過AWS Management Console和 Amazon CloudWatch 使用者指南取得。如需詳細資訊，請參閱「the section called “S3 複寫指標 CloudWatch ”」。

使用 S3 主控台

如需 step-by-step 指示，請參閱為相同帳戶擁有的來源和目的地儲存貯體設定複寫。本主題提供指示，如何在儲存貯體擁有相同且不 AWS 帳戶同的儲存貯體時，在複寫組態中啟用 S3 RTC。

使用 AWS CLI

若要在啟用 S3 RTC 的情況下使用複寫物件，請建立儲存貯體、在儲存貯體上啟用版本控制、建立 IAM 角色以授予 Amazon S3 複寫物件的權限，以及將複寫組態新增至來源儲存貯體。 AWS CLI 複寫組態必須已啟用 S3 複寫時間控制 (S3 RTC)。

在啟用 S3 RTC 的情況下進行複寫 (AWS CLI)

- 在下列範例中，我們會設定 `ReplicationTime` 和 `Metric`，並將複寫組態新增至來源儲存貯體。

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "DeleteMarkerReplication": {
                "Status": "Disabled"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::destination",
```

```
                "Metrics": {
                    "Status": "Enabled",
                    "EventThreshold": {
                        "Minutes": 15
                    }
                },
                "ReplicationTime": {
                    "Status": "Enabled",
                    "Time": {
                        "Minutes": 15
                    }
                }
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

> **Important**
>
> `Metrics:EventThreshold:Minutes` 和 `ReplicationTime:Time:Minutes` 只能使用 15 作為有效值。

使用適用於 Java 的 AWS 開發套件

以下是使用 S3 複寫時間控制 (S3 RTC) 新增複寫組態的 Java 範例。

```
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.DeleteMarkerReplication;
import software.amazon.awssdk.services.s3.model.Destination;
import software.amazon.awssdk.services.s3.model.Metrics;
import software.amazon.awssdk.services.s3.model.MetricsStatus;
import software.amazon.awssdk.services.s3.model.PutBucketReplicationRequest;
import software.amazon.awssdk.services.s3.model.ReplicationConfiguration;
import software.amazon.awssdk.services.s3.model.ReplicationRule;
import software.amazon.awssdk.services.s3.model.ReplicationRuleFilter;
import software.amazon.awssdk.services.s3.model.ReplicationTime;
import software.amazon.awssdk.services.s3.model.ReplicationTimeStatus;
import software.amazon.awssdk.services.s3.model.ReplicationTimeValue;
```

```
public class Main {

  public static void main(String[] args) {
    S3Client s3 = S3Client.builder()
      .region(Region.US_EAST_1)
      .credentialsProvider(() -> AwsBasicCredentials.create(
          "AWS_ACCESS_KEY_ID",
          "AWS_SECRET_ACCESS_KEY")
      )
      .build();

    ReplicationConfiguration replicationConfig = ReplicationConfiguration
      .builder()
      .rules(
          ReplicationRule
            .builder()
            .status("Enabled")
            .priority(1)
            .deleteMarkerReplication(
                DeleteMarkerReplication
                    .builder()
                    .status("Disabled")
                    .build()
            )
            .destination(
                Destination
                    .builder()
                    .bucket("destination_bucket_arn")
                    .replicationTime(
                        ReplicationTime.builder().time(
                            ReplicationTimeValue.builder().minutes(15).build()
                        ).status(
                            ReplicationTimeStatus.ENABLED
                        ).build()
                    )
                    .metrics(
                        Metrics.builder().eventThreshold(
                            ReplicationTimeValue.builder().minutes(15).build()
                        ).status(
                            MetricsStatus.ENABLED
                        ).build()
                    )
                    .build()
```

```
            )
            .filter(
                ReplicationRuleFilter
                    .builder()
                    .prefix("testtest")
                    .build()
            )
        .build())
        .role("role_arn")
        .build();

    // Put replication configuration
    PutBucketReplicationRequest putBucketReplicationRequest =
 PutBucketReplicationRequest
      .builder()
      .bucket("source_bucket")
      .replicationConfiguration(replicationConfig)
      .build();

    s3.putBucketReplication(putBucketReplicationRequest);
  }
}
```

如需詳細資訊，請參閱 使用 S3 複寫時間控制 (S3 RTC) 來達到合規要求。

複寫加密的物件 (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

複製已使用伺服器端加密進行加密的物件時，有一些特殊的考量。Amazon S3 支援下列三種類型的伺服器端加密：

- 使用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密

- 使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 進行伺服器端加密
- 使用 AWS KMS 金鑰 (DSSE-KMS) 進行雙層伺服器端加密
- 使用客戶提供金鑰 (SSE-C) 的伺服器端加密

如需伺服器端加密的詳細資訊，請參閱「the section called “伺服器端加密”」。

本主題說明指示 Amazon S3 複寫已使用伺服器端加密進行加密的物件所需的許可。本主題也提供您可以新增的其他組態元素，以及授與複寫加密物件所需權限的範例 AWS Identity and Access Management (IAM) 政策。

如需包含 step-by-step 指示的範例，請參閱啟用加密物件的複寫。如需建立複寫組態的資訊，請參閱複製物件概觀。

> **Note**
>
> 您可以 AWS KMS keys 在 Amazon S3 中使用多區域。但是，Amazon S3 目前將多區域金鑰視為單區域金鑰，並且不使用金鑰的多區域功能。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的使用多區域金鑰。

主題

預設儲存貯體加密如何影響複寫

在您啟用域複寫目的地儲存貯體的預設加密之後，適用下列加密行為：

- 如果未加密來源儲存貯體中的物件，則會使用目的地儲存貯體的預設加密設定來加密目的地儲存貯體中的複本物件。因此，來源物件的實體標籤 (ETag) 與複本物件的 ETag 不同。如果您有使用 ETag 的應用程式，則必須更新這些應用程式以解決此差異。
- 如果來源儲存貯體中的物件使用伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3)、使用 () 金鑰 (SSE-KMS AWS KMS) 進行伺服器端加密，或使用金鑰的雙層伺服器端加密 (DSSE-KMS) 加密，目

的地儲存貯體中的複本物件會使用與 AWS KMS 來源物件相同的加密類型。 AWS Key Management Service 不會使用目的地儲存貯體的預設加密設定。

複寫使用 SSE-C 加密的物件

透過使用伺服器端加密搭配客戶提供的金鑰 (SSE-C)，您可以管理自己的專屬加密金鑰。使用 SSE-C，您負責管理金鑰，而 Amazon S3 則管理加密和解密程序。您必須提供加密金鑰作為請求的一部分，但不需要撰寫任何程式碼來執行物件加密或解密。當您上傳物件時，Amazon S3 會使用您提供的金鑰加密物件。然後 Amazon S3 會從記憶體中清除該金鑰。當您擷取物件時，您必須在要求中提供相同的加密金鑰。如需詳細資訊，請參閱 the section called “客戶提供的加密金鑰 (SSE-C)”。

S3 複寫支援使用 SSE-C 加密的物件。 您可以在 Amazon S3 主控台或使用 AWS 開發套件設定 SSE-C 物件複寫，方式與設定未加密物件的複寫相同。除了目前複寫所需的許可外，沒有其他 SSE-C 許可。

如果新上傳的 SSE-C 加密物件符合資格，S3 複寫會根據 S3 複寫組態自動複寫這些物件。如需複寫儲存貯體中的現有物件，請使用 S3 批次複寫。如需複寫物件的詳細資訊，請參閱 the section called “設定即時複製” 和 the section called “複寫現有物件”。

複寫 SSE-C 物件沒有額外費用。如需複寫定價的詳細資訊，請參閱 Amazon S3 定價頁面。

複寫使用 SSE-S3、SSE-KMS 或 DSSE-KMS 加密的物件

根據預設，Amazon S3 不會複寫使用 SSE-KMS 或 DSSE-KMS 加密的物件。本節說明您可以新增，以指示 Amazon S3 複寫這些物件的額外組態元素。

如需包含 step-by-step 指示的範例，請參閱啟用加密物件的複寫。如需建立複寫組態的資訊，請參閱複製物件概觀。

在複寫組態中指定其他資訊

在複寫組態中，請執行下列作業：

- 在複寫組態中的`Destination`元素中，新增您希望 Amazon S3 用來加密物件複本之對稱 AWS KMS 客戶受管金鑰的 ID，如以下範例複寫組態所示。
- 透過啟用使用 KMS 金鑰 (SSE-KMS 或 DSSE-KMS) 加密物件的複寫，來明確加入。若要選擇加入，請新增 `SourceSelectionCriteria` 元素，如以下範例複寫組態所示。

```
<ReplicationConfiguration>
```

```
   <Rule>
      ...
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
           <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>

      <Destination>
          ...
          <EncryptionConfiguration>
             <ReplicaKmsKeyID>AWS KMS key ARN or Key Alias ARN that's in the same AWS #
# as the destination bucket.</ReplicaKmsKeyID>
          </EncryptionConfiguration>
       </Destination>
      ...
   </Rule>
</ReplicationConfiguration>
```

> **Important**
>
> KMS 金鑰必須建立在與目標儲存貯體 AWS 區域 相同的位置。
> KMS 金鑰必須有效。PutBucketReplication API 操作不會檢查 KMS 金鑰是否有效。如果使用無效的 KMS 金鑰，您會收到 HTTP 200 OK 狀態碼回應，但複寫會失敗。

下列範例顯示複寫組態，其中包含選用組態元素。此複寫組態具有一項規則。該規則會套用至金鑰前綴為 Tax 的物件。Amazon S3 會使用特定 AWS KMS key ID 來加密這些物件複本。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration>
   <Role>arn:aws:iam::account-id:role/role-name</Role>
   <Rule>
      <ID>Rule-1</ID>
      <Priority>1</Priority>
      <Status>Enabled</Status>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix>Tax</Prefix>
      </Filter>
```

```
      <Destination>
         <Bucket>arn:aws:s3:::example-s3-destination-bucket</Bucket>
         <EncryptionConfiguration>
            <ReplicaKmsKeyID>AWS KMS key ARN or Key Alias ARN that's in the same
 AWS ## as the destination bucket. (S3 uses this key to encrypt object replicas.)</
ReplicaKmsKeyID>
         </EncryptionConfiguration>
      </Destination>
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
            <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>
   </Rule>
</ReplicationConfiguration>
```

授予 IAM 角色的額外許可

若要使用 SSE-S3、SSE-KMS 或 DSSE-KMS 複寫靜態加密的物件，請將下列額外權限授與您在複寫組態中指定的 AWS Identity and Access Management (IAM) 角色。您可以透過更新與 IAM 角色相關聯的許可政策，來授予這些許可。

- 來源物件的 **`s3:GetObjectVersionForReplication`** 動作 – 此動作允許 Amazon S3 複寫未加密的物件，以及使用 SSE-S3、SSE-KMS 或 DSSE-KMS 搭配伺服器端加密建立的物件。

  > **Note**
  >
  > 建議您使用 `s3:GetObjectVersionForReplication` 動作，不要使用 `s3:GetObjectVersion` 動作，因為 `s3:GetObjectVersionForReplication` 只會提供 Amazon S3 複寫所需的最低許可。此外，`s3:GetObjectVersion` 動作還允許複寫未加密物件和 SSE-S3 加密物件，但不允許複寫使用 KMS 金鑰 (SSE-KMS 或 DSSE-KMS) 加密的物件。

- **`kms:Decrypt`**和 KMS 金鑰的**`kms:Encrypt`** AWS KMS 動作
  - 您必須針對用來解密來源物件的 AWS KMS key 授予 `kms:Decrypt` 許可。
  - 您必須針對用來加密物件複本的 AWS KMS key 授予 `kms:Encrypt` 許可。
- **`kms:GenerateDataKey`**複寫純文字物件的動作 – 如果您要將純文字物件複寫到預設已啟用 SSE-KMS 或 DSSE-KMS 加密的儲存貯體，則必須在 IAM 政策中包含目的地加密內容和 KMS 金鑰的 `kms:GenerateDataKey` 許可。

建議您使用條件索引鍵，僅限於目的地值區和物 AWS KMS 件的這些權限。擁 AWS 帳戶 有 IAM 角色的擁有權限必須具有政策中所列 KMS 金鑰的`kms:Encrypt`和`kms:Decrypt`動作的許可。如果 KMS 金鑰由另一個金鑰所擁有 AWS 帳戶，KMS 金鑰的擁有者必須將這些權限授與擁 AWS 帳戶 有 IAM 角色的擁有者。如需管理這些 KMS 金鑰存取權的詳細資訊，請參閱 AWS Key Management Service 開發人員指南 AWS KMS中的搭配使用 IAM 政策。

S3 儲存貯體金鑰和複寫

若要搭配 S3 儲存貯體金鑰使用複寫，用於加密物件複本的 KMS 金鑰 AWS KMS key 政策必須包含呼叫主體的`kms:Decrypt`權限。在使用 S3 儲存貯體金鑰之前，對 `kms:Decrypt` 的呼叫會驗證 S3 儲存貯體金鑰的完整性。如需詳細資訊，請參閱 使用 S3 儲存貯體金鑰與複寫。

針對來源或目的地儲存貯體啟用 S3 儲存貯體金鑰時，加密內容將是儲存貯體 Amazon Resource Name (ARN)，而不是物件的 ARN (例如，`arn:aws:s3:::`*`bucket_ARN`*)。您必須更新 IAM 政策，才能將儲存貯體 ARN 用於加密內容：

```
"kms:EncryptionContext:aws:s3:arn": [
"arn:aws:s3:::bucket_ARN"
]
```

如需詳細資訊，請參閱〈使用 REST API〉一節中的 加密內容 (x-amz-server-side-encryption-context) 和 啟用 S3 儲存貯體金鑰之前，要注意的變更。

範例政策 – 搭配複寫使用 SSE-S3 和 SSE-KMS

下列範例 IAM 政策顯示了搭配複寫使用 SSE-S3 和 SSE-KMS 的陳述式。

Example – 透過不同的目的地儲存貯體使用 SSE-KMS

下列範例政策顯示搭配不同目的地儲存貯體使用 SSE-KMS 的陳述式。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action": ["kms:Decrypt"],
         "Effect": "Allow",
         "Condition": {
             "StringLike": {
                  "kms:ViaService": "s3.source-bucket-region.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn": [
                      "arn:aws:s3:::example-s3-source-bucket/key-prefix1*"
```

```
                ]
            }
         },
         "Resource": [
            "List of AWS KMS key ARNs that are used to encrypt source objects."
         ]
      },
      {
         "Action": ["kms:Encrypt"],
         "Effect": "Allow",
         "Condition": {
            "StringLike": {
               "kms:ViaService": "s3.destination-bucket-1-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn": [
                  "arn:aws:s3:::example-s3-destination-bucket1/key-prefix1*"
               ]
         }
         },
         "Resource": [
            "AWS KMS key ARNs (in the same AWS ## as destination bucket 1). Used to
 encrypt object replicas created in destination bucket 1."
         ]
      },
      {
         "Action": ["kms:Encrypt"],
         "Effect": "Allow",
         "Condition": {
            "StringLike": {
               "kms:ViaService": "s3.destination-bucket-2-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn": [
                  "arn:aws:s3:::example-s3-destination-bucket2/key-prefix1*"
               ]
         }
         },
         "Resource": [
            "AWS KMS key ARNs (in the same AWS ## as destination bucket 2). Used to
 encrypt object replicas created in destination bucket 2."
         ]
      }
   ]
}
```

Example – 複寫使用 SSE-S3 和 SSE-KMS 建立的物件

下列是完整 IAM 政策，其會授予必要的許可來複寫未加密的物件、使用 SSE-S3 建立的物件，以及使用 SSE-KMS 建立的物件。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::example-s3-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::example-s3-source-bucket/key-prefix1*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete"
         ],
         "Resource":"arn:aws:s3:::example-s3-destination-bucket/key-prefix1*"
      },
      {
         "Action":[
            "kms:Decrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.source-bucket-region.amazonaws.com",
```

```
                "kms:EncryptionContext:aws:s3:arn":[
                   "arn:aws:s3:::example-s3-source-bucket/key-prefix1*"
                ]
             }
          },
          "Resource":[
            "List of the AWS KMS key ARNs that are used to encrypt source objects."
          ]
       },
       {
          "Action":[
             "kms:Encrypt"
          ],
          "Effect":"Allow",
          "Condition":{
             "StringLike":{
                "kms:ViaService":"s3.destination-bucket-region.amazonaws.com",
                "kms:EncryptionContext:aws:s3:arn":[
                   "arn:aws:s3:::example-s3-destination-bucket/prefix1*"
                ]
             }
          },
          "Resource":[
             "AWS KMS key ARNs (in the same AWS ## as the destination bucket) to use for
 encrypting object replicas"
          ]
       }
    ]
}
```

Example – 使用 S3 儲存貯體金鑰來複寫物件

以下是完整的 IAM 政策，可以授予使用 S3 儲存貯體金鑰來複寫物件的必要許可。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
```

```
            "arn:aws:s3:::example-s3-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::example-s3-source-bucket/key-prefix1*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete"
         ],
         "Resource":"arn:aws:s3:::example-s3-destination-bucket/key-prefix1*"
      },
      {
         "Action":[
            "kms:Decrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.source-bucket-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::example-s3-source-bucket"
               ]
            }
         },
         "Resource":[
           "List of the AWS KMS key ARNs that are used to encrypt source objects."
         ]
      },
      {
         "Action":[
            "kms:Encrypt"
         ],
         "Effect":"Allow",
         "Condition":{
```

```
            "StringLike":{
               "kms:ViaService":"s3.destination-bucket-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::example-s3-destination-bucket"
               ]
            }
         },
         "Resource":[
            "AWS KMS key ARNs (in the same AWS ## as the destination bucket) to use for
 encrypting object replicas"
         ]
      }
   ]
}
```

跨帳戶案例之授予其他許可

在跨帳戶案例中，來源和目的地儲存貯體擁有不同的情況下 AWS 帳戶，您可以使用 KMS 金鑰來加密物件複本。但是，KMS 金鑰擁有者必須授予來源儲存貯體擁有者使用 KMS 金鑰的許可。

> **Note**
>
> 如果您需要跨帳戶複寫 SSE-KMS 資料，則複寫規則必須從目的地帳戶指定客戶管理 AWS KMS 的金鑰。 AWS 受管金鑰不允許跨帳戶使用，因此無法用於執行跨帳戶複製。

授予來源儲存貯體擁有者使用 KMS 金鑰的許可 (AWS KMS 主控台)

1. 請登入 AWS Management Console 並開啟 AWS KMS 主控台，網址為 https://console.aws.amazon.com/kms。
2. 若要變更 AWS 區域，請使用頁面右上角的「地區」選取器。
3. 若要檢視您所建立及管理帳戶中的金鑰，請在導覽窗格中選擇Customer managed keys (客戶受管金鑰)。
4. 選擇 KMS 金鑰。
5. 在一般組態下，選擇金鑰政策標籤。
6. 向下捲動至「其他 AWS 帳戶」。
7. 選擇 [新增其他] AWS 帳戶。

   這時系統顯示 AWS 帳戶 「其他」對話框

8. 在對話方塊中，選擇 [新增其他] AWS 帳戶。針對 arn:aws:iam::，輸入來源儲存貯體帳戶 ID。
9. 選擇儲存變更。

若要授予來源儲存貯體擁有者使用 KMS 金鑰的許可 (AWS CLI)

- 若要取得有關 `put-key-policy` AWS Command Line Interface (AWS CLI) 指令的資訊，請參閱《指 AWS CLI 令參考》put-key-policy中的〈〉。如需基礎 `PutKeyPolicy` API 操作的相關資訊，請參閱《AWS Key Management Service API 參考》https://docs.aws.amazon.com/kms/latest/APIReference/中的 PutKeyPolicy。

AWS KMS 交易配額考量

當您在啟用跨區域複寫 (CRR) 之後新增許多具有 AWS KMS 加密功能的新物件時，您可能會遇到節流 (HTTP 錯誤)。`503 Service Unavailable`當每秒 AWS KMS 交易數量超過目前配額時，即會發生限流。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的配額。

若要請求提升配額，請使用 Service Quotas。如需詳細資訊，請參閱請求增加配額。如果您的地區不支援 Service Quotas，請提出 AWS Support 案例。

啟用加密物件的複寫

根據預設，Amazon S3 不會複寫使用伺服器端加密 AWS Key Management Service () 金鑰 (SSE-KMS AWS KMS) 加密的物件，或使用金鑰 (DSSE-KMS) 進行雙層伺服器端加密的 AWS KMS 物件。若要複寫 SSE-KMS 或 DSS-KMS 加密的物件，請務必修改儲存貯體複寫組態，指示 Amazon S3 複寫這些物件。此範例說明如何使用 Amazon S3 主控台和 AWS Command Line Interface (AWS CLI) 變更儲存貯體複寫組態以啟用複寫加密物件。

如需詳細資訊，請參閱 複寫加密的物件 (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)。

> **Note**
>
> 針對來源或目的地儲存貯體啟用 S3 儲存貯體金鑰時，加密內容將是儲存貯體 Amazon Resource Name (ARN)，而不是物件的 ARN。您必須更新 IAM 政策，才能將儲存貯體 ARN 用於加密內容。如需詳細資訊，請參閱 S3 儲存貯體金鑰和複寫。

> **Note**
>
> 您可以 AWS KMS keys 在 Amazon S3 中使用多區域。但是，Amazon S3 目前將多區域金鑰視為單區域金鑰，並且不使用金鑰的多區域功能。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的使用多區域金鑰。

使用 S3 主控台

如需 step-by-step 指示，請參閱為相同帳戶擁有的來源和目的地儲存貯體設定複寫。本主題提供指示，說明如何在值區為相同且不同的值區所擁有時設定複製組態 AWS 帳戶。

使用 AWS CLI

若要使用複製加密物件 AWS CLI，請執行下列動作：

- 建立來源和目的地儲存貯體，並對這些儲存貯體啟用版本控制。
- 建立一個 AWS Identity and Access Management (IAM) 服務角色，以授予 Amazon S3 複寫物件的權限。IAM 角色許可包含複寫加密物件的必要許可。
- 將複寫組態新增至來源儲存貯體。複寫組態提供使用 KMS 金鑰複寫加密物件的相關資訊。
- 將加密物件新增到來源儲存貯體。
- 測試設定，確認您的加密物件正在複寫到目的地儲存貯體。

下列步驟將逐步引導您完成此程序。

複寫伺服器端加密物件 (AWS CLI)

1. 在此範例中，您會在相同的 AWS 帳戶中建立 *example-s3-source-bucket* 和 *example-s3-destination-bucket* 儲存貯體。您也會設定 AWS CLI的憑證描述檔。此範例使用描述檔名稱 *acctA*。

   如需有關設定認證設定檔的詳細資訊，請參閱 AWS Command Line Interface 使用指南中的具名設定檔。若要在此範例中使用命令，請以您的資訊取代 *user input placeholders*。
2. 使用下列命令建立 *DOC-EXAMPLE-SOURCE-BUCKET* 儲存貯體並在其上啟用版本控制。下列範例命令會在美國東部 (維吉尼亞北部) (us-east-1) 區域中建立 *DOC-EXAMPLE-SOURCE-BUCKET* 儲存貯體。

   ```
   aws s3api create-bucket \
   ```

```
--bucket DOC-EXAMPLE-SOURCE-BUCKET \
--region us-east-1 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket DOC-EXAMPLE-SOURCE-BUCKET \
--versioning-configuration Status=Enabled \
--profile acctA
```

3. 使用下列命令建立 *DOC-EXAMPLE-DESTINATION-BUCKET* 儲存貯體並在其上啟用版本控制。下列範例命令會在美國西部 (奧勒岡) (us-west-2) 區域中建立 *DOC-EXAMPLE-DESTINATION-BUCKET* 儲存貯體。

> **Note**
>
> 若要在*DOC-EXAMPLE-SOURCE-BUCKET*和*DOC-EXAMPLE-DESTINATION-BUCKET*值區位於相同的情況下設定複製組態 AWS 帳戶，請使用相同的設定檔。在此範例中，我們使用 *acctA*. 若要在儲存貯體屬於不同 AWS 帳戶時設定複寫，則需為每個儲存貯體指定不同的描述檔。

```
aws s3api create-bucket \
--bucket DOC-EXAMPLE-DESTINATION-BUCKET \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket DOC-EXAMPLE-DESTINATION-BUCKET \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. 接著，請建立 IAM 服務角色。您將在複寫組態中指定稍後要新增至 *DOC-EXAMPLE-SOURCE-BUCKET* 儲存貯體的角色。Amazon S3 就會擔任此角色以代您複寫物件。建立 IAM 角色需要兩個步驟：

   - 建立服務角色。
   - 將許可政策連接到角色。

a. 若要建立 IAM 服務角色，請執行下列動作：

i. 複製下列信任政策，並將它儲存至本機電腦目前目錄下的 s3-role-trust-policy-kmsobj.json 檔案中。此政策會授予 Amazon S3 服務主體擔任該角色的許可，所以 Amazon S3 可以代您執行任務。

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

ii. 使用以下命令來建立角色：

```
$ aws iam create-role \
--role-name replicationRolekmsobj \
--assume-role-policy-document file://s3-role-trust-policy-kmsobj.json  \
--profile acctA
```

b. 接著，請將許可政策連接到角色。此政策會授予各種 Amazon S3 儲存貯體與物件動作的許可。

i. 複製下列許可政策，並將它儲存至本機電腦目前目錄中名為 s3-role-permissions-policykmsobj.json 的檔案。您將建立 IAM 角色，並在稍後將政策連接至該角色。

> **Important**
>
> 在權限原則中，您可以指定將用於*example-s3-source-bucket*和*example-s3-destination-bucket*值區加密的 AWS KMS 金鑰 ID。您必須為*example-s3-source-bucket*和*##* -3 目的地值區建立兩個不同的 KMS 金鑰。 AWS KMS keys 不會在建立它們 AWS 區域 的位置之外共用。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration",
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::example-s3-source-bucket",
            "arn:aws:s3:::example-s3-source-bucket/*"
         ]
      },
      {
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLikeIfExists":{
               "s3:x-amz-server-side-encryption":[
                  "aws:kms",
                  "AES256",
                  "aws:kms:dsse"
               ],
               "s3:x-amz-server-side-encryption-aws-kms-key-id":[
                  "AWS KMS key IDs(in ARN format) to use for encrypting
 object replicas"
               ]
            }
         },
         "Resource":"arn:aws:s3:::example-s3-destination-bucket/*"
      },
      {
         "Action":[
            "kms:Decrypt"
```

```
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.us-east-1.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::example-s3-source-bucket/*"
               ]
            }
         },
         "Resource":[
            "AWS KMS key IDs(in ARN format) used to encrypt source
 objects."
         ]
      },
      {
         "Action":[
            "kms:Encrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.us-west-2.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::example-s3-destination-bucket/*"
               ]
            }
         },
         "Resource":[
            "AWS KMS key IDs(in ARN format) to use for encrypting object
 replicas"
         ]
      }
   ]
}
```

ii. 建立政策，並將政策連接至角色。

```
$ aws iam put-role-policy \
--role-name replicationRolekmsobj \
--policy-document file://s3-role-permissions-policykmsobj.json \
--policy-name replicationRolechangeownerPolicy \
--profile acctA
```

5. 接著，請將下列複寫組態新增至 *example-s3-source-bucket* 儲存貯體。該組態會指示 Amazon S3 使用 *example-s3-destination-bucket* 儲存貯體的 Tax/ 字首複寫物件。

> **Important**
>
> 您可以在複寫組態中指定 Amazon S3 可以擔任的 IAM 角色。只有當您有 `iam:PassRole` 許可時才可執行此作業。您在 CLI 命令中指定的描述檔必須有此許可。如需詳細資訊，請參閱《IAM 使用者指南》中授予使用者將角色傳遞至 AWS 服務的許可。

```
<ReplicationConfiguration>
 <Role>IAM-Role-ARN</Role>
 <Rule>
   <Priority>1</Priority>
   <DeleteMarkerReplication>
      <Status>Disabled</Status>
   </DeleteMarkerReplication>
   <Filter>
      <Prefix>Tax</Prefix>
   </Filter>
   <Status>Enabled</Status>
   <SourceSelectionCriteria>
     <SseKmsEncryptedObjects>
       <Status>Enabled</Status>
     </SseKmsEncryptedObjects>
   </SourceSelectionCriteria>
   <Destination>
     <Bucket>arn:aws:s3:::example-s3-destination-bucket</Bucket>
     <EncryptionConfiguration>
       <ReplicaKmsKeyID>AWS KMS key IDs to use for encrypting object replicas</
ReplicaKmsKeyID>
     </EncryptionConfiguration>
   </Destination>
 </Rule>
</ReplicationConfiguration>
```

請執行下列操作，將複寫組態新增至 *example-s3-source-bucket* 儲存貯體：

a. AWS CLI 需要您將複寫組態指定為 JSON。將下列 JSON 儲存至本機電腦目前目錄中的檔案 (`replication.json`)。

```
{
   "Role":"IAM-Role-ARN",
   "Rules":[
      {
         "Status":"Enabled",
         "Priority":1,
         "DeleteMarkerReplication":{
            "Status":"Disabled"
         },
         "Filter":{
            "Prefix":"Tax"
         },
         "Destination":{
            "Bucket":"arn:aws:s3:::example-s3-destination-bucket",
            "EncryptionConfiguration":{
               "ReplicaKmsKeyID":"AWS KMS key IDs (in ARN format) to use for
 encrypting object replicas"
            }
         },
         "SourceSelectionCriteria":{
            "SseKmsEncryptedObjects":{
               "Status":"Enabled"
            }
         }
      }
   ]
}
```

b. 編輯 JSON 以提供 *example-s3-destination-bucket* 儲存貯體、*AWS KMS key IDs (in ARN format)*、和 *IAM-role-ARN* 的值。儲存變更。

c. 使用下列命令，將複寫組態新增至您的 *example-s3-source-bucket* 儲存貯體。請務必提供 *example-s3-source-bucket* 儲存貯體名稱。

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket example-s3-source-bucket \
--profile acctA
```

6. 測試組態，確認已複寫加密的物件。在 Amazon S3 主控台中，執行下列操作：

a. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
b. 在 *example-s3-source-bucket* 儲存貯體中，建立名為 Tax 的資料夾。
c. 將範例物件新增至資料夾。請務必選擇加密選項，並指定您的 KMS 金鑰來加密物件。
d. 確認 *example-s3-destination-bucket* 儲存貯體包含物件複本，且這些複本已使用您在組態中指定的 KMS 金鑰加密。如需詳細資訊，請參閱 the section called “取得複寫狀態”。

使用 AWS 軟體開發套件

如需新增複寫組態的程式碼範例，請參閱 使用 AWS 軟體開發套件。您必須正確修改複寫組態。

如需相關概念資訊，請參閱 複寫加密的物件 (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)。

使用 Amazon S3 複本修改同步複寫中繼資料變更

Amazon S3 複本修改同步可協助您在複本和來源物件之間複寫物件中繼資料，例如標籤、ACL 和物件鎖定設定。根據預設，Amazon S3 只會將來源物件中的中繼資料複寫至複本。啟用複本修改同步時，Amazon S3 會將對複寫複本所進行的中繼資料變更複寫回來源物件，進而進行雙向複寫。

啟用複本修改同步

您可以將 Amazon S3 複本修改同步與全新或現有的複寫規則搭配使用。您可以將其套用至整個 S3 儲存貯體或具有特定前綴的 Amazon S3 物件。

若要使用 Amazon S3 主控台啟用複本修改同步，請參閱 設定即時複製的範例。本主題提供指示，如何在儲存貯體為相同或不同的值區所擁有時，在複寫組態中啟用複本修改同步 AWS 帳戶。

若要使用 AWS Command Line Interface (AWS CLI) 啟用複本修改同步，您必須將複寫組態新增至包含ReplicaModifications已啟用複本的儲存貯體。*################ (##-s3-bucket1) ############ (## s3-bucket2)##################### (##-s3-bucket2) ####### (## s3-bucket1)#*桶可以是相同的，也可以是在不同的 AWS 區域。

> **Note**
>
> 您必須在兩個儲存貯體上啟用複本修改同步，才能複寫本中繼資料變更，例如複寫物件中的物件存取控制清單 (ACL)、物件標籤或物件鎖定設定。如同所有複寫規則，這些規則可套用至整個 Amazon S3 儲存貯體，也可以套用至由字首或物件標籤篩選的 Amazon S3 物件子集。

在下列範例組態中，Amazon S3 會將前置詞 *Tax* 下的中繼資料變更複寫到儲存貯體*##-s3 #######*含來源物件。

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "SourceSelectionCriteria": {
                "ReplicaModifications":{
                    "Status": "Enabled"
                }
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

如需使用建立複製規則的完整指示 AWS CLI，請參閱為相同帳戶擁有的來源和目的地儲存貯體設定複寫。

複寫儲存貯體之間的刪除標記

根據預設，啟用 S3 複寫且在來源儲存貯體中刪除物件時，Amazon S3 只會在來源儲存貯體中新增刪除標記。此動作可防止資料遭到惡意刪除。

如果啟用了刪除標記複寫，這些標記會複製至目的地儲存貯體，而 Amazon S3 的行為就像在來源和目的地儲存貯體中刪除物件一樣。如需有關刪除標記如何運作的詳細資訊，請參閱 使用刪除標記。

> **Note**
>
> 標籤型複寫規則不支援刪除標記複寫。刪除標記複寫也不會遵守使用 S3 複寫時間控制時所授與的 15 分鐘 SLA。

如果您未使用最新的複寫組態版本，刪除作業會以不同的方式影響複寫。如需詳細資訊，請參閱 刪除操作對複寫的影響。

啟用刪除標記複寫

您可以開始使用新的或現有複寫規則的刪除標記複寫。您可以將其套用至整個 S3 儲存貯體或具有特定前綴的 Amazon S3 物件。

> **Note**
>
> 當您啟用刪除標記複寫且儲存貯體具有 S3 生命週期到期規則時，S3 生命週期到期規則新增的刪除標記不會複寫到目的地儲存貯體。

若要使用 Amazon S3 主控台啟用刪除標記複寫，請參閱 使用 S3 主控台。本主題提供指示，說明如何在儲存貯體屬於相同或不同的值區時，在複寫組態中啟用刪除標記複寫 AWS 帳戶。

若要使用 AWS Command Line Interface (AWS CLI) 啟用刪除標記複寫，您必須在啟用的情`DeleteMarkerReplication`況下將複寫組態新增至來源值區。

在下列範例組態中，刪除標記會複寫至前綴 *`Tax`* 下物件的目的地儲存貯體 *`DOC-EXAMPLE-BUCKET`*。

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "DeleteMarkerReplication": {
                "Status": "Enabled"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

如需透過建立複製規則的完整指示 AWS CLI，請參閱複製逐步解說一節為相同帳戶擁有的來源和目的地儲存貯體設定複寫中的。

管理或暫停即時複寫

即時複製是針對相同或不同值區中物件的自動非同步複製 AWS 區域。設定複寫組態後，Amazon S3 會將新建立的物件和物件更新從來源儲存貯體複寫到一或多個指定的目標儲存貯體。

您可以使用 Amazon S3 主控台，將複寫規則新增至來源儲存貯體。複寫規則定義要複寫的來源儲存貯體物件，以及用以存放複寫物件的目的地儲存貯體。如需複寫的詳細資訊，請參閱 複製物件概觀。

您可以在 Replication (複寫) 頁面中管理複寫規則。您可以新增、檢視、啟用、停用或刪除複製規則。您也可以變更複製規則的優先順序。如需如何新增儲存貯體之複寫規則的資訊，請參閱 使用 S3 主控台。

使用 Amazon S3 主控台管理 S3 儲存貯體的複寫規則

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在「一般用途值區」索引標籤上，選擇您要的值區名稱。
4. 選擇 [管理] 索引標籤，然後向下捲動至 [複寫規則]。
5. 您可以透過下列方式變更複製規則：
   - 若要啟用或停用複製規則，請選擇規則左側的選項按鈕。在 [動作] 功能表上選擇 [啟用規則] 或 [停用規則]。您也可以從 「動作」功能表停用、啟用或刪除值區中的所有規則。

     > **Note**
     >
     > 如果您停用複製規則，之後再重新啟用該規則，則在規則停用時未複製的任何新物件或變更的物件將不會在重新啟用規則時自動複寫。若要複寫這些物件，您必須使用 S3 Batch 複寫。如需詳細資訊，請參閱 the section called “複寫現有物件”。

   - 若要變更規則的優先順序，請選擇規則左側的選項按鈕，然後選擇 [編輯規則]。

     設定規則優先順序，以免多項規則範圍內的物件引發衝突。如果發生規則重疊的情況，Amazon S3 會使用規則優先順序來判斷要套用哪一個規則。數字愈高，優先順序愈高。如需有關規則優先順序的詳細資訊，請參閱 複寫組態。

暫停或停止複製

若要暫時暫停複製並讓複製稍後自動繼續，您可以使用中的動`aws:s3:bucket-pause-replication`作 AWS Fault Injection Service。如需詳細資訊，請參閱AWS Fault Injection Service 使用者指南中的aws:s3:bucket-pause-replication和暫停 S3 複寫。

若要停止 Amazon S3 中的複寫，我們建議您停用複寫規則。如果您停用複製規則，之後再重新啟用該規則，則在規則停用時未複製的任何新物件或變更的物件將不會在重新啟用規則時自動複寫。若要複寫這些物件，您必須使用 S3 Batch 複寫。如需詳細資訊，請參閱 the section called “複寫現有物件”。

如果您移除 AWS Identity and Access Management (IAM) 角色、 AWS Key Management Service (AWS KMS) 許可或儲存貯體政策許可授予 Amazon S3 所需許可的儲存貯體政策許可，則複寫也會停止。不過，我們不建議使用這些方法，因為這些方法會造成複寫失敗。Amazon S3 會將受影響物件的複寫狀態回報為 `FAILED`。如果稍後還原權限，則不會自動複製標記為`FAILED`的物件。若要複寫這些物件，您必須使用 S3 Batch 複寫。

使用複寫指標和 S3 事件通知監控進度

S3 複寫指標提供了複寫組態中複寫規則的詳細指標。使用複寫指標，您可以追蹤擱置的位元組、擱置中的作業、複寫失敗的作業以及複寫延遲來監視 minute-by-minute 進度。

當您啟用 S3 複寫時間控制 (S3 RTC) 時，系統會自動開啟 S3 複寫指標。您也可以在建立或編輯規則時，獨立於 S3 RTC 啟用 S3 複寫指標。S3 RTC 包含其他功能，例如服務水準協議 (SLA) 和遺漏臨界值的通知。如需詳細資訊，請參閱 使用 S3 複寫時間控制 (S3 RTC) 來達到合規要求。

擱置中的位元組、擱置中的操作和複寫延遲指標僅適用於透過 S3 跨區域複寫 (S3 CRR) 或 S3 相同區域複寫 (S3 SRR) 複寫的新物件。複寫失敗操作指標會追蹤使用 S3 CRR 或 S3 SRR 複寫的新物件，以及使用 S3 批次複寫的現有物件。您可以設定 Amazon S3 事件通知以接收複寫失敗事件，以協助疑難排解任何組態問題。

啟用後，S3 複寫指標會將下列指標發佈到 Amazon CloudWatch：

- 擱置複寫的位元組 — 針對指定的複寫規則，擱置複寫的物件位元組總數。
- 複寫延遲 — 針對指定的複寫規則，複寫目的地儲存貯體位於來源儲存貯體後方的秒數上限。
- 擱置複寫的作業 – 針對指定的複寫規則，擱置複寫的作業數量。此指標會追蹤與物件、刪除標記、標籤、存取控制清單 (ACL) 和 S3 物件鎖定相關的操作。
- 複寫失敗的操作 — 針對指定的複寫規則，複寫失敗的操作數量。此指標會追蹤與物件、刪除標記、標籤、ACL 和物件鎖定相關的操作。與其他複寫指標不同，此指標適用於使用 S3 CRR 或 S3 SRR 複寫的新物件，以及使用 S3 批次複寫的現有物件。

> **Note**
>
> 複寫失敗的操作會追蹤每分鐘彙總的 S3 複寫失敗。若要識別複寫失敗的特定物件及其失敗原因，請在 Amazon S3 事件通知中訂閱 `OperationFailedReplication` 事件。如需詳細資訊，請參閱 使用 Amazon S3 事件通知接收複寫失敗事件。

如果任務完全無法執行，則不會將指標傳送至 Amazon CloudWatch。例如，若您沒有執行 S3 批次複寫任務的必要許可，或者複寫組態中的標籤或字首不相符，任務將不會執行。

主題

啟用 S3 複寫指標

您可以開始使用 S3 複寫指標搭配全新或現有的複寫規則。您可以選擇將複寫規則套用至整個 S3 儲存貯體，或套用至具有特定前綴或標籤的 Amazon S3 物件。

本主題提供當來源與目的地儲存貯體為相同或不同的 AWS 帳戶所擁有時，在您的複寫組態中啟用 S3 複寫指標的說明。

若要使用 AWS Command Line Interface (AWS CLI) 啟用複寫量度，您必須在啟用的情`Metrics`況下將複寫組態新增至來源值區。在此範例組態中，字首 *`Tax`* 下的物件會複寫至目的地儲存貯體 *`DOC-EXAMPLE-BUCKET`*，而且會針對這些物件產生指標。

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "Destination": {
```

```
                "Bucket": "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "Metrics": {
                    "Status": "Enabled"
                }
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

如需建立複寫規則的完整說明，請參閱 為相同帳戶擁有的來源和目的地儲存貯體設定複寫。

如需有關在 S3 主控台中檢視複寫指標的詳細資訊，請參閱 使用 Amazon S3 主控台檢視複寫指標。

> **Note**
>
> S3 複寫指標的費率與 Amazon CloudWatch 自訂指標相同。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

使用 Amazon S3 事件通知接收複寫失敗事件

當物件未複寫至其目的地 AWS 區域時，S3 事件通知可以在執行個體中通知您。Amazon S3 事件可透過 Amazon Simple Queue Service (Amazon SQS)、亞馬遜簡單通知服務 (Amazon SNS) 或者 AWS Lambda取得。如需詳細資訊，請參閱 the section called “Amazon S3 事件通知”。

如需 S3 事件通知擷取的失敗代碼清單，請參閱 Amazon S3 複寫失敗原因。

使用 S3 儲存鏡頭檢視複寫指標

若要取得 S3 複寫的詳細指標 (包括複寫規則計數指標)，您可以使用 Amazon S3 Storage Lens。S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。如需詳細資訊，請參閱使用 S3 Storage Lens 保護資料。如需指標的完整清單，請參閱 S3 儲存鏡頭指標詞彙表。

使用 Amazon S3 主控台檢視複寫指標

Amazon Amazon S3 有三種類型的 Amazon CloudWatch 指標：儲存指標、請求指標和複寫指標。當您使用 AWS Management Console 或 Amazon S3 API 啟用具有 S3 複寫時間控制 (S3 RTC) 的複寫時，會自動開啟 S3 複寫指標。您也可以在建立或編輯規則時，獨立於 S3 RTC 啟用 S3 複寫指標。

複寫指標會追蹤複寫組態的規則 ID。複寫規則 ID 可以是字首、標籤或兩者的組合。

如需 Amazon S3 CloudWatch 指標的詳細資訊，請參閱使用 Amazon 監控指標 CloudWatch。

必要條件

啟用具有 S3 複寫指標的複寫規則。

檢視複寫指標

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。在儲存貯體清單中，選擇包含您要複寫指標之物件的儲存貯體名稱。
3. 選擇指標標籤。
4. 在 Replication metrics (複寫指標) 下方選擇 Replication rules (複製規則)。
5. 選擇 Display charts (顯示圖表)。

   Amazon S3 會顯示複寫延遲 (以秒為單位)、擱置複寫的位元組、擱置複寫的操作，以及複寫失敗的操作圖表。

然後，您就可以檢視所選取規則的複寫延遲 (以秒為單位)、擱置複寫的操作，以及擱置複寫的位元組，以及複寫失敗的操作等複寫指標。如果您使用 S3 複寫時間控制，Amazon 會在您針對個別複寫規則啟用 S3 RTC 後 15 分鐘 CloudWatch 開始報告複寫指標。您可以在 Amazon S3 主控台或主控台上檢視複寫指標。 CloudWatch 如需詳細資訊，請參閱 使用 S3 RTC 的複寫指標。

> **Note**
>
> 您也可以使用 Amazon S3 儲存鏡頭在 Amazon S3 主控台中檢視 S3 複寫的詳細指標。S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。如需詳細資訊，請參閱使用 S3 Storage Lens 保護資料。如需指標的完整清單，請參閱 S3 儲存鏡頭指標詞彙表。

Amazon S3 複寫失敗原因

下表列出 Amazon S3 複寫失敗原因。您可以透過 Amazon S3 事件通知接收 failureReason 事件，檢視這些原因。您可以透過 Amazon Simple Queue Service (Amazon SQS)、亞馬遜簡易通知服務 (Amazon SNS) 或 AWS Lambda接收 S3 事件通知。如需詳細資訊，請參閱 Amazon S3 事件通知。

您也可以在 S3 批次複寫完成報告中檢視這些失敗原因。如需詳細資訊，請參閱 批次複寫完成報告。

複寫失敗原因	描述
AssumeRoleNotPermitted	Amazon S3 無法承擔在複寫組態或 Batch 操作任務中指定的 AWS Identity and Access Management (IAM) 角色。
DstBucketInvalidRegion	目的地時段與「Batch 作業」工單所指定的不同 AWS 區域 。此錯誤僅是批次複寫特有的。
DstBucketNotFound	Amazon S3 找不到複寫組態中指定的目的地儲存貯體。
DstBucketObjectLockConfigMissing	若要從啟用物件鎖定的來源儲存貯體複寫物件，目的地儲存貯體亦須啟用物件鎖定。此錯誤指出目的地儲存貯體中可能未啟用物件鎖定。如需詳細資訊，請參閱 物件鎖定的考量事項。
DstBucketUnversioned	S3 目的地儲存貯體未啟用版本控制。在目的地儲存貯體上啟用版本控制，以使用 S3 複寫來複寫物件。
DstDelObjNotPermitted	Amazon S3 無法將刪除標記複寫到目的地儲存貯體。可能缺少目的地儲存貯體的 s3:ReplicateDelete 許可。
DstKmsKeyInvalidState	目的地值區的 AWS Key Management Service (AWS KMS) 金鑰不是有效的狀態。檢閱並啟用所需的 AWS KMS 金鑰。如需有關管理 AWS KMS 金鑰的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的 AWS KMS 金鑰金鑰狀態。
DstKmsKeyNotFound	複寫組態中針對目的地儲存貯體所設定的 AWS KMS 金鑰不存在。

複寫失敗原因	描述
DstMultipartCompleteNotPermitted	Amazon S3 無法完成目標儲存貯體中物件的分段上傳。可能缺少目的地儲存貯體的 s3:ReplicateObject 許可。
DstMultipartInitNotPermitted	Amazon S3 無法對目的地儲存貯體啟動物件的分段上傳。可能缺少目的地儲存貯體的 s3:ReplicateObject 許可。
DstMultipartPartUploadNotPermitted	Amazon S3 無法在目的地儲存貯體中上傳分段物件。可能缺少目的地儲存貯體的 s3:ReplicateObject 許可。
DstObjectHardDeleted	刪除的物件若具有來自目的地儲存貯體之物件的版本 ID，S3 批次複寫不支援重新複寫這些物件。此錯誤僅是批次複寫特有的。
DstPutAclNotPermitted	Amazon S3 無法將物件存取控制清單 (ACL) 複寫到目的地儲存貯體。可能缺少目的地儲存貯體的 s3:ReplicateObject 許可。
DstPutLegalHoldNotPermitted	在複寫不可變物件時，Amazon S3 無法對目的地物件放置物件鎖定合法保留。可能缺少目的地儲存貯體的 s3:PutObjectLegalHold 許可。如需詳細資訊，請參閱 法務保存。
DstPutObjectNotPermitted	Amazon S3 無法將物件複寫到目的地儲存貯體。可能缺少目的地儲存貯體的 s3:ReplicateObject 或 s3:ObjectOwnerOverrideToBucketOwner 許可。
DstPutTaggingNotPermitted	Amazon S3 無法將物件標籤複寫到目的地儲存貯體。可能缺少目的地儲存貯體的 s3:ReplicateObject 許可。

複寫失敗原因	描述
DstVersionNotFound	Amazon S3 無法在需要複寫中繼資料的目的地儲存貯體中找到所需的物件版本。
InitiateReplicationNotPermitted	Amazon S3 無法啟動物件上的複寫。可能缺少批次操作任務的 s3:InitiateReplication 許可。此錯誤僅是批次複寫特有的。
SrcBucketInvalidRegion	來源時段與「Batch 作業」工單所指定的不同 AWS 區域 。此錯誤僅是批次複寫特有的。
SrcBucketNotFound	Amazon S3 無法找到來源儲存貯體。
SrcBucketReplicationConfigMissing	Amazon S3 找不到來源儲存貯體的複寫組態。
SrcGetAclNotPermitted	Amazon S3 無法存取來源儲存貯體中的物件以進行複寫。可能缺少來源儲存貯體物件的 s3:GetObjectVersionAcl 許可。 來源儲存貯體中的物件必須為儲存貯體擁有者所擁有。如果已啟用 ACL，請確認「物件擁有權」設定為「偏好的儲存貯體擁有者」或「物件寫入器」。如果將「物件擁有權」設定為「偏好的儲存貯體擁有者」，則來源儲存貯體物件必須有 bucket-owner-full-control ACL，儲存貯體擁有者才能成為物件擁有者。透過將「物件擁有權」設定為「強制執行的儲存貯體擁有者」並停用 ACL，來源帳戶就能取得其儲存貯體中所有物件的擁有權。
SrcGetLegalHoldNotPermitted	Amazon S3 無法存取 S3 物件鎖定法務保存資訊。

複寫失敗原因	描述
SrcGetObjectNotPermitted	Amazon S3 無法存取來源儲存貯體中的物件以進行複寫。可能缺少來源儲存貯體的 s3:GetObjectVersionForReplication 許可。
SrcGetRetentionNotPermitted	Amazon S3 無法存取 S3 物件鎖定保留期間資訊。
SrcGetTaggingNotPermitted	Amazon S3 無法從來源儲存貯體存取物件標籤資訊。可能缺少來源儲存貯體的 s3:GetObjectVersionTagging 許可。
SrcHeadObjectNotPermitted	Amazon S3 無法從來源儲存貯體擷取物件中繼資料。可能缺少來源儲存貯體的 s3:GetObjectVersionForReplication 許可。
SrcKeyNotFound	Amazon S3 找不到要複寫的來源物件金鑰。來源物件可能已在複寫完成之前遭到刪除。
SrcKmsKeyInvalidState	來源儲存貯體的 AWS KMS 金鑰不是有效的狀態。檢閱並啟用所需的 AWS KMS 金鑰。如需有關管理 AWS KMS 金鑰的詳細資訊，請參閱AWS Key Management Service 開發人員指南中的 AWS KMS 金鑰金鑰狀態。
SrcObjectNotEligible	有些物件不符合複寫資格。這可能是因為物件的儲存類別或物件標籤與複寫組態不相符。
SrcObjectNotFound	來源物件不存在。
SrcReplicationNotPending	Amazon S3 已複寫此物件。此物件不再處於待複寫狀態。
SrcVersionNotFound	Amazon S3 找不到要複寫的來源物件版本。來源物件版本可能已在複寫完成之前遭到刪除。

相關主題

[設定即時複製的權限]

[故障排除複寫]

取得複寫狀態資訊

複寫狀態可協助您判斷要複寫之物件的目前狀態。來源物件的複寫狀態將會傳回 PENDING、COMPLETED 或 FAILED。複本的複寫狀態將會傳回 REPLICA。

主題

- [複寫狀態概觀]
- [複寫至多個目的地儲存貯體時的複寫狀態]
- [啟用 Amazon S3 複本修改同步時的複寫狀態]
- [尋找複寫狀態]

複寫狀態概觀

在複寫中，您有一個來源儲存貯體，您可以在其上設定複寫與 Amazon S3 複寫物件的目的地。當您從這些儲存貯體請求物件 (使用 GET 物件) 或物件中繼資料 (使用 HEAD 物件) 時，Amazon S3 會在回應中傳回 x-amz-replication-status 標頭：

- 當您從來源儲存貯體請求物件時，如果請求中的物件符合複寫資格，Amazon S3 即會傳回 x-amz-replication-status 標頭。

  例如，假設您在複寫組態中指定物件前綴 TaxDocs，告知 Amazon S3 複寫僅具有金鑰名稱前綴 TaxDocs 的物件。系統會複寫您上傳且具有此金鑰名稱前綴 (例如 TaxDocs/document1.pdf) 的任何物件。針對具有此金鑰名稱前綴的物件請求，Amazon S3 會傳回 x-amz-replication-status 標頭，以及代表物件複寫狀態的下列其中一個值：PENDING、COMPLETED 或 FAILED。

  > **Note**
  >
  > 若物件複寫在您上傳完物件後失敗，則您無法重試複寫。您必須再次上傳物件。對於缺少複寫角色許可、 AWS KMS 許可，或儲存貯體許可等問題，物件會轉換成 FAILED 狀態。對於暫時性錯誤，例如，如果儲存貯體或區域無法使用，複寫狀態將不會轉換成 FAILED，但會保持 PENDING。資源恢復線上狀態後，S3 將繼續複寫這些物件。

- 當您從目的地儲存貯體請求物件時，如果您請求中的物件是 Amazon S3 建立的複本，Amazon S3 會傳回值為 REPLICA 的 x-amz-replication-status 標頭。

> **Note**
>
> 在從已啟用複寫的來源儲存貯體中刪除物件之前，您應該先檢查物件的複寫狀態，確保已複寫物件。
> 如果已啟用來源儲存貯體上的生命週期組態，Amazon S3 會暫停生命週期動作，直到將物件狀態標示為 COMPLETED 或 FAILED 為止。

複寫至多個目的地儲存貯體時的複寫狀態

當您將物件複寫至多個目的地儲存貯體時，x-amz-replication-status 標頭的運作方式會有所不同。成功複寫至所有目的地時，來源物件的標頭只會傳回 COMPLETED 的值。標頭會保留在 PENDING 值，直到對所有目的地的複寫完成為止。如果一或多個目的地複寫失敗，標頭會傳回 FAILED。

啟用 Amazon S3 複本修改同步時的複寫狀態

當您的複寫規則啟用 Amazon S3 複本修改同步時，複本可以報告 REPLICA 以外的狀態。如果中繼資料變更正在複寫過程中，則 x-amz-replication-status 標頭會傳回 PENDING。如果複本修改同步無法複寫中繼資料，則標頭會傳回 FAILED。如果中繼資料正確複寫，複本將會傳回標頭 REPLICA。

尋找複寫狀態

若要取得儲存貯體中物件的複寫狀態，您可使用 Amazon S3 庫存工具。Amazon S3 會將 CSV 檔案傳送到您在庫存組態中指定的目的地儲存貯體。您也可以使用 Amazon Athena，來查詢庫存報告中的複寫狀態。如需 Amazon S3 庫存的詳細資訊，請參閱 Amazon S3 清查。

您也可以使用主控台、 AWS Command Line Interface (AWS CLI) 或 AWS SDK 尋找物件複寫狀態。

使用 S3 主控台

在 S3 主控台中，您可以在 Object management overview (物件管理概觀) 下的物件 Details (詳細資訊) 頁面上檢視物件的複寫狀態。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

2. 在 Buckets (儲存貯體) 清單中，選擇您的儲存貯體名稱。
3. 在 Objects (物件) 清單中，選擇物件名稱。
4. 在 Properties (屬性) 下尋找 Object management overview (物件管理概觀)，您可以在這裡看到 Replication status (複寫狀態)。

使用 AWS CLI

使用 `head-object` 命令來擷取物件中繼資料，如下所示。

```
aws s3api head-object --bucket source-bucket --key object-key --version-id object-version-id
```

此命令會傳回物件中繼資料，包括 `ReplicationStatus`，如下列範例回應所示。

```
{
   "AcceptRanges":"bytes",
   "ContentType":"image/jpeg",
   "LastModified":"Mon, 23 Mar 2015 21:02:29 GMT",
   "ContentLength":3191,
   "ReplicationStatus":"COMPLETED",
   "VersionId":"jfnW.HIMOfYiD_9rGbSkmroXsFj3fqZ.",
   "ETag":"\"6805f2cfc46c0f04559748bb039d69ae\"",
   "Metadata":{

   }
}
```

使用 AWS 軟體開發套件

下列程式碼片段會分別取得與 AWS SDK for Java 和 AWS SDK for .NET的複寫狀態。

Java

```
GetObjectMetadataRequest metadataRequest = new GetObjectMetadataRequest(bucketName,
 key);
ObjectMetadata metadata = s3Client.getObjectMetadata(metadataRequest);

System.out.println("Replication Status : " +
 metadata.getRawMetadataValue(Headers.OBJECT_REPLICATION_STATUS));
```

.NET

```
GetObjectMetadataRequest getmetadataRequest = new GetObjectMetadataRequest
    {
         BucketName = sourceBucket,
         Key        = objectKey
    };

GetObjectMetadataResponse getmetadataResponse =
 client.GetObjectMetadata(getmetadataRequest);
Console.WriteLine("Object replication status: {0}",
 getmetadataResponse.ReplicationStatus);
```

使用 S3 批次複寫來複寫現有物件

透過使用 S3 Batch 複寫，您可以複寫下列類型的物件：

- 在複製組態到位之前就已存在的物件
- 先前已複製的物件
- 複寫失敗的物件

您可以使用「Batch 作業」工作視需求複製這些物件。S3 Batch 複寫與即時複寫不同，即時複寫會持續且自動地跨 Amazon S3 儲存貯體複寫新物件。

若要開始使用 Batch 複寫，您可以：

- 為新的複寫規則或目的地啟動 Batch 複寫 — 當您在新複寫組態中建立第一個規則，或透過 Amazon S3 主控台將新目標新增至現有組態時，可以建立一次性 Batch 複寫任務。
- 啟動現有複寫組態的 Batch 複寫 — 您可以透過 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、開 AWS 發套件或 Amazon S3 REST API 使用 S3 Batch 操作來建立新的 Batch 複寫任務。

當批次複寫任務完成時，您將收到一份完成報告。如需使用報告檢查任務的詳細資訊，請參閱追蹤任務狀態和完成報告。

S3 批次複寫注意事項

- 來源儲存貯體必須具有現有的複寫組態。若要啟用複寫，請參閱設定即時複製和設定即時複製的範例。
- 如果您已為儲存貯體設定 S3 生命週期，建議您在 Batch 複寫任務處於作用中狀態時停用生命週期規則。這樣做有助於確保來源值區和目標值區之間的同位檢查。否則，這些值區可能會發生分歧，而且目標值區將不會是來源值區的精確複本。例如，考量以下情境：
  - 您的來源值區具有物件的多個版本，以及該物件上的刪除標記。
  - 您的來源和目的地儲存貯體具有生命週期組態，可移除過期的刪除標記。

  在這個案例中，Batch 複寫可能會先將刪除標記複寫到目的地儲存貯體，再複寫物件版本。此行為可能會導致您的生命週期組態將刪除標記標記標記為已過期，並在複製物件版本之前從目的地值區移除刪除標記。
- 您指定用來執行 Batch 作業工作的 AWS Identity and Access Management (IAM) 角色必須具有執行基礎 Batch 複寫作業的必要權限。如需建立 IAM 角色的詳細資訊，請參閱 設定批次複寫的 IAM 政策。
- Batch 複寫需要可由 Amazon S3 產生的資訊清單。產生的資訊清單必須儲存在與來源儲存貯體 AWS 區域 相同的位置。如果您選擇不產生資訊清單，您可以提供包含要複寫之物件的 Amazon S3 庫存報告或 CSV 檔案。
- Batch 複寫不支援使用目的地值區中物件的版本識別碼刪除的重新複寫物件。若要重新複寫這些物件，您可以使用批次複製 (Batch Copy) 任務將來源物件複製到位。就地複製這些物件會在來源值區中建立物件的新版本，並自動啟動目的地值區的複寫。刪除和重新建立目的地值區並不會起始複寫。

  如需「Batch 複製」的更多資訊，請參閱使用批次操作複製物件的範例。
- 如果您在 S3 儲存貯體上使用複寫規則，請確保透過授與附加到複寫規則的 IAM 角色具有複寫物件的適當許可來更新複寫組態。此 IAM 角色必須具備必要的許可，才能在來源和目的地儲存貯體上執行複寫。
- 如果您在短時間內為同一儲存貯體提交多個 Batch 複寫任務，Amazon S3 將同時執行這些任務。
- 如果您為兩個不同的儲存貯體提交多個 Batch 複寫任務，請注意，Amazon S3 可能無法同時執行所有任務。如果您超過帳戶一次可以執行的 Batch 複寫任務數量，Amazon S3 將暫停較低優先順序的任務，以處理優先順序較高的任務。在優先順序較高的項目完成之後，所有暫停的工作都會再次變為作用中狀態。
- 存放在 S3 Glacier 彈性擷取和 S3 Glacier 深度存檔儲存類別中的物件不支援 Batch 複寫。
- 若要批次複寫存放在封存存取或深度封存存取儲存層中的 S3 智慧型分層物件，您必須先啟動還原請求，然後等待物件移至頻繁存取層。

指定批次複寫任務的資訊清單

清單檔案是 Amazon S3 物件，其中包含您希望 Amazon S3 採取行動的物件索引鍵。如果要建立 Batch 複寫任務，則必須提供使用者產生的資訊清單，或讓 Amazon S3 根據複寫組態產生資訊清單。

如果您提供使用者產生的資訊清單，資訊清單必須採用 Amazon S3 庫存報告或 CSV 檔案的形式。若您資訊清單中的物件位於啟用版本控制的儲存貯體，您必須指定物件的版本 ID。只有具有在資訊清單中指定的版本 ID 的物件才會被複寫。若要進一步了解如何指定資訊清單，請參閱指定資訊清單。

如果您選擇讓 Amazon S3 代表您產生資訊清單檔案，列出的物件將使用與來源儲存貯體所有複寫組態相同的來源儲存貯體、前綴和標籤。Amazon S3 會使用產生的資訊清單複寫物件的所有合格版本。

> **Note**
>
> 如果您選擇讓 Amazon S3 產生資訊清單，資訊清單必須存放在與來源儲存貯體 AWS 區域 相同的位置。

批次複寫任務的篩選條件

建立 Batch 複製工作時，您可以選擇性地指定其他篩選器，例如物件建立日期和複寫狀態，以減少工作的範圍。

您可以根據 `ObjectReplicationStatuses` 值來篩選要複寫的物件，方法是提供下列一或多個值：

- `"NONE"` – 表示 Amazon S3 之前從未嘗試複寫該物件。
- `"FAILED"`— 表示 Amazon S3 之前已嘗試複寫物件，但失敗。
- `"COMPLETED"` – 表示 Amazon S3 之前已成功複寫該物件。
- `"REPLICA"`— 指示這是 Amazon S3 已從其他來源複寫的複本物件。

如需複寫狀態的詳細資訊，請參閱取得複寫狀態資訊。

如果您未篩選「Batch 複寫」工作，Batch 作業會嘗試複寫資訊清單中符合複寫組態中規則的所有物件 (不論它們`ObjectReplicationStatus`)，但預設情況下未複寫的某些物件除外。如需更多資訊，請參閱the section called “使用複寫組態不會複寫什麼項目？”

視您的目標而定，您可能會設定`ObjectReplicationStatuses`下列一或多個值：

- 若只要複製從未複製的現有物件，請僅包含`"NONE"`。

- 若要僅重試複製先前失敗的物件，請僅包含"FAILED"。
- 若要複製現有物件並重試複製之前失敗的物件，請同時包含"NONE"和"FAILED"。
- 若要使用已複製到其他目的地的物件回填目的地值區，請包括"COMPLETED"。
- 若要複製先前複製的物件，請包括"REPLICA"。

批次複寫完成報告

當您建立批次複寫任務時，您可以請求 CSV 完成報告。此報告會顯示物件、複寫成功或失敗代碼、輸出，以及描述。如需工作追蹤和完成報告的詳細資訊，請參閱完成報告。

如需複製失敗代碼和說明的清單，請參閱Amazon S3 複寫失敗原因。

如需有關疑難排解 Batch 複寫的資訊，請參閱 批次複寫錯誤

開始使用批次複寫

若要進一步了解如何使用批次複寫，請參閱教學：使用 S3 批次複寫複寫 Amazon S3 儲存貯體中的現有物件。

設定批次複寫的 IAM 政策

因為 S3 批次複寫是一種批次操作任務，因此您必須建立批次操作 AWS Identity and Access Management (IAM) 角色以授予 Amazon S3 代您執行動作的許可。您還必須將批次複寫 IAM 政策連接到批次操作 IAM 角色。以下範例建立了一個 IAM 角色，該角色授予批次操作啟動批次複寫任務的許可。

建立 IAM 角色和政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 Access management (存取管理) 下，請選擇 Roles (角色)。
3. 選擇建立角色。
4. 選擇 AWS 服務 做為信任實體類型、Amazon S3 做為服務、S3 Batch Operations (S3 批次操作) 做為使用案例。
5. 選擇 Next: Permissions (下一步：許可)。

6. 選擇 Create Policy (建立政策)。
7. 選擇 JSON 並根據您的資訊清單插入以下政策之一。

> **Note**
>
> 如果您要產生資訊清單或提供資訊清單，則需要不同的許可。若要取得更多資訊，請參閱指定批次複寫任務的資訊清單。

使用和存放 S3 產生的資訊清單時的政策

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
         ]
      },
      {
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:PutInventoryConfiguration"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***"
         ]
      },
      {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** manifest bucket ***/*"
         ]
```

```
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::*** completion report bucket ****/*",
            "arn:aws:s3:::*** manifest bucket ****/*"
         ]
      }
   ]
}
```

使用使用者提供的資訊清單時的政策

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
         ]
      },
            {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** manifest bucket ***/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
```

```
            "Resource":[
               "arn:aws:s3:::*** completion report bucket ****/*"
            ]
         }
      ]
}
```

8. 選擇下一步：標籤。
9. 選擇下一步：檢閱。
10. 選擇政策的名稱，然後選擇 Create policy (建立政策)。
11. 將此政策連接到您的角色，然後選擇 Next: Tags (下一步：標籤)。
12. 選擇下一步：檢閱。
13. 選擇角色的名稱，然後選擇 Create role (建立角色)。

驗證信任政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 Access management (存取管理) 下，選擇 Roles (角色)，然後選取您新建立的角色。
3. 在 Trust relationships (信任關係) 標籤下，選擇 Edit trust relationship (編輯信任關係)。
4. 驗證此角色是否使用下列信任政策：

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

為第一個複寫規則或新目的地建立批次複寫任務

當您在新的複製組態中建立第一個規則，或透過新增目的地至現有組態時 AWS Management Console，您可以選擇性地建立 Batch 複製工作。

如要將「批次複寫」用於不新增目的地的現有組態，請參閱 為現有複寫規則建立批次複寫任務。

針對新複製規則或目的地使用 Batch 複製 AWS Management Console

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇儲存貯體的名稱，其中包含您要複寫的物件。
3. 如要建立新的複寫規則或編輯現有規則，請選擇 Management (管理)，然後向下捲動至 Replication rules (複寫規則)：

   - 如要建立新的複寫規則，請選擇 Create replication rule (建立複寫規則)。

     > **Note**
     >
     > 如需如何設定基本複寫規則的範例，請參閱設定即時複製的範例。

   - 如要編輯現有的複寫規則，請選取該規則，然後選擇 Edit rule (編輯規則)。
4. 建立新的複寫規則或編輯現有複寫規則的目的地，然後選擇 Save (儲存)。

   在您於新的複寫組態中建立第一個規則後，或編輯現有組態以新增目的地之後，Replicate existing objects? (複寫現有物件嗎？) 對話方塊隨即顯示，為您提供了建立「批次複寫」任務的選項。
5. 如果您要立即執行此任務，請選擇是，複製現有物件。

   或者，如果您想要稍後執行此任務，請選擇否，不複寫現有物件。
6. 建立「S3 批次複寫」任務。S3 批次複寫任務具有多個設定：

   任務執行選項

   若您希望「S3 批次複寫」任務立即執行，您可選擇 Job runs automatically when ready (任務準備就緒時自動執行)。若要於稍後執行任務，請選擇 Job waits to be run when ready (任務準備就緒時等待執行)。

   如果選擇 Job runs automatically when ready (任務準備就緒時自動執行)，則無法建立和儲存批次操作資訊清單。若要儲存批次操作資訊清單，請選擇 Job waits to be run when ready (任務準備就緒時等待執行)。

批次操作資訊清單

資訊清單是您希望對其執行指定動作之所有物件的清單。您可以選擇儲存批次操作資訊清單。與 S3 庫存檔案類似，資訊清單將儲存為 CSV 檔案並存放在儲存貯體中。若要進一步了解批次操作資訊清單，請參閱指定資訊清單。

完成報告

S3 批次操作會為資訊清單中指定的每個物件執行一個任務。完成報告可讓您以合併形式輕鬆檢視任務結果，無需進行額外設定。您可以請求所有任務或僅失敗任務的完成報告。若要進一步了解完成報告，請參閱完成報告。

許可

複寫失敗的最常見原因之一是提供的 AWS Identity and Access Management (IAM) 角色中的許可不足。如需建立此角色的詳細資訊，請參閱設定批次複寫的 IAM 政策。

7. 選擇 Create Batch Operations job (建立批次操作任務)。

為現有複寫規則建立批次複寫任務

您可以使用 AWS 開發套件、 AWS Command Line Interface (AWS CLI) 或 Amazon S3 主控台為現有複寫組態設定 S3 Batch 複寫。如需批次複寫的概觀，請參閱使用 S3 批次複寫來複寫現有物件。

先決條件是，您必須建立 Batch 操作 AWS Identity and Access Management (IAM) 角色，以授與 Amazon S3 許可以代表您執行動作，請參閱設定批次複寫的 IAM 政策。

當批次複寫任務完成時，您將收到一份完成報告。如需使用報告檢查任務的詳細資訊，請參閱追蹤任務狀態和完成報告。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Amazon S3 主控台的導覽窗格上，選擇 Batch Operations (批次操作)。
3. 選擇 Create job (建立任務)。
4. 選擇要在其中建立任務的 Region (區域)。
5. 選取 Manifest format (資訊清單格式)。此範例將展示如何根據現有 S3 複寫組態來建立資訊清單。

> **Note**
>
> 資訊清單是您希望對其執行指定動作之所有物件的清單。若要進一步了解批次操作資訊清單，請參閱指定資訊清單。如果您已準備好資訊清單，請選擇 S3 inventory report (manifest.json) (S3 庫存報告 (manifest.json)) 或 CSV。若您資訊清單中的物件位於啟用版本控制的儲存貯體，建議您指定物件的版本 ID。如需建立資訊清單的詳細資訊，請參閱指定資訊清單。

6. 若要根據複寫組態建立資訊清單，請選擇 Create manifest using S3 Replication configuration (使用 S3 複寫組態建立資訊清單)。接著選擇複寫組態的來源儲存貯體。
7. (選用) 您可以包含其他篩選條件，例如物件建立日期和複寫狀態。如需如何依照複寫狀態篩選的範例，請參閱指定批次複寫任務的資訊清單。
8. 若要儲存資訊清單，請選取 Save Batch Operations manifest (儲存批次操作資訊清單)。

   a. 如果選擇產生和儲存清單檔案，則必須選擇 Bucket in this account (此帳戶中的儲存貯體) 或者 Bucket in another AWS 帳戶 (另一個 AWS 帳戶中的儲存貯體)。在文字方塊中指定儲存貯體名稱。

   > **Note**
   >
   > 產生的資訊清單必須儲存在與來源儲存貯體 AWS 區域 相同的位置。

   b. 選擇加密類型。
9. (選用) 提供 Description (描述)。
10. 視需要調整任務的 Priority (優先順序)。數字愈大表示優先順序愈高。Amazon S3 會嘗試在優先順序較低的任務之前，先執行優先順序較高的任務。如需任務優先順序的詳細資訊，請參閱 指派任務優先順序。
11. (選用) 產生完成報告。若要產生，請選取 Generate completion report (產生完成報告)。

    如果選擇產生完成報告，則必須選擇報告 Failed tasks only (僅限失敗的任務) 或者 All tasks (所有任務)，並為報告提供目的地儲存貯體。
12. 選取有效的 IAM 角色。

> **Note**
>
> 如需建立 IAM 角色的詳細資訊，請參閱設定批次複寫的 IAM 政策。

13. (選用) 將任務標籤新增到批次複寫任務。
14. 選擇 Next (下一步)。
15. 檢閱您的組態，然後選取 Create job (建立任務)。

AWS CLI 搭配 S3 資訊清單使用

下列範例使用 S3 產生的資訊清單建立 S3 Batch 複寫任務 AWS 帳戶 *111122223333*。此範例將嘗試複寫現有物件和先前複寫失敗的物件。如需依照複寫狀態篩選的相關資訊，請參閱[指定批次複寫任務的資訊清單](#)。

```
aws s3control create-job --account-id 111122223333 --operation
 '{"S3ReplicateObject":{}}' --report '{"Bucket":"arn:aws:s3:::***
 completion report bucket ****","Prefix":"batch-replication-report",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}'
 --manifest-generator '{"S3JobManifestGenerator": {"ExpectedBucketOwner":
 "111122223333", "SourceBucket": "arn:aws:s3:::*** replication source bucket
 ***", "EnableManifestOutput": false, "Filter": {"EligibleForReplication": true,
 "ObjectReplicationStatuses": ["NONE","FAILED"]}}}' --priority 1 --role-arn
 arn:aws:iam::111122223333:role/batch-Replication-IAM-policy --no-confirmation-required
 --region source-bucket-region
```

> **Note**
>
> 工作必須從相同的 AWS 區域 複寫來源儲存貯體啟動。IAM 角色 `role/batch-Replication-IAM-policy` 之前已建立。請參閱[設定批次複寫的 IAM 政策](#)。

成功啟動批次複寫任務後，您將收到任務 ID 做為回應。您可使用下列命令來監控此任務。

```
aws s3control describe-job --account-id 111122223333 --job-id job-id --region source-
bucket-region
```

AWS CLI 搭配使用者提供的資訊清單使用

下列範例會使用使用者為 AWS 帳戶 *111122223333* 定義的資訊清單，建立 S3 批次複寫任務。若您資訊清單中的物件位於啟用版本控制的儲存貯體，您必須指定物件的版本 ID。只有在清單檔案中指定版本 ID 的物件，才會被複寫。如需建立資訊清單的詳細資訊，請參閱[指定資訊清單](#)。

```
aws s3control create-job --account-id 111122223333 --operation
 '{"S3ReplicateObject":{}}' --report '{"Bucket":"arn:aws:s3:::***
 completion report bucket ****","Prefix":"batch-replication-report",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}'
 --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key","VersionId"]},"Location":{"ObjectArn":"arn:aws:s3:::*** completion
 report bucket ****/manifest.csv","ETag":"Manifest Etag"}}' --priority 1 --role-arn
 arn:aws:iam::111122223333:role/batch-Replication-IAM-policy --no-confirmation-required
 --region source-bucket-region
```

> **Note**
>
> 工作必須從相同的 AWS 區域 複寫來源儲存貯體啟動。IAM 角色 `role/batch-Replication-IAM-policy` 之前已建立。請參閱設定批次複寫的 IAM 政策。

成功啟動批次複寫任務後，您將收到任務 ID 做為回應。您可使用下列命令來監控此任務。

```
aws s3control describe-job --account-id 111122223333 --job-id job-id --region source-
bucket-region
```

使用標籤分類儲存空間

使用物件標記，可讓您分類儲存。每個標籤都是金鑰值對。

您可以在上傳新的物件時，為這些物件新增標籤，也可以對現有的物件新增標籤。

- 一個物件最多可以與 10 個標籤相關聯。與物件相關聯的標籤，必須具備唯一的標籤金鑰。
- 標籤金鑰最長可包含 128 個 Unicode 字元，標籤值最長可包含 256 個 Unicode 字元。Amazon S3 物件標籤在 UTF-16 內部表示。請注意，在 UTF-16 中，字元佔用 1 或 2 個字元位置。
- 金鑰與值皆會區分大小寫。
- 如需有關標籤限制的詳細資訊，請參閱 B AWS illing and Cost Management 使用者指南中的使用者定義標籤限制。如需基本標籤限制，請參閱 Amazon EC2 使用者指南中的標籤限制。

範例

請考量下列標記範例：

Example PHI 資訊

假設物件包含受保護的醫療資訊 (PHI) 資料。您可以使用下列金鑰/值對標記物件。

```
PHI=True
```

或

```
Classification=PHI
```

Example 專案檔案

假設您將該專案檔存放在您的 S3 儲存貯體中。您可以使用名稱為 Project 的金鑰與值來標記這些物件。

```
Project=Blue
```

Example 多個標籤

您可以為物件新增多個標籤，如下所示。

```
Project=x
Classification=confidential
```

金鑰名稱前綴和標籤

物件金鑰的前綴名稱可讓您分類儲存體。不過，以前綴為基礎的分類是單一維度的。請考量下列物件金鑰名稱：

```
photos/photo1.jpg
project/projectx/document.pdf
project/projecty/document2.pdf
```

這些金鑰名稱使用下列前綴：photos/、project/projectx/ 及 project/projecty/。您可以將這些前綴應用在特定分類。亦即，某前綴下的所有項目均屬於同一個類別。例如，前綴 project/projectx 即表示所有與專案 x 相關的文件。

有了標記功能之後，現在可有不同的區分方式。若希望將 photo1 歸入專案 x 類別，可以依此將該物件加上標記。

其他優勢

除了資料分類之外，標記功能另有其他優點，如下所示：

- 物件標籤可以更精細地分級許可存取控制。例如，您可以將許可授權給一位使用者，只允許其讀取具有特定標籤的物件。
- 除了金鑰名稱前綴之外，物件標籤還可以讓您依據標籤作為篩選條件，在生命週期規則中定義更精細的物件生命週期管理。
- 使用 Amazon S3 分析時，您可以配置篩選條件，依物件標籤、金鑰名稱前綴，或同時使用前綴與標籤來將物件分組。
- 您也可以自訂 Amazon CloudWatch 指標，依特定標籤篩選器顯示資訊。下列各節將詳細說明。

> **Important**
>
> 標籤可用於標記內涵機密資料 (例如個人識別資訊 (PII) 或受保護的醫療資訊 (PHI))。不過，標籤本身無法包含任何機密資訊。

使用單一請求新增物件標籤集至多個 Amazon S3 物件

若要使用單一請求新增超過一個 Amazon S3 物件的物件標籤，您可以使用 S3 Batch Operations。您可以為 S3 批次操作提供一份要進行操作的物件清單。S3 批次操作會呼叫相應的 API 操作來執行指定操作。單一批次作業任務可在包含數 EB 資料的數十億個物件上執行指定的操作。

S3 批次操作功能會追蹤進度、傳送通知，並存放所有動作的詳細完成報告，提供完整受管、可稽核、無伺服器的體驗。您可以透過 Amazon S3 主控台、 AWS CLI AWS 開發套件或 REST API 使用 S3 Batch 操作。如需詳細資訊，請參閱 the section called “批次操作基礎知識”。

如需物件標籤的詳細資訊，請參閱 管理物件標籤。

與物件標記相關的 API 操作

Amazon S3 支援下列專供物件標記之用的 API 操作：

物件 API 操作

- PUT 物件標記 – 取代物件上的標籤。請在要求內文中指定標籤。下列兩種不同的狀況都可以使用此 API 管理物件標籤。
  - 物件沒有任何標籤 - 您可以使用此 API 為物件 (先前沒有任何標籤的物件) 新增一組標籤。

- 物件目前已有一組標籤 - 若要修改現有的標籤組，您必須先擷取現有的標籤組在用戶端上修改，然後再使用此 API 取代該標籤組。

  > **Note**
  >
  > 若傳送此請求時附上空的標籤組，則 Amazon S3 會刪除物件上的現有標籤組。如果您使用此方法，系統會向您收取 Tier 1 Request (PUT) 的費用。如需詳細資訊，請參閱 Amazon S3 定價。
  > DELETE 物件標記請求是更適合的方法，因為結果相同，而不會產生費用。

- GET 物件標記 - 傳回與物件相關聯的標籤組。Amazon S3 會在回應內文中傳回物件的標籤。
- DELETE 物件標記 – 刪除與物件相關聯的標籤組。

其他支援標記的 API 操作

- PUT 物件和初始化分段上傳 – 您可以在建立物件時指定標籤。使用 `x-amz-tagging` 要求標頭可指定標籤。
- GET 物件 - 因為標頭回應大小不得超過 8 K 個位元組，所以 Amazon S3 不會傳回標籤組，而會在 `x-amz-tag-count` 標頭中傳回物件標籤計數 (只有當請求者有許可能讀取標籤時)。若希望檢視標籤，您必須另外發出 GET 物件標記 API 操作的請求。
- POST 物件 – 您可以在 POST 請求中指定標籤。

  只要要求中的標籤未超過 8 K 個位元組的 HTTP 要求標頭大小限制，您就能使用 `PUT Object` API 建立包含標籤的物件。若指定的標籤超過了標頭大小限制，可以使用此 POST 方法，在內文中加入標籤。

  PUT Object - Copy – 您可以在請求中指定 `x-amz-tagging-directive`，指示 Amazon S3 要複製 (預設行為) 標籤，或是用請求中提供的新標籤組取代標籤。

注意下列事項：

- S3 物件標記強式一致。如需詳細資訊，請參閱「Amazon S3 資料一致性模式」。

其他組態

本節說明物件標記與其他組態的相關性。

物件標記與生命週期管理

您可以在儲存貯體的生命週期組態中指定篩選條件，從物件中選取要套用規則的一部分物件。您可以使用金鑰名稱字首、物件標籤或同時使用兩者，來指定篩選條件。

假設您將相片 (原始與完稿格式) 存放在您的 Amazon S3 儲存貯體中。您可以標記這些物件，如下所示。

```
phototype=raw
or
phototype=finished
```

您可能會想要在建立相片之後，將原始相片封存到 S3 Glacier。您可以使用篩選條件設定生命週期規則，指定一組金鑰名稱字首 (photos/) 包含特定標籤 (phototype=raw) 的物件。

如需詳細資訊，請參閱「管理儲存生命週期」。

物件標記和複製

您的儲存貯體如已配置複寫，且您以授予 Amazon S3 讀取標籤的許可，Amazon S3 即會複寫標籤。如需詳細資訊，請參閱 設定即時複製。

物件標記事件通知

您可以設定在為物件新增物件標籤或刪除物件標籤時收到 Amazon S3 事件通知。s3:ObjectTagging:Put 事件類型會在物件上的標籤為 PUT 或現有標籤已更新時通知您。s3:ObjectTagging:Delete 事件類型會在標籤從物件移除時通知您。如需詳細資訊，請參閱啟用事件通知。

如需有關物件標記的詳細資訊，請參閱下列主題：

主題

- 標記與存取控制政策
- 管理物件標籤

標記與存取控制政策

您也可以使用許可政策 (儲存貯體與使用者政策) 來管理與物件標記相關的許可。如需了解政策動作，請參閱下列主題：

- 物件操作
- 儲存貯體操作

物件標籤可以為管理許進行更精細地分級存取控制。您可以使用物件標籤來授予條件式許可。Amazon S3 支援下列條件金鑰，可讓您用於授予採用依物件標籤區分的條件式許可。

- `s3:ExistingObjectTag/<tag-key>` - 使用此條件金鑰可驗證現有的物件標籤是否包含特定的金鑰與值。

> **Note**
>
> 對 PUT Object 與 DELETE Object 操作授予許可時，不支援此條件金鑰。亦即，您無法建立政策來根據現有的標籤授予或拒絕使用者刪除或覆寫物件的許可。

- `s3:RequestObjectTagKeys` - 使用此條件金鑰可限制能對物件使用的標籤金鑰。這在使用 PutObjectTagging 和 PutObject和 POST 物件要求將標籤新增至物件時非常有用。
- `s3:RequestObjectTag/<tag-key>` - 使用此條件金鑰可限制能對物件使用的標籤金鑰與值。這在使用 AND PutObjectTagging 和 PutObject POST 值區要求將標籤新增至物件時非常有用。

如需 Amazon S3 服務專用的條件金鑰詳細清單，請參閱使用條件鍵值區政策範例。下列許可政策示範物件標記如何精細分級許可管理。

Example 1：允許使用者只讀取具有特定標籤和金鑰值的物件

下列許可政策會限制使用者只能讀取具有 `environment: production` 標籤金鑰和值的物件。此政策會使用 `s3:ExistingObjectTag` 條件金鑰來指定標籤金鑰和值。

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Principal": {
      "AWS": [
        "arn:aws:iam::111122223333:role/JohnDoe"
      ]
    },
    "Effect": "Allow",
    "Action": ["s3:GetObject", "s3:GetObjectVersion"],
    "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
```

```
    "Condition": {
      "StringEquals":
        {"s3:ExistingObjectTag/environment": "production"}
    }
  }
  ]
}
```

Example 2：限制使用者可以新增哪些物件標籤金鑰

下列許可政策為使用者授予執行 s3:PutObjectTagging 動作的許可，該動作允許使用者為現有的物件新增標籤。此條件使用 s3:RequestObjectTagKeys 條件金鑰指定允許的標籤金鑰，例如 Owner 或 CreationDate。如需詳細資訊，請參閱《IAM 使用者指南》中的建立測試多個金鑰值的條件。

此政策可確保請求中指定的每個索引標籤金鑰都是授權的標籤金鑰。條件中的 ForAnyValue 限定詞可確保請求中至少會出現其中一個指定的金鑰。

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
            "arn:aws:iam::111122223333:role/JohnDoe"
         ]
       },
 "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"ForAnyValue:StringEquals": {"s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        }
      }
    }
  ]
}
```

Example 3：允許使用者新增物件標籤時，需要特定標籤金鑰和值

下列範例政策授予使用者執行 s3:PutObjectTagging 動作的許可，允許使用者將標籤新增至現有的物件。此條件需要使用者包括值設為 *X* 的特定標籤 (例如 *Project*)。

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
       "arn:aws:iam::111122223333:user/JohnDoe"
        ]
       },
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"StringEquals": {"s3:RequestObjectTag/Project": "X"
        }
      }
    }
  ]
}
```

管理物件標籤

本節說明如何使用適用於 Java 和 .NET 的 AWS 開發套件或 Amazon S3 主控台來管理物件標籤。

物件標記提供您一個分類儲存的方法。每一個標記都是符合以下規則的金鑰對數值：

- 一個物件最多可以與 10 個標籤相關聯。與物件相關聯的標籤，必須具備唯一的標籤金鑰。
- 標籤金鑰最長可包含 128 個 Unicode 字元，標籤值最長可包含 256 個 Unicode 字元。Amazon S3 物件標籤在 UTF-16 內部表示。請注意，在 UTF-16 中，字元佔用 1 或 2 個字元位置。
- 金鑰與值皆會區分大小寫。

如需物件標籤的詳細資訊，請參閱 使用標籤分類儲存空間。如需標籤限制的詳細資訊，請參閱《AWS Billing and Cost Management 使用者指南》中的使用者定義的標籤限制。

使用 S3 主控台

對物件新增標籤

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇儲存貯體的名稱，其中包含您要新增標籤的物件。

   您也可以選擇性地導覽至資料夾。
3. 在 Objects (物件) 清單中，選取要新增標籤的物件名稱旁的核取方塊。
4. 在 Actions (動作) 選單上，選擇 Edit tags (編輯標籤)。
5. 檢閱列出的物件，然後選擇 Add tags (新增標籤)。
6. 每個物件標籤都是一個鍵值對。輸入 Key (索引鍵) 和 Value (數值)。若要新增其他標籤，選擇 Add Tag (新增標籤)。

   一個物件最多可以輸入 10 個標籤。
7. 選擇 Save changes (儲存變更)。

   Amazon S3 會將標籤新增至指定的物件。

如需詳細資訊，請參閱本指南中的 在 Amazon S3 主控台中檢視物件屬性 和 上傳物件。

使用 AWS 軟體開發套件

Java

下列範例顯示如何使用 AWS SDK for Java 來設定新物件的標籤，以及擷取或取代現有物件的標籤。如需更多物件標記的詳細資訊，請參閱「使用標籤分類儲存空間」。如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.File;
```

```
import java.util.ArrayList;
import java.util.List;

public class ManagingObjectTags {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Object key ***";
        String filePath = "*** File path ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Create an object, add two new tags, and upload the object to Amazon
 S3.
            PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName,
 new File(filePath));
            List<Tag> tags = new ArrayList<Tag>();
            tags.add(new Tag("Tag 1", "This is tag 1"));
            tags.add(new Tag("Tag 2", "This is tag 2"));
            putRequest.setTagging(new ObjectTagging(tags));
            PutObjectResult putResult = s3Client.putObject(putRequest);

            // Retrieve the object's tags.
            GetObjectTaggingRequest getTaggingRequest = new
 GetObjectTaggingRequest(bucketName, keyName);
            GetObjectTaggingResult getTagsResult =
 s3Client.getObjectTagging(getTaggingRequest);

            // Replace the object's tags with two new tags.
            List<Tag> newTags = new ArrayList<Tag>();
            newTags.add(new Tag("Tag 3", "This is tag 3"));
            newTags.add(new Tag("Tag 4", "This is tag 4"));
            s3Client.setObjectTagging(new SetObjectTaggingRequest(bucketName,
 keyName, new ObjectTagging(newTags)));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
```

```
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

下列範例顯示如何使用 AWS SDK for .NET 設定新物件的標籤，以及擷取或取代現有物件的標籤。如需更多物件標記的詳細資訊，請參閱「使用標籤分類儲存空間」。

如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    public class ObjectTagsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for the new object ***";
        private const string filePath = @"*** file path ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            PutObjectWithTagsTestAsync().Wait();
        }

        static async Task PutObjectWithTagsTestAsync()
        {
```

```
            try
            {
                // 1. Put an object with tags.
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    FilePath = filePath,
                    TagSet = new List<Tag>{
                        new Tag { Key = "Keyx1", Value = "Value1"},
                        new Tag { Key = "Keyx2", Value = "Value2" }
                    }
                };

                PutObjectResponse response = await
client.PutObjectAsync(putRequest);
                // 2. Retrieve the object's tags.
                GetObjectTaggingRequest getTagsRequest = new GetObjectTaggingRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };

                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);
                for (int i = 0; i < objectTags.Tagging.Count; i++)
                    Console.WriteLine("Key: {0}, Value: {1}",
objectTags.Tagging[i].Key, objectTags.Tagging[i].Value);


                // 3. Replace the tagset.

                Tagging newTagSet = new Tagging();
                newTagSet.TagSet = new List<Tag>{
                    new Tag { Key = "Key3", Value = "Value3"},
                    new Tag { Key = "Key4", Value = "Value4" }
                };


                PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
```

```
                    Tagging = newTagSet
                };
                PutObjectTaggingResponse response2 = await
 client.PutObjectTaggingAsync(putObjTagsRequest);

                // 4. Retrieve the object's tags.
                GetObjectTaggingRequest getTagsRequest2 = new
 GetObjectTaggingRequest();
                getTagsRequest2.BucketName = bucketName;
                getTagsRequest2.Key = keyName;
                GetObjectTaggingResponse objectTags2 = await
 client.GetObjectTaggingAsync(getTagsRequest2);
                for (int i = 0; i < objectTags2.Tagging.Count; i++)
                    Console.WriteLine("Key: {0}, Value: {1}",
 objectTags2.Tagging[i].Key, objectTags2.Tagging[i].Value);

            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an
 object"
                        , e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Encountered an error. Message:'{0}' when writing an object"
                    , e.Message);
            }
        }
    }
}
```

使用成本分配 S3 儲存貯體標籤

若要追蹤個別專案或一組專案的儲存成本或其他條件，請使用成本分配標籤來標示 Amazon S3 儲存貯體。成本分配標籤是可與 S3 儲存貯體建立關聯的一組鍵/值對。在您啟用成本分配標籤之後，AWS 會使用這些標籤來組織成本分配報告上的資源成本。成本分配標籤僅使用在標示儲存貯體。如需有關標籤標示物件的詳細資訊，請參閱「使用標籤分類儲存空間」。

成本分配報告會依照產品類別和連結帳戶使用者列出您帳戶的 AWS 用量。這份報告包含與詳細帳單報告相同的明細項目 (請參閱「了解 Amazon S3 的 AWS 帳單和用量報告」) 以及標籤金鑰的額外欄。

AWS 提供兩種類型的成本分配標籤：AWS 產生的標籤和使用者定義的標籤。AWS 在 Amazon S3 CreateBucket 事件之後為您定義、建立和套用 AWS 產生的 `createdBy` 標籤。您可以定義、建立使用者定義標籤並將其套用至您的 S3 儲存貯體。

您必須先在「帳單和成本管理」主控台中分別啟用這兩種類型的標籤，它們才會顯示在帳單報告中。如需 AWS 產生的標籤詳細資訊，請參閱 AWS 產生的成本分配標籤。

- 若要在主控台中建立標籤，請參閱「檢視 S3 儲存貯體的屬性」。
- 若要使用 Amazon S3 API 建立標籤，請參閱 Amazon Simple Storage Service API 參考中的 PUT 儲存貯體標記。
- 若要使用 AWS CLI 建立標籤，請參閱《AWS CLI 命令參考》中的 put-bucket-tagging。
- 如需啟用標籤的詳細資訊，請參閱《AWS Billing 使用者指南》中的使用成本分配標籤。

使用者定義的成本分配標籤

使用者定義的成本分配標籤具有下列元件：

- 標籤金鑰。標籤金鑰是標籤名稱。例如，在 project/Trinity 標籤中，project 是金鑰。標籤金鑰是區分大小寫字串，可包含 1 到 128 個 Unicode 字元。

- 標籤值。標籤值是必要字串。例如，在 project/Trinity 標籤中，Trinity 是值。標籤值是區分大小寫字串，可包含 0 到 256 個 Unicode 字元。

如需使用者定義標籤允許的字元及其他限制詳細資訊，請參閱《AWS Billing 使用者指南》中的使用者定義標籤限制。如需使用者定義標籤的詳細資訊，請參閱《AWS Billing 使用者指南》中的使用者定義的成本分配標籤。

S3 儲存貯體標籤

每個 S3 儲存貯體都有標籤集。標籤集包含所有指派給該儲存貯體的標籤。標籤集最多可以包含 50 個標籤，也可以是空的。金鑰在標籤集內必須是唯一的，但標籤集中的值不需要是唯一的。例如，您可以在名為 project/Trinity 和 cost-center/Trinity 的標籤集中具有相同值。

在儲存貯體內，如果您所新增標籤的金鑰與現有標籤相同，則新值會覆寫舊值。

AWS 不會將任何語意意義套用至標籤。我們會將標籤嚴格解譯為字元字串。

若要新增、列出、編輯或刪除標籤，您可以使用 Amazon S3 主控台 AWS Command Line Interface (AWS CLI) 或 Amazon S3 API。

詳細資訊

- 《AWS Billing 使用者指南》中的使用成本分配標籤
- 了解 Amazon S3 的 AWS 帳單和用量報告
- AWS Billing Amazon S3 的報告

Amazon S3 的帳單和用量報告

> **Important**
>
> 我們於 2024 年 5 月 13 日開始部署變更，以免除非值區擁有者發起的未經授權要求的費用。完成此變更部署後，如果要求是從個別 AWS 帳戶或 AWS 組織以外的地方起始，儲存貯體擁有者將永遠不會針對傳回 `AccessDenied` (HTTP`403 Forbidden`) 錯誤的要求產生要求或頻寬費用。如需完整的 HTTP 清單3XX和不會計費的4XX狀態碼清單的詳細資訊，請參閱Amazon S3 錯誤回應的帳單。此帳單變更不需要更新您的應用程式，並適用於所有 S3 儲存貯體。當這項變更的部署完成時 AWS 區域，我們會更新我們的文件。

使用 Amazon S3 時，您無需支付任何預付費用或承諾存放多少內容。像其他人一樣 AWS 服務，您可以隨時隨地付費，並且只為您使用的內容付費。

AWS 為 Amazon S3 提供以下報告：

- 帳單報告 — 多個報告可提供您正在使用的所有活動 (包括 Amazon S3) 的高階檢視。 AWS 服務 AWS 一律向 S3 儲存貯體的擁有者收取 Amazon S3 費用，除非儲存貯體是建立為請求者付款儲存貯體。如需申請者付款的詳細資訊，請參閱「使用儲存體傳輸和用量的申請者付款儲存貯體」。如需帳單報告的詳細資訊，請參閱「AWS Billing Amazon S3 的報告」。
- 用量報告 – 特定服務的活動摘要 (依小時、日或月彙整)。您可以選擇要包含的用量類型及操作。您也可以選擇資料彙整方式。如需詳細資訊，請參閱「AWS Amazon S3 的使用報告」。

下列主題提供 Amazon S3 之帳單與用量報告的相關資訊。

主題

- AWS Billing Amazon S3 的報告

- AWS Amazon S3 的使用報告
- 了解 Amazon S3 的 AWS 帳單和用量報告
- Amazon S3 錯誤回應的帳單

AWS Billing Amazon S3 的報告

您的每月帳單 AWS 將您的使用信息和成本按功能 AWS 服務 分開。提供數種 AWS Billing 報告：每月報告、成本分配報告和詳細的帳單報告。如需如何查看帳單報告的資訊，請參閱《AWS Billing 使用者指南》中的檢視帳單。

若要追蹤您的 AWS 使用情況並提供與帳戶相關聯的預估費用，您可以進行設定 AWS Cost and Usage Reports。如需詳細資訊，請參閱什麼是 AWSCost and Usage Reports？ 在《資AWS 料匯出指南》中。

您也可以下載用量報告，而用量報告所提供的 Amazon S3 儲存體用量詳細資訊多於帳單報告。如需詳細資訊，請參閱「AWS Amazon S3 的使用報告」。

下表列出與 Amazon S3 用量相關聯的費用。

Amazon S3 使用費

費用	評論
儲存空間	您支付在 S3 儲存貯體中存放物件的費用。您需支付的費率取決於物件的大小、該月儲存物件的時間長度，以及儲存空間類別。Amazon S3 提供下列儲存類別：S3 標準、S3 快速單區域、S3 智慧型分層、S3 標準 — IA (不常存取的 IA)、S3 單區域 — IA、S3 冰川即時擷取、S3 冰川彈性擷取、S3 Glacier Deep Archive 或低冗餘儲存 (RRS)。如需儲存體方案的詳細資訊，請參閱「使用 Amazon S3 儲存體方案」。 請注意，如果您已啟用 S3 版本控制，則會針對保留的物件的每個版本向您收費。如需版本控制的詳細資訊，請參閱「S3 版本控制的運作方式」。

費用	評論
監控和自動化	您支付存放在 S3 Intelligent-Tiering 儲存類別中每個物件的每月監控和自動化費用，以監控存取模式和在 S3 Intelligent-Tiering 中的存取層間移動物件。
請求	您需要支付針對 S3 儲存貯體和物件發出的GET請求 (例如請求) 的費用。這包含生命週期要求。請求的費率取決於您提出的請求類型。如需要求定價的詳細資訊，請參閱 Amazon S3 定價。
擷取	您支付 S3 標準 – IA、S3 單區域 – IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 儲存中所存放物件的擷取費用。
早期刪除	如果您在經過最低儲存承諾之前，刪除存放在 S3 標準 – IA、S3 單區域 – IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存中的物件，則需要支付該物件的提早刪除費。
儲存體管理	您需要支付帳戶儲存貯體啟用的儲存管理功能 (Amazon S3 庫存、分析和物件標記) 費用。

費用	評論
頻寬	您需支付傳入和傳出 Amazon S3 的所有頻寬費用，以下除外： • 從網際網路傳入的資料 • 當執行個體與 S3 儲存貯體位於相同 AWS 區域 時，將資料傳出至 Amazon 彈性運算雲端 (Amazon EC2) 執行個體 • 資料傳出到 Amazon CloudFront (CloudFront) 您也需要為使用 Amazon S3 Transfer Acceleration 的任何資料支付費用。

如需 Amazon S3 儲存、資料傳輸和服務使用費用的詳細資訊，請參閱 Amazon S3 定價和 Amazon S3 常見問答集。

如需瞭解 Amazon S3 帳單和用量報告中使用的代碼和縮寫的相關資訊，請參閱了解 Amazon S3 的 AWS 帳單和用量報告。

詳細資訊

- AWS Amazon S3 的使用報告
- 使用成本分配 S3 儲存貯體標籤
- AWS Billing 和成本管理
- Amazon S3 定價

AWS Amazon S3 的使用報告

當您下載用量報告時，可以選擇依小時、日或月來彙整用量資料。Amazon S3 用量報告會依用量類型和列出操作 AWS 區域。如需 Amazon S3 儲存體用量的更詳細的報告，請下載動態產生的 AWS 用量報告。您可以選擇要包含的用量類型、操作與時段。您也可以選擇資料彙整方式。如需有關使用情況報告的詳細資訊，請參閱資AWS 料匯出AWS 使用指南中的使用情況報告

Amazon S3 用量報告包含下列資訊：

- 服務 – Amazon S3
- 操作 – 對儲存貯體或物件執行的操作。如需 Amazon S3 操作的詳細說明，請參閱 追蹤用量報告中的操作。
- UsageType— 下列其中一個值：
  - 識別儲存體類型的代碼
  - 識別要求類型的代碼
  - 識別擷取類型的代碼
  - 識別資料傳輸類型的代碼
  - 從 S3 Intelligent-Tiering、S3 標準 – IA、S3 One Zone-Infrequent Access (S3 One Zone-IA)、S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存中識別早期刪除的程式碼。
  - `StorageObjectCount` – 指定儲存貯體內所存放物件的計數

  如需 Amazon S3 用量類型的詳細說明，請參閱 了解 Amazon S3 的 AWS 帳單和用量報告。
- 資源 – 與所列出用量相關聯的儲存貯體名稱。
- StartTime— 適用於使用當天的開始時間，以國際標準時間 (UTC) 為單位。
- EndTime— 使用套用當天的結束時間，以國際標準時間 (UTC) 為單位。
- UsageValue— 下列其中一個體積值。資料的典型測量單位為 GB。不過，視服務和報告而定，可能會出現 TB。
  - 指定時段期間的請求數目
  - 資料傳輸量
  - 指定小時內儲存的資料量
  - 與從 S3 標準 – IA、S3 單區域 – IA、S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存還原相關聯的資料量

> **Tip**
>
> 如需 Amazon S3 針對物件所接收之每個要求的詳細資訊，請開啟您儲存貯體的伺服器存取記錄日誌。如需詳細資訊，請參閱「使用伺服器存取記錄記錄要求」。

您可以下載用量報告作為 XML 或逗號分隔值 (CSV) 檔案。下列是以試算表應用程式開啟的 CSV 用量報告範例。

Service	Operation	UsageType	Resource	StartTime	EndTime	UsageValue
AmazonS3	HeadBucket	USW2-C3DataTransfer-Out-Bytes	admin-created3	6/1/2017 0:00	7/1/2017 0:00	15309
AmazonS3	PutObject	USW2-C3DataTransfer-In-Bytes	admin-created3	6/1/2017 0:00	7/1/2017 0:00	19062
AmazonS3	HeadBucket	USW2-Requests-Tier2	admin-created3	6/1/2017 0:00	7/1/2017 0:00	68
AmazonS3	PutObjectForRepl	USW1-Requests-SIA-Tier1	ca-example-bucket	6/1/2017 0:00	7/1/2017 0:00	178294
AmazonS3	PutObjectForRepl	USW1-USW2-AWS-In-Bytes	ca-example-bucket	6/1/2017 0:00	7/1/2017 0:00	387929083
AmazonS3	GetObjectForRepl	USW2-Requests-NoCharge	admin-created3	6/1/2017 0:00	7/1/2017 0:00	108
AmazonS3	GetObjectForRepl	USW2-USW1-AWS-Out-Bytes	my-test-bucket-bash	6/1/2017 0:00	7/1/2017 0:00	387910021

如需詳細資訊，請參閱 了解 Amazon S3 的 AWS 帳單和用量報告。

下載 AWS 使用情況報告

您可以將使用情況報告下載為 XML 或 CSV 檔案。

下載用量報告

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在標題列中，選擇您的使用者名稱或帳戶 ID，然後選擇 [B illing and Cost Management]。
3. 在導航窗格中，選擇「成本」和「使用情況」報表。
4. 在「 AWS 使用情況報告」下，選擇「建立使用報告」
5. 在 [下載使用情況報告] 頁面上，選擇下列設定：
   - 服務 — 選擇 Amazon 簡單存儲服務。
   - Usage Types (使用類型) – 如需 Amazon S3 使用類型的詳細說明，請參閱 了解 Amazon S3 的 AWS 帳單和用量報告。
   - Operation (操作) – 如需 Amazon S3 操作的詳細說明，請參閱追蹤用量報告中的操作。
   - Time Period (時段) – 您希望報告涵蓋的時段。
   - Report Granularity (報告精細程度) – 讓報告依小時、日或月來包含小計。
6. 選擇 [下載]，選擇下載格式 (XML 報告或 CSV 報告)，然後依照提示開啟或儲存報告。

詳細資訊

- 了解 Amazon S3 的 AWS 帳單和用量報告
- AWS Billing Amazon S3 的報告

了解 Amazon S3 的 AWS 帳單和用量報告

> **Important**
>
> 我們於 2024 年 5 月 13 日開始部署變更，以免除非值區擁有者發起的未經授權要求的費用。完成此變更部署後，如果要求是從個別 AWS 帳戶或 AWS 組織以外的地方起始，儲存貯體擁有者將永遠不會針對傳回 `AccessDenied` (HTTP`403 Forbidden`) 錯誤的要求產生要求或頻寬費用。如需完整的 HTTP 清單`3XX`和不會計費的`4XX`狀態碼清單的詳細資訊，請參閱Amazon S3 錯誤回應的帳單。此帳單變更不需要更新您的應用程式，並適用於所有 S3 儲存貯體。當這項變更的部署完成時 AWS 區域，我們會更新我們的文件。

Amazon S3 帳單與用量報告使用代碼和縮寫。對於下表中的用法類型，請使用此清單中*region2*的縮寫來取*region*代和：*region1*

- APE1：亞太區域 (香港)
- APN1：亞太區域 (東京)
- APN2：亞太區域 (首爾)
- APN3：亞太區域 (大阪)
- APS1：亞太區域 (新加坡)
- APS2：從亞太區域 (雪梨)
- APS3：從亞太區域 (孟買)
- APS4：亞太區域 (雅加達)
- APS5：亞太區域 (海德拉巴)
- APS6：亞太區域 (墨爾本)
- CAN1： 加拿大 (中部)
- CAN2: 加拿大西部 (卡爾加里)
- CNN1：中國 (北京)
- CNW1：中國 (寧夏)
- AFS1: 非洲 (開普敦)
- EUC2：歐洲 (蘇黎世)
- EUN1：歐洲 (斯德哥爾摩)
- EUS2：歐洲 (西班牙)

- EUC1：歐洲 (法蘭克福)
- EU：歐洲 (愛爾蘭)
- EUS1：歐洲 (米蘭)
- EUW2：歐洲 (倫敦)
- EUW3： 歐洲 (巴黎)
- ILC1：以色列 (特拉維夫)
- MEC1：中東 (阿拉伯聯合大公國)
- MES1：中東 (巴林)
- SAE1：南美洲 (聖保羅)
- UGW1: AWS GovCloud (美國西部)
- UG1: AWS GovCloud (美國東部)
- USE1 (或無字首)：美國東部 (維吉尼亞北部)
- USE2：美國東部 (俄亥俄)
- USW1：美國西部 (加利佛尼亞北部)
- USW2：美國西部 (奧勒岡)

對於下表中的 S3 多區域存取點使用類型，請將此清單中*regiongroup2*的縮寫取代*regiongroup1*並取代為：

- AP：亞太地區
- AU：澳洲
- EU：歐洲
- IN：印度
- NA：北美洲
- SA：南美洲

區域群組是幾個 AWS 區域地理群組。如需更多詳細資訊，請參閱區域和可用區域。如需 AWS 區域的定價資訊，請參閱 Amazon S3 定價。

下表中的第一欄列出帳單與用量報告中所顯示的用量類型。資料的典型測量單位為 GB。不過，視服務和報告而定，可能會出現 TB。

用量類型

用量類型	個單位	精細程度	描述
region1-*region2*-AWS-In-ABytes	GB	每小時	*region1*從中傳輸的加速資料量 *region2*
region1-*region2*-AWS-In-ABytes-T1	GB	每小時	T1 加速傳輸至的資料量 *region2*，其*region1*中 T1 是指美國、歐洲和日本對存在點 (POPs) 的 CloudFront 請求
region1-*region2*-AWS-In-ABytes-T2	GB	每小時	傳輸到*region1*的 T2 加速資料量*region2*，其 POPs中 T2 代表所有其他 AWS 節點的 CloudFront 要求
region1-*region2*-AWS-In-Bytes	GB	每小時	傳輸至的資料*region1*量 *region2*
region1-*region2*-AWS-Out-ABytes	GB	每小時	從傳輸到的加速資料 *region1*量 *region2*
region1-*region2*-AWS-Out-ABytes-T1	GB	每小時	T1 加速傳輸*region1*到的資料量*region2*，其中 T1 是指美國、歐洲和日本向 POP 提出的 CloudFront 請求
region1-*region2*-AWS-Out-ABytes-T2	GB	每小時	從傳輸*region1*到的 T2 加速資料量*region2*，其中 T2 是指所有其他 AWS 節點對 PoP 的 CloudFront 請求
region1-*region2*-AWS-Out-Bytes	GB	每小時	從傳輸到的資料*region1*量 *region2*

用量類型	個單位	精細程度	描述
region-BatchOperations-Jobs	計數	每小時	執行的 S3 批次操作任務數目
region-BatchOperations-Objects	計數	每小時	S3 批次操作執行的物件操作數目
region-Bulk-Retrieval-Bytes	GB	每小時	使用大量 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 請求所擷取的資料量
region-BytesDeleted-GDA	GB	每月	DeleteObject 作業從 S3 Glacier 深度存檔儲存刪除的資料量
region-BytesDeleted-GIR	GB	每月	DeleteObject 作業從 S3 Glacier 即時擷取儲存刪除的資料量。
region-BytesDeleted-GLACIER	GB	每月	DeleteObject 作業從 S3 Glacier 彈性擷取儲存刪除的資料量
region-BytesDeleted-INT	GB	每月	DeleteObject 作業從 S3 智慧型分層儲存刪除的資料量
region-BytesDeleted-RRS	GB	每月	DeleteObject 作業從低冗餘儲存體 (RRS) 儲存體刪除的資料量
region-BytesDeleted-SIA	GB	每月	DeleteObject 作業從 S3 標準 — IA 儲存刪除的資料量
region-BytesDeleted-STANDARD	GB	每月	DeleteObject 作業從 S3 標準儲存刪除的資料量

用量類型	個單位	精細程度	描述
region-BytesDeleted-ZIA	GB	每月	DeleteObject 作業從 S3 單區域 — IA 儲存刪除的資料量
region-C3DataTransfer-In-Bytes	GB	每小時	從相同的亞馬遜 EC2 傳輸到 Amazon S3 的數據量 AWS 區域
region-C3DataTransfer-Out-Bytes	GB	每小時	在相同 AWS 區域內，從 Amazon S3 傳輸到 Amazon EC2 的資料量
region-CloudFront-In-Bytes	GB	每小時	從 CloudFront分發傳輸到的 AWS 區域 資料量
region-CloudFront-Out-Bytes	GB	每小時	從一個傳輸 AWS 區域 到 CloudFront 分發的資料量
region-DataTransfer-In-Bytes	GB	每小時	從網際網路到 Amazon S3 的資料傳輸量
region-DataTransfer-Out-Bytes	GB	每小時	從 Amazon S3 到網際網路 [1] 的資料傳輸量
region-DataTransfer-Regional-Bytes	GB	每小時	從 Amazon S3 傳輸到相同資 AWS 源的資料量 AWS 區域
region-EarlyDelete-ByteHrs	GB 小時	每小時	在 90 天最低承諾結束之前，已從 S3 Glacier Flexible Retrieval 儲存刪除之物件的依比例分配儲存用量 [2]

用量類型	個單位	精細程度	描述
region-EarlyDelete-GDA	GB 小時	每小時	在 180 天最低承諾結束之前，已從 S3 Glacier Deep Archive 儲存體刪除之物件的依比例分配儲存用量 [2]
region-EarlyDelete-GIR	GB 小時	每小時	在 90 天最低承諾結束之前，已從 S3 Glacier Instant Retrieval 儲存刪除之物件的依比例分配儲存用量。
region-EarlyDelete-GIR-SmObjects	GB 小時	每小時	在 90 天最低承諾結束之前，已從 S3 Glacier Instant Retrieval 刪除之小型物件 (小於 128 KB) 的依比例分配儲存用量。
region-EarlyDelete-SIA	GB 小時	每小時	在 30 天最低承諾結束之前，已從 S3 標準 – IA 刪除之物件的依比例分配的儲存體用量[3]
region-EarlyDelete-SIA-SmObjects	GB 小時	每小時	在 30 天最低承諾結束之前，已從 S3 標準 – IA 刪除之小型物件 (小於 128 KB) 的依比例分配儲存用量[3]
region-EarlyDelete-ZIA	GB 小時	每小時	在 30 天最低承諾結束之前，已從 S3 單區域 – IA 刪除之物件的依比例分配的儲存體用量[3]

用量類型	個單位	精細程度	描述
region-EarlyDelete-ZIA-SmObjects	GB 小時	每小時	在 30 天最低承諾結束之前，已從 S3 單區域 – IA 刪除之小型物件 (小於 128 KB) 的依比例分配儲存用量[3]
region-Expedited-Retrieval-Bytes	GB	每小時	使用快速 S3 Glacier Flexible Retrieval 請求所擷取資料量
region-Inventory-Objects Listed	物件	每小時	針對含存放庫清單之物件群組列出的物件數目 (物件是依儲存貯體或字首分組)
region-Monitoring-Automation-INT	物件	每小時	在 S3 Intelligent-Tiering 儲存體方案中監控與自動分層的唯一物件數量
region-MRAP-Out-Bytes	GB	每小時	透過 S3 多區域存取點端點從區域中的儲存貯體傳輸的資料量 (MRAP 資料路由定價)。
region-MRAP-In-Bytes	GB	每小時	透過 S3 多區域存取點端點從區域中的儲存貯體傳輸的資料量 (MRAP 資料路由定價)。
regiongroup1-regiongroup2- -MRAP-Out-Bytes	GB	每小時	透過 S3 多區域存取點端點從儲存貯體傳輸*regiongroup1* 到*regiongroup2* 位於 AWS 網路外部用戶端的資料量。

用量類型	個單位	精細程度	描述
regiongroup1-regiongroup2- -MRAP-In-Bytes	GB	每小時	透過 S3 多區域存取點端點*regiongroup1* 從 AWS 網路外部用戶端傳輸到儲存貯體的資料量。*regiongroup2*
region-OverwriteBytes-Copy-GDA	GB	每月	S3 Glacier 深層存檔儲存CopyObject 作業所覆寫的資料量
region-OverwriteBytes-Copy-GIR	GB	每月	S3 Glacier 即時擷取儲存的CopyObject 作業所覆寫的資料量。
region-OverwriteBytes-Copy-GLACIER	GB	每月	S3 Glacier 彈性擷取儲存CopyObject 作業所覆寫的資料量
region-OverwriteBytes-Copy-INT	GB	每月	S3 智慧型分層儲存作CopyObject 業覆寫的資料量
region-OverwriteBytes-Copy-RRS	GB	每月	從低冗餘儲存體 (RRS) 儲存體之CopyObject 作業覆寫的資料量
region-OverwriteBytes-Copy-SIA	GB	每月	S3 標準 — IA 儲存作CopyObject 業覆寫的資料量
region-OverwriteBytes-Copy-STANDARD	GB	每月	S3 標準儲存中作CopyObject 業覆寫的資料量

用量類型	個單位	精細程度	描述
region-OverwriteBytes-Copy-ZIA	GB	每月	S3 單區域 — IA 儲存作CopyObject 業所覆寫的資料量
region-OverwriteBytes-Put-GDA	GB	每月	S3 Glacier 深層存檔儲存PutObject 作業所覆寫的資料量
region-OverwriteBytes-Put-GIR	GB	每月	S3 Glacier 即時擷取儲存的PutObject 作業所覆寫的資料量。
region-OverwriteBytes-Put-GLACIER	GB	每月	S3 Glacier 彈性擷取儲存PutObject 作業所覆寫的資料量
region-OverwriteBytes-Put-INT	GB	每月	S3 智慧型分層儲存作PutObject 業覆寫的資料量
region-OverwriteBytes-Put-RRS	GB	每月	從低冗餘儲存體 (RRS) 儲存體之PutObject 作業覆寫的資料量
region-OverwriteBytes-Put-SIA	GB	每月	S3 標準 — IA 儲存作PutObject 業覆寫的資料量
region-OverwriteBytes-Put-STANDARD	GB	每月	S3 標準儲存中作PutObject 業覆寫的資料量
region-OverwriteBytes-Put-ZIA	GB	每月	S3 單區域 — IA 儲存作PutObject 業所覆寫的資料量

用量類型	個單位	精細程度	描述
region1-*region2*-S3RTC-In-Bytes	GB	每月	S3 複寫時間控制 (S3 RTC) 從PutObject ReplTime 、、、GetObject ReplTime 、InitiateMultipartUploadReplTime 和WriteACLReplTime 操*region1*作傳輸*region2*到的資料量 UploadPartReplTime CompleteMultipartUploadReplTime
region1-*region2*-S3RTC-Out-Bytes	GB	每月	S3 複寫時間控制 (S3 RTC) 從PutObject ReplTime 、、、GetObject ReplTime 、InitiateMultipartUploadReplTime 和WriteACLReplTime 操*region2*作傳輸*region1*到的資料量 UploadPartReplTime CompleteMultipartUploadReplTime

用量類型	個單位	精細程度	描述
region-Requests-GDA-Tier1	計數	每小時	S3 Glacier Deep Archive 物件上的PUTCreateMultipartUpload UploadPart 、、、、或CompleteMultipartUpload 請求數目 6 COPY POST
region-Requests-GDA-Tier2	計數	每小時	S3 Glacier Deep Archive 物件上的數量GET和HEAD請求數
region-Requests-GDA-Tier3	計數	每小時	S3 Glacier Deep Archive 標準還原請求的數目
region-Requests-GDA-Tier5	計數	每小時	大量 S3 Glacier Deep Archive 還原請求的數目
region-Requests-GIR-Tier1	計數	每小時	S3 冰川即時擷取物件上的PUTCOPY、或POST請求數目。
region-Requests-GIR-Tier2	計數	每小時	S3 Glacier 即時擷取物件上所有其他非 S3 Glacier 即時擷取-第 1 層請求的數目GET和所有其他請求。

用量類型	個單位	精細程度	描述
region-Requests-GLACIER-Tier1	計數	每小時	S3 Glacier 彈性擷取物件上的PUTCreateMultipartUpload UploadPart 、、、、或CompleteMultipartUpload 請求數目 [6] COPY POST
region-Requests-GLACIER-Tier2	計數	每小時	S3 Glacier 彈性擷取物件上未列出的要求數目GET和所有其他要求
region-Requests-INT-Tier1	計數	每小時	S3 智慧型分PUT層物件上的COPY、或POST請求數
region-Requests-INT-Tier2	計數	每小時	S3 智慧型分層物件的數目GET和所有其他非 Tier1 請求
region-Requests-SIA-Tier1	計數	每小時	S3 標準 — IA 物PUT件上的COPY、或POST請求數
region-Requests-SIA-Tier2	計數	每小時	S3 標準 — IA 物件上所有其他非 S3 冰川即時擷取-第 1 層請求的數目GET和
region-Requests-Tier1	計數	每小時	S3 標準PUTCOPY、RRS和標籤的、或POST請求數量，以及所有值區和物件的LIST請求
region-Requests-Tier2	計數	每小時	以GET及所有其他非TIER1 請求的數量

用量類型	個單位	精細程度	描述
region-Requests-Tier3	計數	每小時	對 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 和標準 S3 Glacier Flexible Retrieval 還原請求的生命週期請求數量
region-Requests-Tier4	計數	每小時	生命週期轉換至 S3 Glacier Instant Retrieval 、S3 Intelligent-Tierin g、S3 標準 – IA 或 S3 單區域 – IA 儲存的數量
region-Requests-Tier5	計數	每小時	大量 S3 Glacier Flexible Retrieval 還原請求數目
region-Requests-Tier6	計數	每小時	加速 S3 Glacier Flexible Retrieval 還原請求數目
region-Requests-Tier8	計數	每小時	S3 存取授與請求的數目
region-Requests-XZ-Tier1	計數	每小時	S3 快速單區域物件上的PUT或COPY請求數目
region-Requests-XZ-Tier2	計數	每小時	S3 快速單區域物件上所有其他非 S3 快速單區域一層 1 請求的數量GET及其他
region-Requests-ZIA-Tier1	計數	每小時	S3 單區域 — IA 物PUT件上的COPY、或POST請求數

用量類型	個單位	精細程度	描述
region-Requests-ZIA-Tier2	計數	每小時	S3 單區域 — IA 物件上所有其他非 S3 單區域 IA-TIER 1 請求的數目GET和數目
region-Retrieval-GIR	GB	每小時	從 S3 Glacier Instant Retrieval 儲存擷取的資料量。
region-Retrieval-SIA	GB	每小時	從 S3 標準 – IA 儲存體擷取的資料量
region-Retrieval-XZ	GB	每小時	在具有 S3 快速單區域儲存的指定擷取請求 (PUT或COPY) 中，資料超過 512 KB 的部分
region-Retrieval-ZIA	GB	每小時	從 S3 單區域 – IA 儲存體擷取的資料量
region-S3DSSE-In-Bytes	GB	每月	Amazon S3 雙重加密的數據量
region-S3DSSE-Out-Bytes	GB	每月	Amazon S3 解密的雙重加密資料量
region-S3G-DataTransfer-In-Bytes	GB	每小時	傳輸至 Amazon S3 以從 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存還原物件的資料量
region-S3G-DataTransfer-Out-Bytes	GB	每小時	來自 Amazon S3 的傳送資料量，以將物件轉移至 S3 Glacier 或 S3 Glacier Deep Archive 儲存體

用量類型	個單位	精細程度	描述
region-Select-Returned-Bytes	GB	每小時	使用 Select 要求從 S3 Standard 儲存體傳回的資料量
region-Select-Returned-GIR-Bytes	GB	每小時	從 S3 Glacier Instant Retrieval 儲存選取請求而傳回的資料量。
region-Select-Returned-INT-Bytes	GB	每小時	使用 Select 要求從 S3 Intelligent-Tiering 儲存體傳回的資料量
region-Select-Returned-SIA-Bytes	GB	每小時	使用 Select 要求從 S3 標準 – IA 儲存體傳回的資料量
region-Select-Returned-ZIA-Bytes	GB	每小時	使用 Select 要求從 S3 單區域 – IA 儲存體傳回的資料量
region-Select-Scanned-Bytes	GB	每小時	使用 Select 要求從 S3 Standard 儲存體掃描的資料量
region-Select-Scanned-GIR-Bytes	GB	每小時	從 S3 Glacier Instant Retrieval 儲存選取請求而掃描的資料量。
region-Select-Scanned-INT-Bytes	GB	每小時	使用 Select 要求從 S3 Intelligent-Tiering 儲存體掃描的資料量
region-Select-Scanned-SIA-Bytes	GB	每小時	使用 Select 要求從 S3 標準 – IA 儲存體掃描的資料量

用量類型	個單位	精細程度	描述
region-Select-Scanned-ZIA-Bytes	GB	每小時	使用 Select 要求從 S3 單區域 – IA 儲存體掃描的資料量
region-Standard-Retrieval-Bytes	GB	每小時	使用標準 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 請求所擷取的資料量
region-StorageAnalytics-ObjCount	物件	每小時	在每個儲存類別分析組態中監視到的唯一物件的數量。
region-StorageLens-ObjCount	物件	每日	由 S3 Storage Lens 進階指標和建議所追蹤的每個 S3 Storage Lens 儀表板中的唯一物件的數量。
region-StorageLensFreeTier-ObjCount	物件	每日	由 S3 Storage Lens 用量指標所追蹤的每個 S3 Storage Lens 儀表板中的唯一物件的數量。
StorageObjectCount	計數	每日	給定儲存貯體內所存放物件的數目
region-TagStorage-TagHrs	每小時標籤數	每日	依小時報告之儲存貯體中所有物件的標籤總數
region-TimedStorage-ByteHrs	GB-月	每日	資料存放在 S3 標準儲存中的 GB 月數
region-TimedStorage-GDA-ByteHrs	GB-月	每日	資料存放在 S3 Glacier 深層存檔儲存中的 GB 月數

用量類型	個單位	精細程度	描述
region-TimedStorage-GDA-Staging	GB-月	每日	資料存放在 S3 Glacier 深度存檔暫存儲中的 GB 月數
region-TimedStorage-GIR-ByteHrs	GB-月	每日	資料存放在 S3 Glacier 即時擷取儲存中的 GB 月數。
region-TimedStorage-GIR-SmObjects	GB-月	每日	小型物件 (小於 128 KB) 存放在 S3 冰川即時擷取儲存中的 GB 月數。
region-TimedStorage-GlacierByteHrs	GB-月	每日	資料存放在 S3 Glacier 彈性擷取儲存中的 GB 月數
region-TimedStorage-GlacierStaging	GB-月	每日	資料存放在 S3 Glacier 中的 GB 月數彈性擷取暫存儲
region-TimedStorage-INT-FA-ByteHrs	GB-月	每日	資料存放在 S3 智慧型分層儲存的頻繁存取層中的 GB 月數 5
region-TimedStorage-INT-IA-ByteHrs	GB-月	每日	資料存放在 S3 智慧型分層儲存的不常存取層中的 GB 月數
region-TimedStorage-INT-AA-ByteHrs	GB-月	每日	資料存放在 S3 智慧型分層儲存的封存存取層中的 GB 月數
region-TimedStorage-INT-AIA-ByteHrs	GB-月	每日	資料存放在 S3 智慧型分層儲存的封存即時存取層中的 GB 月數
region-TimedStorage-INT-DAA-ByteHrs	GB-月	每日	資料存放在 S3 智慧型分層儲存的深度存檔存取層中的 GB 月數

用量類型	個單位	精細程度	描述
region-TimedStorage-RRS-ByteHrs	GB-月	每日	資料儲存在低冗餘儲存體 (RRS) 儲存體中的 GB 月數
region-TimedStorage-SIA-ByteHrs	GB-月	每日	資料存放在 S3 標準 — IA 儲存中的 GB 月數
region-TimedStorage-SIA-SmObjects	GB-月	每日	小型物件 (小於 128 KB) 在 S3 標準 — IA 儲存中存放的 GB 月數 4
region-TimedStorage-XZ-ByteHrs	GB-月	每日	資料存放在 S3 快速單區域儲存中的 GB 月數
region-TimedStorage-ZIA-ByteHrs	GB-月	每日	資料存放在 S3 單區域 — IA 儲存中的 GB 月數
region-TimedStorage-ZIA-SmObjects	GB-月	每日	小型物件 (小於 128 KB) 存放在 S3 單區域 — IA 儲存中的 GB 月數
region-Upload-XZ	GB	每小時	S3 快速單一區域的指定上傳請求 (PUT或COPY) 中超過 512 KB 的資料量

備註

1. 如果您在完成前終止傳輸，則傳輸的資料量可能會超過應用程式接收的資料量。這種差異可能是因為傳輸終止要求無法立即執行，而且某些資料量可能正在傳輸中，等待終止要求執行。傳輸中的資料會以「傳出」的資料計費。
2. 當存檔至 S3 Glacier 即時擷取、S3 Glacier 彈性擷取或 S3 Glacier 深度存檔儲存類別的物件遭到刪除、覆寫或轉換至不同的儲存類別時，如果是 S3 Glacier 即時擷取和 S3 Glacier 彈性擷取則為 90 天，如果是 S3 Glacier 深度存檔，剩餘天數則按比例分配收費。
3. 對於 S3 標準 — IA 或 S3 單區域 — IA 儲存中的物件，在 30 天前刪除、覆寫或轉換至不同的儲存類別時，剩餘天數會按比例分配收費。

4. 對於 S3 標準 — IA 或 S3 單區域 — IA 儲存中的小型物件 (小於 128 KB)，當它們在 30 天前遭到刪除、覆寫或轉換至不同的儲存類別時，剩餘天數會按比例計費。
5. S3 Intelligent-Tiering 儲存方案中的物件沒有應計費的物件大小下限。小於 128 KB 的物件未受監控且不符合自動分層的資格。較小的物件一律存放在 S3 Intelligent-Tiering 經常存取層中。
6. 當您啟動`CreateMultipartUpload`、`UploadPart`或請`UploadPartCopy`求 S3 Glacier 彈性擷取或 S3 Glacier 深層存檔儲存類別時，請求會按 S3 標準請求費率計費，直到您完成分段上傳為止。上傳完成後，單一`CompleteMultipartUpload`請求會以目的地 S3 Glacier 儲存的費`PUT`率計費。進行中的多部分上傳部分`PUT`到 S3 Glacier 彈性擷取儲存類別會按 S3 Glacier 彈性擷取暫存費率計費，直到上傳完成為止。同樣地，對於 S3 Glacier 深度存檔儲存類別的正在進行中的多部分上傳部分，會以 S3 Glacier 深度存檔暫存費率計費，直到上傳完成為止。`PUT`
7. 對於最大 512 KB 的請求大小，S3 單一區域會針對每個請求收取固定費用。若要求和`GET`要`PUT`求超過 512 KB 的部分，則需額外支付每 GB 費用。
8. 如需 S3 Express One Zone 儲存類別所支援功能的詳細資訊，請參閱 S3 Express One Zone 不支援的 Amazon S3 功能。
9. 以 GB 計費單位的使用量類型會在使用量報告中以位元組計算。
10. GB-月是通過採取總 GB 小時數，在一個月的過程中聚總這些小時，然後除以該月的小時數來得出。要進一步了解，請參閱常見問題：使用 Amazon S3 時如何收費和計費？

> **Note**
>
> 一般而言，S3 儲存貯體擁有者會針對具有 HTTP `200 OK` 成功回應和 HTTP `4XX` 用戶端錯誤回應的請求收費。儲存貯體擁有者不會針對 HTTP `5XX` 伺服器錯誤回應 (例如 HTTP `503 Slow Down` 錯誤) 收費。如需 HTTP 下的 S3 錯誤碼`3XX`和未計費之`4XX`狀態碼的詳細資訊，請參閱Amazon S3 錯誤回應的帳單。如需儲存貯體設定為要求者付款值區時計費的詳細資訊，請參閱申請者如何支付工作的費用。

追蹤用量報告中的操作

作業會說明依指定使用類型 AWS 對物件或值區所採取的動作。操作是透過自述程式碼 (例如 `PutObject` 或 `ListBucket`) 指出。若要查看儲存貯體上的哪些動作產生特定類型的用量，請使用這些程式碼。當您建立用量報告時，可以選擇包含 All Operations (所有操作) 或特定操作 (例如 `GetObject`) 來產生報告。

詳細資訊

- AWS Amazon S3 的使用報告
- AWS Billing Amazon S3 的報告
- Amazon S3 定價
- Amazon S3 常見問

Amazon S3 錯誤回應的帳單

Important

我們於 2024 年 5 月 13 日開始部署變更，以免除非值區擁有者發起的未經授權要求的費用。完成此變更部署後，如果要求是從個別 AWS 帳戶或 AWS 組織以外的地方起始，儲存貯體擁有者將永遠不會針對傳回 AccessDenied (HTTP403 Forbidden) 錯誤的要求產生要求或頻寬費用。目前的頁面會顯示完整的 HTTP 清單3XX和不會計費的4XX狀態碼。此帳單變更不需要更新您的應用程式，並適用於所有 S3 儲存貯體。當這項變更的部署完成時 AWS 區域，我們會更新我們的文件。

一般而言，S3 儲存貯體擁有者會針對具有 HTTP 200 OK 成功回應和 HTTP 4XX 用戶端錯誤回應的請求收費。儲存貯體擁有者不會針對 HTTP 5XX 伺服器錯誤回應 (例如 HTTP 503 Slow Down 錯誤) 收費。如需儲存貯體設定為要求者付款值區時計費的詳細資訊，請參閱申請者如何支付工作的費用。

下表列出 HTTP 下的特定錯誤碼3XX和未計費的4XX狀態碼。對於使用網站託管設定的儲存貯體，當 S3 傳回自訂錯誤文件或自訂重新導向時，仍會收取適用的請求和其他費用。

Note

對於 AccessDenied (HTTP403 Forbidden)，當請求在儲存貯體擁有者的個別 AWS 帳戶或儲存貯體擁有者的 AWS 組織外部啟動時，S3 不會向儲存貯體擁有者收取費用。

HTTP 狀態碼	錯誤代碼	錯誤代碼說明	
301 Moved Permanent	PermanentRedirect	您嘗試存取的值區必須使用指定的端點進行定址。	

HTTP 狀態碼	錯誤代碼	錯誤代碼說明	
ly (301 永久移除)		將所有 future 的請求發送到此端點。	
	PermanentRedirectControlError	您嘗試存取的 API 作業必須使用指定的端點進行處理。將所有 future 的請求發送到此端點。	
307 臨時重定向	TemporaryRedirect	網域名稱系統 (DNS) 伺服器正在更新時，系統會將您重新導向至值區。	
400 錯誤的請求	AuthorizationHeaderMalformed	您提供的授權標頭無效。	
	AuthorizationQueryParametersError	您提供的授權查詢參數無效。	
	ExpiredToken	提供的令牌已過期。	
	IllegalLocationConstraintException	您嘗試從不同的區域存取值區，而非儲存貯體所在的區域。若要避免此錯誤，請使用`--region`選項。例如：`aws s3 cp` *`awsexample.txt`* `s3://` *`example-s3-bucket/`* `--region` *`ap-east-1`* 。	

HTTP 狀態碼	錯誤代碼	錯誤代碼說明	
	InvalidArgument	此錯誤可能發生的原因如下： • 指定的引數無效。 • 要求缺少必要的標頭。 • 指定的引數不完整或格式錯誤。 • 指定引數的長度必須大於或等於 3。	
	InvalidDigest	您指定的內容 MD5 或總和檢查碼值無效。	
	InvalidEncryptionAlgorithmError	您指定的加密要求無效。有效值為 AES256。	

HTTP 狀態碼	錯誤代碼	錯誤代碼說明	
	InvalidRequest	此錯誤可能發生的原因如下： • 請求使用了錯誤的簽名版本。使用AWS4-HMAC-SHA256 （簽名版本 4）。 • 存取點只能針對現有值區建立。 • 存取點不處於可刪除的狀態。 • 存取點只能針對現有值區列出。 • 下一個令牌無效。 • 生命週期規則中至少必須指定一個動作。 • 至少必須指定一個生命週期規則。 • 生命週期規則的數目不得超過 1000 個規則的允許限制。 • MaxResults 參數的範圍無效。 • SOAP 要求必須透過 HTTPS 連線提出。	

HTTP 狀態碼	錯誤代碼	錯誤代碼說明	
		• 具有非 DNS 合規名稱的儲存貯體不支援 Amazon S3 Transfer Acceleration。 • 名稱中包含句點 (.) 的儲存貯體不支援 Amazon S3 Transfer Acceleration。 • Amazon S3 Transfer Acceleration 端點僅支援虛擬樣式請求。 • 此儲存貯體未設定 Amazon S3 Transfer Acceleration。 • 此儲存貯體上的 Amazon S3 Transfer Acceleration 已停用。 • 此儲存貯體不支援 Amazon S3 Transfer Acceleration。如需協助，請聯絡AWS Support。 • 無法在此儲存貯體啟用 Amazon S3 Transfer Acceleration。如需協助，請聯絡AWS Support。 •	

HTTP 狀態碼	錯誤代碼	錯誤代碼說明	
		HTTP 標頭和查詢參數中提供的值發生衝突。 • HTTP 標頭和 POST 表單欄位中提供的值發生衝突。 • CopyObject 對大小超過 5GB 的物件提出要求。	
	無效皂請求	SOAP 要求主體無效。	
	InvalidStorageClass	您指定的儲存區類別無效。	
	InvalidTag	您的請求包含無效的標籤輸入。例如，您的請求可能包含重複的索引鍵、太長的索引鍵或值，或是系統標籤。	
	InvalidToken	提供的令牌格式錯誤或無效。	
	无效期	無法剖析指定的 URI。	
	KeyTooLongError	你的鑰匙太長了	
	惡性醫療錯誤	您提供的 ACL 格式不正確或未根據我們發佈的結構描述進行驗證。	
	不良醫療請求	您的 POST 請求的主體不是格式良好的多部分/表單數據。	

HTTP 狀態碼	錯誤代碼	錯誤代碼說明	
	马尔福梅德	您提供的 XML 格式不正確或未根據我們發佈的結構描述進行驗證。	
	MaxPostPreDataLengthExceededError	上傳檔案之前的 POST 要求欄位太大。	
	MetadataTooLarge	中繼資料標頭超過允許的中繼資料大小上限。	
	MissingRequestBodyError	您傳送空的 XML 文件做為要求。	
	MissingSecurityHeader	您的要求缺少必要的標頭。	
	NoLoggingStatusForKey	沒有像密鑰的日誌狀態子資源這樣的東西。	
	RequestHeaderSectionTooLarge	用於使請求的請求頭和查詢參數超過允許的最大大小	
	UnexpectedContent	此要求包含不受支援的內容。	
	UserKeyMustBeSpecified	存儲桶 POST 請求必須包含指定的字段名稱。如果已指定，請檢查欄位的順序。	
	IncorrectEndpoint	指定的存儲桶存在於另一個區域中。將要求導向至正確的端點。	

HTTP 狀態碼	錯誤代碼	錯誤代碼說明	
403 Forbidden (403 禁止)	RequestTimeTooSkewed	請求時間和服務器時間之間的差異太大。	
	SignatureDoesNotMatch	伺服器計算的要求簽章與您提供的簽章不符。檢查您的 AWS 密鑰訪問密鑰和簽名方法。如需詳細資訊，請參閱 REST 驗證和 SOAP 驗證。	
	NotSignedUp	您的帳戶尚未註冊 Amazon S3 服務。您必須先註冊才能使用 Amazon S3。您可以通過以下網址註冊：https://aws.amazon.com/s3	
	InvalidSecurity	提供的安全憑證無效。	
	InvalidPayer	已禁用對此物件的所有存取權。如需進一步協助，請參閱聯絡我們。	
	InvalidAccessKeyId	您提供的 AWS 存取金鑰 ID 不存在於我們的記錄中。	
	AccountProblem	有一個問題導 AWS 帳戶致操作無法成功完成。如需進一步協助，請參閱聯絡我們。	

HTTP 狀態碼	錯誤代碼	錯誤代碼說明	
	UnauthorizedAccessError	僅適用於中國地區。向沒有 ICP 授權的值區發出要求時傳回。如需詳細資訊，請參閱 ICP 記錄。	
404 Not Found (404 找不到)	NoSuchUpload	指定的多部分上傳不存在。上傳 ID 可能無效，或者分段上傳可能已中止或完成。	
	NoSuchWebsiteConfiguration	指定的值區沒有網站設定。	
405 方法不允許	MethodNotAllowed	不允許針對此資源使用指定的方法。	
四十二衝突	BucketAlreadyExists	請求的存儲桶名稱不可用。值區命名空間由系統的所有使用者共用。請指定其他名稱，然後再試一次。	
	InvalidBucketState	該請求對存儲桶的當前狀態無效。	
	OperationAborted	目前正在針對此資源執行衝突的條件式作業。再試一次	
所需長度	MissingContentLength	您必須提供內容長度 HTTP 標頭。	
412 前提條件失敗	RequestIsNotMultiPartContent	存儲桶 POST 請求必須是附件類型的多部分/表單數據。	

使用 Amazon S3 Select 篩選和擷取資料

利用 Amazon S3 Select，您可以使用結構式查詢語言 (SQL) 陳述式篩選與擷取 Amazon S3 物件內容，取得您只需要的資料子集。藉由使用 Amazon S3 Select 篩選資料，您可以降低 Amazon S3 的傳輸量，減少成本和擷取資料的延遲。

Amazon S3 選取只允許您一次查詢一個物件。它適用於以 CSV，JSON 或Apache Parquet格式存儲的對象。它也適用於使用 GZIP 或 BZIP2 壓縮的物件 (僅適用於 CSV 和 JSON 物件)，以及伺服器端加密物件。您也可以指定結果格式，CSV 或 JSON，決定您要如何分隔記錄結果。

您在要求中，傳遞 SQL 運算式給 Amazon S3。Amazon S3 Select 支援 SQL 的子集。關於由 Amazon S3 Select 所支援的 SQL 元素之詳細資訊，請參閱 適用於 Amazon S3 Select 的 SQL 參考。

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI)、`SelectObjectContent`其餘 API 作業或 AWS 開發套件來執行 SQL 查詢。

> **Note**
>
> Amazon S3 主控台限制傳回資料檔流量為 40 MB。若要擷取更多資料，請使用 AWS CLI 或 API。

需求與限制

以下是使用 Amazon S3 Select 的需求：

- 您必須擁有您正在查詢物件的 `s3:GetObject` 許可。
- 如果要查詢的物件，使用客戶提供的加密金鑰 (SSE-C) 進行加密，您需使用 `https`，也必須在請求中提供加密金鑰。

以下是使用 Amazon S3 Select 時套用的限制：

- S3 Select 每個請求只能查詢一個物件。
- SQL 運算式字串的長度上限為 256 KB。
- 輸入或結果記錄的長度上限為 1 MB。
- Amazon S3 Select 只可使用 JSON 輸出格式發送巢狀資料。

- 您無法查詢存放在 S3 Glacier 彈性擷取、S3 Glacier Deep Archive 或低冗餘儲存 (RRS) 儲存類別中的物件。您也無法查詢存放在 S3 智慧型分層封存存取層或 S3 智慧型分層深度存檔存取層中的物件。如需儲存體方案的詳細資訊，請參閱「使用 Amazon S3 儲存體方案」。

將 Amazon S3 選取與Parquet物件搭配使用時，會套用其他限制：

- Amazon S3 Select 只支援使用 GZIP 或 Snappy 的單欄式壓縮。Amazon S3 選擇不支援物件的整個Parquet物件壓縮。
- Amazon S3 Select 不支援 Parquet 輸出。您必須將輸出格式指定為 CSV 或 JSON。
- 未壓縮的列群組大小上限為 512 MB。
- 您必須使用物件結構描述中指定的資料類型。
- 選取重複的欄位只會傳回最後一個值。

建構與要求

當您建構一項請求時，您需提供使用 `InputSerialization` 物件查詢的物件的詳細資訊。您提供使用 `OutputSerialization` 物件傳回結果的詳細資訊。也還包況括使用 Amazon S3 的 SQL 運算式，篩選要求。

如需建構 Amazon S3 Select 請求的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 SelectObjectContent。您也可以在以下章節中，看到其中一則 SDK 程式碼範例：

使用掃描範圍的要求

Amazon S3 Select 可讓您以指定要查詢的位元範圍方式掃描一個物件的子集。此功能可讓您透過將一系列非重疊掃描範圍的工作分割成個別 Amazon S3 Select 要求的方式，平行掃描整個物件。

掃描範圍不需要與記錄界線對齊。Amazon S3 Select 掃描範圍要求將在指定的位元範圍中執行。記錄如果在指定的掃描範圍中開始，但超出掃描範圍，則會由查詢處理。例如：以下內容顯示了一個 Amazon S3 物件，其中包含以行分隔的 CSV 格式的一系列記錄：

```
A,B
C,D
D,E
E,F
G,H
I,J
```

假設您使用 Amazon S3 Select ScanRange 參數，並以 (位元) 1 開始，然後以 (位元) 4 結束。因此，掃描範圍會從「,」開始，然後掃描至從記錄結尾 C 為止。您的掃描範圍請求將返回結果 C, D，因為這是記錄的結尾。

Amazon S3 選取掃描範圍請求支援Parquet、CSV (不含加引號的分隔符號) 或 JSON 物件 (僅限LINES模式)。CSV 和 JSON 物件必須為未壓縮。如為行式 CSV 和 JSON 物件，當掃描範圍指定為 Amazon S3 Select 要求的一部分時，所有在掃描範圍內開始的記錄都會進行處理。如為 Parquet 物件，所有在請求的掃描範圍內開始的列群組都會進行處理。

Amazon S3 精選掃描範圍請求可與 Amazon S3 API 和 AWS 開發套件搭配使用。 AWS CLI您可以在此功能的 Amazon S3 Select 要求中使用 ScanRange 參數。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 SelectObjectContent。

錯誤

當 Amazon S3 Select 嘗試執行查詢且遇到問題時，會傳回錯誤代碼及其相關訊息。如需錯誤代碼和說明清單，請參閱 Amazon Simple Storage Service API 參考中，錯誤回應頁面的 SELECT 物件內容錯誤代碼清單一節。

如需 Amazon S3 Select 的詳細資訊，請參閱下方主題。

主題

- 在物件上使用 Amazon S3 選擇的範例
- 適用於 Amazon S3 Select 的 SQL 參考

在物件上使用 Amazon S3 選擇的範例

您可以使用 S3 選取從一個物件選取內容，方法是使用 Amazon S3 主控台、REST API 和 AWS 開發套件。

如需 S3 Select 支援的 SQL 函數的詳細資訊，請參閱 SQL 函數。

使用 S3 主控台

若要從 Amazon S3 主控台中的物件選取內容

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。

3. 選擇包含您要從中選取內容的物件的儲存貯體，然後選擇物件名稱。
4. 選擇物件動作，然後選擇使用 S3 Select 查詢。
5. 根據輸入資料的格式設定輸入設定。
6. 根據要接收的輸出格式設定輸出設定。
7. 若要從所選物件擷取記錄，請在 SQL 查詢下輸入 SELECT SQL 命令。如需如何撰寫 SQL 命令的詳細資訊，請參閱 適用於 Amazon S3 Select 的 SQL 參考。
8. 輸入 SQL 查詢之後，選擇執行 SQL 查詢。然後，在查詢結果下，您可以看到 SQL 查詢的結果。

使用 REST API

您可以使用 AWS SDK 從物件中選取內容。但也可視應用程式所需，直接傳送 REST 要求。如需請求與回應格式的詳細資訊，請參閱 SelectObjectContent。

使用 AWS 軟體開發套件

您可以使用 Amazon S3 選取`selectObjectContent`方法來選取物件的某些內容。如果此方法成功，則會傳回 SQL 運算式的結果。

Java

以下 Java 程式碼傳回每個儲存在物件記錄裡第一欄位的數值，同時包含儲存於 CSV 格式裡的資料。同時也要求回傳 `Progress` 和 `Stats` 訊息。您必須提供有效儲存貯體名稱與還有 CSV 格式資料的物件。

如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
package com.amazonaws;

import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CSVInput;
import com.amazonaws.services.s3.model.CSVOutput;
import com.amazonaws.services.s3.model.CompressionType;
import com.amazonaws.services.s3.model.ExpressionType;
import com.amazonaws.services.s3.model.InputSerialization;
import com.amazonaws.services.s3.model.OutputSerialization;
import com.amazonaws.services.s3.model.SelectObjectContentEvent;
import com.amazonaws.services.s3.model.SelectObjectContentEventVisitor;
import com.amazonaws.services.s3.model.SelectObjectContentRequest;
import com.amazonaws.services.s3.model.SelectObjectContentResult;
```

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.concurrent.atomic.AtomicBoolean;

import static com.amazonaws.util.IOUtils.copy;

/**
 * This example shows how to query data from S3Select and consume the response in
 the form of an
 * InputStream of records and write it to a file.
 */

public class RecordInputStreamExample {

    private static final String BUCKET_NAME = "${my-s3-bucket}";
    private static final String CSV_OBJECT_KEY = "${my-csv-object-key}";
    private static final String S3_SELECT_RESULTS_PATH = "${my-s3-select-results-
path}";
    private static final String QUERY = "select s._1 from S3Object s";

    public static void main(String[] args) throws Exception {
        final AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();

        SelectObjectContentRequest request = generateBaseCSVRequest(BUCKET_NAME,
 CSV_OBJECT_KEY, QUERY);
        final AtomicBoolean isResultComplete = new AtomicBoolean(false);

        try (OutputStream fileOutputStream = new FileOutputStream(new File
 (S3_SELECT_RESULTS_PATH));
             SelectObjectContentResult result =
 s3Client.selectObjectContent(request)) {
            InputStream resultInputStream =
 result.getPayload().getRecordsInputStream(
                    new SelectObjectContentEventVisitor() {
                        @Override
                        public void visit(SelectObjectContentEvent.StatsEvent event)
                        {
                            System.out.println(
                                    "Received Stats, Bytes Scanned: " +
 event.getDetails().getBytesScanned()
```

```
                                                +  " Bytes Processed: " +
event.getDetails().getBytesProcessed());
                        }

                        /*
                         * An End Event informs that the request has finished
successfully.
                         */
                        @Override
                        public void visit(SelectObjectContentEvent.EndEvent event)
                        {
                            isResultComplete.set(true);
                            System.out.println("Received End Event. Result is
complete.");
                        }
                    }
            );

            copy(resultInputStream, fileOutputStream);
        }

        /*
         * The End Event indicates all matching records have been transmitted.
         * If the End Event is not received, the results may be incomplete.
         */
        if (!isResultComplete.get()) {
            throw new Exception("S3 Select request was incomplete as End Event was
not received.");
        }
    }

    private static SelectObjectContentRequest generateBaseCSVRequest(String bucket,
String key, String query) {
        SelectObjectContentRequest request = new SelectObjectContentRequest();
        request.setBucketName(bucket);
        request.setKey(key);
        request.setExpression(query);
        request.setExpressionType(ExpressionType.SQL);

        InputSerialization inputSerialization = new InputSerialization();
        inputSerialization.setCsv(new CSVInput());
        inputSerialization.setCompressionType(CompressionType.NONE);
        request.setInputSerialization(inputSerialization);
```

```
        OutputSerialization outputSerialization = new OutputSerialization();
        outputSerialization.setCsv(new CSVOutput());
        request.setOutputSerialization(outputSerialization);

        return request;
    }
}
```

JavaScript

JavaScript 如需使用 AWS SDK for JavaScript 搭配 S3 `SelectObjectContent` API 操作從 Amazon S3 中存放的 JSON 和 CSV 檔案中選取記錄的範例，請參閱中的部落格文章介紹 Amazon S3 Select 支援的部落格文章 AWS SDK for JavaScript。

Python

如需有關使用 SQL 查詢，透過使用 S3 Select 來搜尋以逗號分隔值 (CSV) 檔案載入到 Amazon S3 的資料的 Python 範例，請參閱部落格文章使用 Amazon S3 Select 在無伺服器或資料庫的情形下查詢資料。

適用於 Amazon S3 Select 的 SQL 參考

此參考包含 Amazon S3 Select 所支援之結構化查詢語言 (SQL) 元素的描述。

主題

- SELECT 命令
- 資料類型
- 電信業者
- 保留的關鍵字
- SQL 函數

SELECT 命令

Amazon S3 Select 僅支援 `SELECT` SQL 命令。支援下列 ANSI 標準條款 `SELECT`：

- `SELECT` 清單
- `FROM` 子句

- WHERE 子句
- LIMIT 子句

> **Note**
>
> Amazon S3 Select 查詢目前不支援子查詢或聯結。

SELECT 清單

SELECT 清單會指出您要查詢傳回的欄位、函數及表達式。清單查詢的輸出。

```
SELECT *
SELECT projection1 AS column_alias_1, projection2 AS column_alias_2
```

第一個具 * (星號) 的 SELECT 表單會傳回已通過 WHERE 子句的每個資料列。SELECT 的第二個表單會針對每個欄以使用者定義的輸出純量運算式 ***projection1*** 和 ***projection2*** 建立資料列。

FROM 子句

Amazon S3 Select 支援以下形式的 FROM 子句：

```
FROM table_name
FROM table_name alias
FROM table_name AS alias
```

在 FROM 子句的每種形式中，table_name 是正在查詢的 S3Object。來自傳統關聯式資料庫的使用者可以將其做為資料庫結構描述，其中包含多個對表格的檢視。

以下標準 SQL 的FROM 子句會建立在 WHERE 子句中篩選以及在 SELECT 清單中投射的資料列。

若為存放在 Amazon S3 Select 中的 JSON 物件，您也可以使用以下形式的 FROM 子句：

```
FROM S3Object[*].path
FROM S3Object[*].path alias
FROM S3Object[*].path AS alias
```

使用 FROM 的此形式，您可以在 JSON 物件中選取陣列或物件。您可以使用下列其中一種形式來指定 path：

- 依名稱 (在物件中)：*.name* 或 ['*name*']
- 依索引 (在陣列中)：[*index*]
- 依萬用字元 (在物件中)：.*
- 依萬用字元 (在陣列中)：[*]

> **Note**
>
> - FROM 子句的此形式僅適用於 JSON 物件。
> - 萬用字元一律至少會發出一個記錄。如果沒有相符的記錄，Amazon S3 Select 會發出 MISSING 值。在輸出序列化期間 (查詢執行完成後)，Amazon S3 Select 會將 MISSING 值替換成空白記錄。
> - 彙總函數 (AVG、COUNT、MAX、MIN 和 SUM) 會略過 MISSING 值。
> - 如果您未在使用萬用字元時提供別名，您可以使用路徑中的最後一個元素來參考該列。例如，您可以使用查詢 `SELECT price FROM S3Object[*].books[*].price` 來選取書籍清單中的所有價格。如果路徑以萬用字元結尾，而不是名稱，則您可以使用值 `_1` 來參考該列。例如，您可以使用查詢 `SELECT price FROM S3Object[*].books[*].price`，而不是 `SELECT _1.price FROM S3Object[*].books[*]`。
> - Amazon S3 Select 一律會將 JSON 文件視為根層級值的陣列。因此，即使您要查詢的 JSON 物件僅具有一個根元素，FROM 子句也必須以 `S3Object[*]` 開頭。但基於相容性因素，Amazon S3 Select 允許您在不包含路徑的情況下省略萬用字元。因此，完整的 `FROM S3Object` 子句相當於 `FROM S3Object[*] as S3Object`。如果您包含路徑，則必須也使用萬用字元。因此 `FROM S3Object` 和 `FROM S3Object[*].path` 兩者皆為有效的子句，但是 `FROM S3Object.path` 則否。

Example

範例：

範例 #1

此範例顯示使用下列資料集和查詢時的結果：

```
{ "Rules": [ {"id": "1"}, {"expr": "y > x"}, {"id": "2", "expr": "z = DEBUG"} ]}
```

```
{ "created": "June 27", "modified": "July 6" }
```

```
SELECT id FROM S3Object[*].Rules[*].id
```

```
{"id":"1"}
{}
{"id":"2"}
{}
```

Amazon S3 Select 產生每項結果的原因如下：

- `{"id":"id-1"}` – `S3Object[0].Rules[0].id` 製作了比對。
- `{}` – `S3Object[0].Rules[1].id` 並不符合任何記錄，因此 Amazon S3 Select 發出 `MISSING`，該值在輸出序列化期間會變更為空白記錄並傳回。
- `{"id":"id-2"}` – `S3Object[0].Rules[2].id` 製作了比對。
- `{}` – `S3Object[1]` 在 `Rules` 上不相符，因此 Amazon S3 Select 發出 `MISSING`，該值在輸出序列化期間會變更為空白記錄並傳回。

如果您不希望 Amazon S3 Select 在找不到相符項目時傳回空白記錄，則可測試 `MISSING` 值。下列查詢會傳回與先前查詢同樣的結果，但會省略空白值：

```
SELECT id FROM S3Object[*].Rules[*].id WHERE id IS NOT MISSING
```

```
{"id":"1"}
{"id":"2"}
```

範例 #2

此範例顯示使用下列資料集和查詢時的結果：

```
{ "created": "936864000", "dir_name": "important_docs", "files": [ { "name": "." },
 { "name": ".." }, { "name": ".aws" }, { "name": "downloads" } ], "owner": "Amazon
 S3" }
{ "created": "936864000", "dir_name": "other_docs", "files": [ { "name": "." },
 { "name": ".." }, { "name": "my stuff" }, { "name": "backup" } ], "owner": "User" }
```

```
SELECT d.dir_name, d.files FROM S3Object[*] d
```

```
{"dir_name":"important_docs","files":[{"name":"."},{"name":".."},{"name":".aws"},
{"name":"downloads"}]}
{"dir_name":"other_docs","files":[{"name":"."},{"name":".."},{"name":"my stuff"},
{"name":"backup"}]}
```

```
SELECT _1.dir_name, _1.owner FROM S3Object[*]
```

```
{"dir_name":"important_docs","owner":"Amazon S3"}
{"dir_name":"other_docs","owner":"User"}
```

WHERE 子句

WHERE 子句遵循此語法：

```
WHERE condition
```

WHERE 子句根據 *condition* 篩選資料列。 條件是具有布林值結果的表達式。僅限當條件評估為在結果中傳回 TRUE 的資料列。

LIMIT 子句

LIMIT 子句遵循此語法：

```
LIMIT number
```

此 LIMIT 子句會限制您希望查詢而根據 *number* 傳回的記錄數量。

屬性存取

SELECT 和 WHERE 子句可以在以下部分使用其中一個方法參閱記錄資料，這取決於受到查詢檔案是否是 CSV 或 JSON 格式。

CSV

- 欄編號 – 您可以提及欄名稱為 *_N* 的某列第 Nth 欄，其中 *N* 是欄的位置。位置計算從 1 開始。例如，第一個欄位名稱為 _1，而第二個欄位名稱為 _2。

  您可以提及欄做為 *_N* 或 *alias*.*_N*。例如，_2 和 myAlias._2 都是在 SELECT 清單和 WHERE 子句中提及欄的有效方法。

- 欄標題 – 若 CSV 格式的物件擁有標頭列，則可將標頭用於 SELECT 清單和 WHERE 子句。尤其是，在 SELECT 和 WHERE 子句運算式中的傳統 SQL 運算式，您可以透過 *alias*.*column_name* 或 *column_name* 來提及欄。

JSON

- 文件 – 您可存取 JSON 文件欄位做為 *alias*.*name*。您也可以存取巢狀欄位；例如 *alias*.*name1*.*name2*.*name3*。
- 清單 – 您可使用以零為起始且含有 [] 運算子的索引來存取 JSON 清單中的元素。例如，您可以存取元素的第二個清單做為 *alias*[1]。您可以將存取清單元素與欄位相結合，例如，*alias*.*name1*.*name2*[1].*name3*。
- 範例：將此 JSON 物件做為範例資料集：

```
{"name": "Susan Smith",
"org": "engineering",
"projects":
    [
     {"project_name":"project1", "completed":false},
     {"project_name":"project2", "completed":true}
    ]
}
```

範例 #1

下列查詢會傳回這些結果：

```
Select s.name from S3Object s
```

```
{"name":"Susan Smith"}
```

範例 #2

下列查詢會傳回這些結果：

```
Select s.projects[0].project_name from S3Object s
```

```
{"project_name":"project1"}
```

區分大小寫的標頭和屬性名稱

搭配 Amazon S3 Select，您可以使用雙引號來指出欄標題 (適用於 CSV 物件) 和屬性 (適用於 JSON 物件) 會區分大小寫。如果沒有雙引號、物件標頭和屬性不區分大小寫。在發生模糊時即會拋出錯誤。

以下範例是 1) CSV 格式的 Amazon S3 物件，內含指定欄標頭，且將 `FileHeaderInfo` 設為 `"Use"` 以進行查詢請求；或者 2) 含指定屬性且為 JSON 格式的 Amazon S3 物件。

範例 #1：正在受到查詢的物件有標頭或屬性 `NAME`。

- 以下運算式成功將值從物件傳回。因為沒有引號，所以查詢不區分大小寫。

```
SELECT s.name from S3Object s
```

- 以下運算式產生 400 錯誤 `MissingHeaderName`。因為有引號，所以查詢區分大小寫。

```
SELECT s."name" from S3Object s
```

範例 #2：要查詢的 Amazon S3 物件內含一個標頭或屬性 `NAME` 和另一個標頭或屬性 `name`。

- 以下運算式產生 400 錯誤 `AmbiguousFieldName`。因為沒有引號，所以查詢不區分大小寫，但有兩個相符項目，因此會擲回錯誤。

```
SELECT s.name from S3Object s
```

- 以下運算式成功將值從物件傳回。因為有引號，所以查詢區分大小寫，因此沒有歧義。

```
SELECT s."NAME" from S3Object s
```

使用預留關鍵字做為使用者定義的條件

Amazon S3 Select 有一組預留關鍵字，這些關鍵字是執行用於查詢物件內容的 SQL 運算式所必需的。預留關鍵字包含函數名稱、資料類型、運算子，以此類推。在某些情況下，使用者定義的條件，像是欄標題 (適用於 CSV 檔案) 或屬性 (適用於 JSON 物件) 可能會與預留的關鍵字產生衝突。當發生這種情況，您必須使用雙引號特別表示您使用的是會與預留關鍵字衝突的使用者定義條件。否則將會發生 400 剖析錯誤。

對於預留關鍵字的完整清單，請參閱 保留的關鍵字。

以下範例是 1) CSV 格式的 Amazon S3 物件，內含指定欄標頭，且將 FileHeaderInfo 設為 "Use" 以進行查詢請求；或者 2) 含指定屬性且為 JSON 格式的 Amazon S3 物件。

範例：正受到查詢的物件有名為 CAST 的標頭或屬性，這是預留的關鍵字。

- 以下運算式成功將值從物件傳回。由於查詢中使用引號，所以 S3 Select 會使用使用者定義的標頭或屬性。

```
SELECT s."CAST" from S3Object s
```

- 以下運算式會產生 400 剖析錯誤。由於查詢中未使用引號，因此 CAST 與保留關鍵字發生衝突。

```
SELECT s.CAST from S3Object s
```

純量表達式

在 WHERE 子句和 SELECT 清單內，您可以擁有 SQL 純量表達式，這是會傳回純量值的表達式。其格式如下：

- ***literal***

  SQL 常值。

- ***column_reference***

  對表單 *column_name* 或 *alias*.*column_name* 中列的引用。

- ***unary_op expression***

  在此情況下，***unary_op*** 是一種 SQL 一元運算子。

- ***expression binary_op expression***

  在此情況下，***binary_op*** 是一種 SQL 二元運算子。

- ***func_name***

  在此情況下，***func_name*** 是要叫用的純量函數。

- ***expression*** [NOT] BETWEEN ***expression*** AND ***expression***

- ***expression*** LIKE ***expression*** [ESCAPE ***expression***]

資料類型

Amazon S3 Select 支援數種基本資料類型。

資料類型轉換

如果一般規則是遵循 CAST 函數 (如有定義的話)。如果 CAST 沒有定義，則所有輸入資料會視為一個字串。在此情況下，必要時您必須將輸入資料轉換為相關資料類型。

如需 CAST 函數的詳細資訊，請參閱「CAST」。

支援的資料類型

Amazon S3 Select 支援下列一組基本資料類型。

名稱	描述	範例
`bool`	布林值，TRUE 或 FALSE。	`FALSE`
`int`, `integer`	8 個位元組簽署的整數在從 -9,223,372,036,854,775,808 到 9,223,372,036,854,775,807 的範圍內。	`100000`
`string`	UTF8 編碼的變數長度字串。預設限制為 1 個字元。字元數上限是 2,147,483,647 個。	`'xyz'`
`float`	8 個位元組的浮點數。	`CAST(0.456 AS FLOAT)`
`decimal`, `numeric`	以 10 為底數的數字，最大的精確度為 38 (也就是有效數字的上限)，並在 -2^{31} 到 $2^{31}-1$ (也就是以 10 為底數的指數) 的範圍內。 **Note** 當您同時提供兩者時，Amazon S3 Select 會忽略比例和精確度。	`123.456`
`timestamp`	時間戳記代表特定時間點，始終包含區域位移和任意精確度。	`CAST('2007-04-05T1`

名稱	描述	範例
	在文字格式中，時間戳記遵循 W3C 的日期和時間格式，但時間戳記如果不是最少一整天的精確度，則結尾必須為常值 T。允許分數的秒值，至少一位數的精確度和無限最大值。當地時間位移可代表從 UTC 位移小時:分鐘或以常值 Z 標註 UTC 的當地時間。時間戳記上需要本機時間位移，日期值則不允許使用時間戳記。	`4:30Z' AS TIMESTAMP)`

支援的 Parquet 類型

Amazon S3 Select 支援下列 Parquet 類型。

- `DATE`
- `DECIMAL`
- `ENUM`
- `INT(8)`
- `INT(16)`
- `INT(32)`
- `INT(64)`
- `LIST`

> **Note**
>
> 對於 `LIST` Parquet 類型輸出，Amazon S3 Select 僅支援 JSON 格式。但是，如果查詢將資料限制為簡單值，也可以使用 CSV 格式查詢 `LIST` Parquet 類型。

- `STRING`
- `TIMESTAMP` 支援的精確度 (`MILLIS/MICROS/NANOS`)

> **Note**
>
> 不支援儲存為 `INT(96)` 的時間戳記。

由於 INT(64) 類型的範圍，使用 NANOS 單位的時間戳記只能代表介於 1677-09-21 00:12:43 和 2262-04-11 23:47:16 之間的值。超出此範圍的值無法以 NANOS 單位表示。

將 Parquet 類型對應到 Amazon S3 Select 中的支援資料類型

Parquet 類型	支援的資料類型
DATE	timestamp
DECIMAL	decimal, numeric
ENUM	string
INT(8)	int, integer
INT(16)	int, integer
INT(32)	int, integer
INT(64)	decimal, numeric
LIST	清單中的每個 Parquet 類型都會映設到對應的資料類型
STRING	string
TIMESTAMP	timestamp

電信業者

Amazon S3 Select 支援下列運算子。

邏輯運算子

- AND
- NOT
- OR

比較運算子

- <
- >
- <=
- >=
- =
- <>
- !=
- BETWEEN
- IN例如，。IN ('a', 'b', 'c')

模式比對運算子

- LIKE
- _ (比對任何字元)
- % (比對任何字元序列)

單一運算子

- IS NULL
- IS NOT NULL

數學運算子

支援加法、減法、乘法、除法和模數，如下所示：

- +

- -
- *
- /
- %

運算子優先順序

下表顯示運算子優先順序或遞減順序。

運算子或元素	關聯性	必要
-	右	一元減號
*, /, %	左	乘法、減法、模數
+, -	左	加法、減法
IN		設定成員資格
BETWEEN		範圍內含項目
LIKE		字串模式比對
<>		小於、大於
=	右	等式、指派
NOT	右	邏輯否定
AND	左	邏輯結合
OR	左	邏輯分離

保留的關鍵字

以下是 Amazon S3 Select 的預留關鍵字清單。這些關鍵字包含執行 SQL 運算式所需的函數名稱、資料類型、運算子等，該運算式會用來查詢物件內容。

```
absolute
action
add
all
allocate
alter
and
any
are
as
asc
assertion
at
authorization
avg
bag
begin
between
bit
bit_length
blob
bool
boolean
both
by
cascade
cascaded
case
cast
catalog
char
char_length
character
character_length
check
clob
close
coalesce
collate
collation
column
commit
connect
```

```
connection
constraint
constraints
continue
convert
corresponding
count
create
cross
current
current_date
current_time
current_timestamp
current_user
cursor
date
day
deallocate
dec
decimal
declare
default
deferrable
deferred
delete
desc
describe
descriptor
diagnostics
disconnect
distinct
domain
double
drop
else
end
end-exec
escape
except
exception
exec
execute
exists
external
```

```
extract
false
fetch
first
float
for
foreign
found
from
full
get
global
go
goto
grant
group
having
hour
identity
immediate
in
indicator
initially
inner
input
insensitive
insert
int
integer
intersect
interval
into
is
isolation
join
key
language
last
leading
left
level
like
limit
list
```

```
local
lower
match
max
min
minute
missing
module
month
names
national
natural
nchar
next
no
not
null
nullif
numeric
octet_length
of
on
only
open
option
or
order
outer
output
overlaps
pad
partial
pivot
position
precision
prepare
preserve
primary
prior
privileges
procedure
public
read
real
```

```
references
relative
restrict
revoke
right
rollback
rows
schema
scroll
second
section
select
session
session_user
set
sexp
size
smallint
some
space
sql
sqlcode
sqlerror
sqlstate
string
struct
substring
sum
symbol
system_user
table
temporary
then
time
timestamp
timezone_hour
timezone_minute
to
trailing
transaction
translate
translation
trim
true
```

```
tuple
union
unique
unknown
unpivot
update
upper
usage
user
using
value
values
varchar
varying
view
when
whenever
where
with
work
write
year
zone
```

SQL 函數

Amazon S3 Select 支援下列 SQL 函數。

主題

- 彙總函數
- 條件函數
- 轉換函數
- 日期函數
- 字串函數

彙總函數

Amazon S3 Select 支援下列彙總函數。

函數	引數類型	傳回類型
AVG(*expressio n*)	INT, FLOAT, DECIMAL	DECIMAL 適用於 INT 引數，FLOAT 適用於浮點數，否則即為相同的引數資料類型。
COUNT	-	INT
MAX(*expressio n*)	INT, DECIMAL	相同的引數類型。
MIN(*expressio n*)	INT, DECIMAL	相同的引數類型。
SUM(*expressio n*)	INT, FLOAT, DOUBLE, DECIMAL	INT 適用於 INT 引數，FLOAT 適用於浮點數，否則即為相同的引數資料類型。

SUM 範例

若要彙總 S3 清查報告中資料夾的物件大小總計，請使用 SUM 運算式。

下列 S3 清查報告是以 GZIP 壓縮的 CSV 檔案。共有三個欄。

- 第一欄是 S3 清查報告所用的 S3 儲存貯體 (*DOC-EXAMPLE-BUCKET*) 名稱。
- 第二個欄是物件金鑰名稱，可唯一識別儲存貯體中的物件。

  第一列中的 *example-folder/* 值用於資料夾 *example-folder*。在 Amazon S3 中，當您在儲存貯體建立資料夾時，S3 會建立一個 0 位元組的物件，其索引鍵設定為您提供的資料夾名稱。

第二列中的 *example-folder/object1* 值是資料夾 *example-folder* 中的物件 *object1*。

第三列中的 *example-folder/object2* 值是資料夾 *example-folder* 中的物件 *object2*。

如需 S3 資料夾的詳細資訊，請參閱 在 Amazon S3 主控台中使用資料夾整理物件。

- 第三欄是以字節為單位的物件大小。

```
"DOC-EXAMPLE-BUCKET","example-folder/","0"
"DOC-EXAMPLE-BUCKET","example-folder/object1","2011267"
"DOC-EXAMPLE-BUCKET","example-folder/object2","1570024"
```

若要使用 SUM 運算式來計算資料夾 *example-folder* 的總大小，請使用 Amazon S3 選取執行 SQL 查詢。

```
SELECT SUM(CAST(_3 as INT)) FROM s3object s WHERE _2 LIKE 'example-folder/%' AND _2 !=
 'example-folder/';
```

查詢結果：

```
3581291
```

條件函數

Amazon S3 Select 支援下列條件函數。

主題

- CASE
- COALESCE
- NULLIF

CASE

CASE 運算式是條件式運算式，類似於其他語言中的 if/then/else 陳述式。有多個條件時會使用 CASE 來指定結果。CASE 表達式有兩種類型：簡單和搜尋。

在簡單 CASE 運算式中，表達式與值相比較。發現相符時，就套用 THEN 子句中指定的動作。未發現相符時，就套用 ELSE 子句中的動作。

在搜尋 CASE 運算式中，每一個 CASE 的評估根據為布林值運算式，而 CASE 陳述式會傳回第一個相符的 CASE。如果在 WHEN 子句之間找不到相符的 CASE，就傳回 ELSE 子句中的動作。

語法

> **Note**
>
> 目前，Amazon S3 Select 不支援 ORDER BY 或包含新行的查詢。請務必使用沒有分行符號的查詢。

以下是用來比對條件的簡單 CASE 陳述式：

```
CASE expression WHEN value THEN result [WHEN...] [ELSE result] END
```

以下是用來評估每一個條件的搜尋 CASE 陳述式：

```
CASE WHEN boolean condition THEN result [WHEN ...] [ELSE result] END
```

範例

> **Note**
>
> 如果您使用 Amazon S3 主控台執行以下範例，並且 CSV 檔案包含標題列，請選擇排除 CSV 資料的第一行。

範例 1：使用簡單的 CASE 運算式在查詢中以 Big Apple 替換 New York City。以 other 取代其他所有城市名稱。

```
SELECT venuecity, CASE venuecity WHEN 'New York City' THEN 'Big Apple' ELSE 'other' END
 FROM S3Object;
```

查詢結果：

```
venuecity        |   case
-----------------+-----------
Los Angeles      | other
```

```
New York City    | Big Apple
San Francisco    | other
Baltimore        | other
...
```

範例 2：使用搜尋 CASE 運算式以根據個別門票銷售的 pricepaid 值來指派群組號碼：

```
SELECT pricepaid, CASE WHEN CAST(pricepaid as FLOAT) < 10000 THEN 'group 1' WHEN
 CAST(pricepaid as FLOAT) > 10000 THEN 'group 2' ELSE 'group 3' END FROM S3Object;
```

查詢結果：

```
pricepaid |  case
-----------+---------
12624.00 | group 2
10000.00 | group 3
10000.00 | group 3
9996.00 | group 1
9988.00 | group 1
...
```

COALESCE

COALESCE 依順序評估引數，並傳回第一個非不明值，也就是第一個非空值或非遺失值。此函數不會傳播 null 值和遺失值。

語法

```
COALESCE ( expression, expression, ... )
```

參數

expression

該函數對其運作的目標運算式。

範例

```
COALESCE(1)              -- 1
```

```
COALESCE(null)              -- null
COALESCE(null, null)        -- null
COALESCE(missing)           -- null
COALESCE(missing, missing) -- null
COALESCE(1, null)           -- 1
COALESCE(null, null, 1)     -- 1
COALESCE(null, 'string')    -- 'string'
COALESCE(missing, 1)        -- 1
```

NULLIF

由於有兩個運算式，如果兩個運算式評估為相同的值，則 NULLIF 會傳回 NULL 值，否則 NULLIF 會傳回第一個運算式評估的結果。

語法

```
NULLIF ( expression1, expression2 )
```

參數

expression1, *expression2*

該函數對其運作的目標運算式。

範例

```
NULLIF(1, 1)              -- null
NULLIF(1, 2)              -- 1
NULLIF(1.0, 1)            -- null
NULLIF(1, '1')            -- 1
NULLIF([1], [1])          -- null
NULLIF(1, NULL)           -- 1
NULLIF(NULL, 1)           -- null
NULLIF(null, null)        -- null
NULLIF(missing, null)     -- null
NULLIF(missing, missing) -- null
```

轉換函數

Amazon S3 Select 支援下列轉換函數。

主題

- CAST

CAST

CAST 函數會轉換實體，例如，將評估為單一值的表達式，從一個類型轉換為另一個類型。

語法

```
CAST ( expression AS data_type )
```

參數

expression

一或多個值的組合、運算子以及評估為值的 SQL 函數。

data_type

目標資料類型，例如 INT 轉換表達式為。如需支援的資料類型清單，請參閱 資料類型。

範例

```
CAST('2007-04-05T14:30Z' AS TIMESTAMP)
CAST(0.456 AS FLOAT)
```

日期函數

Amazon S3 Select 支援下列日期函數。

主題

- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

DATE_ADD

考量到日期部分、數量和時間戳記，DATE_ADD 透過數量更改的日期部分傳回更新的時間戳記。

語法

```
DATE_ADD( date_part, quantity, timestamp )
```

參數

date_part

指定哪些日期部分要進行修改。此可為下列其中之一：

- 年
- 月
- 天
- 小時
- 分鐘
- 秒

quantity

此值套用到更新的時間戳記。*quantity* 正值會新增至時間戳記 date_part，負值則相減。

timestamp

該函數對其運作的目標時間戳記。

範例

```
DATE_ADD(year, 5, `2010-01-01T`)                -- 2015-01-01 (equivalent to
 2015-01-01T)
DATE_ADD(month, 1, `2010T`)                     -- 2010-02T (result will add precision
 as necessary)
DATE_ADD(month, 13, `2010T`)                    -- 2011-02T
DATE_ADD(day, -1, `2017-01-10T`)                -- 2017-01-09 (equivalent to
 2017-01-09T)
DATE_ADD(hour, 1, `2017T`)                      -- 2017-01-01T01:00-00:00
DATE_ADD(hour, 1, `2017-01-02T03:04Z`)          -- 2017-01-02T04:04Z
DATE_ADD(minute, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:05:05.006Z
DATE_ADD(second, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:04:06.006Z
```

DATE_DIFF

考量到日期部分和兩個有效的時間戳記，DATE_DIFF 會傳回日期部分中的差異。當傳回的值是負整數，而 *date_part* 的 *timestamp1* 值大於 *date_part* 的 *timestamp2* 值。當傳回的值是正整數，而 *date_part* 的 *timestamp1* 值少於 *date_part* 的 *timestamp2* 值。

語法

```
DATE_DIFF( date_part, timestamp1, timestamp2 )
```

參數

date_part

指定哪些時間戳記的部分要進行比較。如需定義的詳細資訊 date_part，請參閱「DATE_ADD」。

timestamp1

要比較的第一種時間戳記。

timestamp2

要比較的第二種時間戳記。

範例

```
DATE_DIFF(year, `2010-01-01T`, `2011-01-01T`)            -- 1
DATE_DIFF(year, `2010T`, `2010-05T`)                     -- 4 (2010T is equivalent to
 2010-01-01T00:00:00.000Z)
DATE_DIFF(month, `2010T`, `2011T`)                       -- 12
DATE_DIFF(month, `2011T`, `2010T`)                       -- -12
DATE_DIFF(day, `2010-01-01T23:00`, `2010-01-02T01:00`) -- 0 (need to be at least 24h
 apart to be 1 day apart)
```

EXTRACT

考量到日期部分和時間戳記，EXTRACT 會傳回時間戳記的日期部分值。

語法

```
EXTRACT( date_part FROM timestamp )
```

參數

date_part

指定哪些時間戳記的部分要進行擷取。此可為下列其中之一：

- YEAR
- MONTH
- DAY
- HOUR
- MINUTE
- SECOND
- TIMEZONE_HOUR
- TIMEZONE_MINUTE

timestamp

該函數對其運作的目標時間戳記。

範例

```
EXTRACT(YEAR FROM `2010-01-01T`)                              -- 2010
EXTRACT(MONTH FROM `2010T`)                                   -- 1 (equivalent to
 2010-01-01T00:00:00.000Z)
EXTRACT(MONTH FROM `2010-10T`)                                -- 10
EXTRACT(HOUR FROM `2017-01-02T03:04:05+07:08`)                -- 3
EXTRACT(MINUTE FROM `2017-01-02T03:04:05+07:08`)              -- 4
EXTRACT(TIMEZONE_HOUR FROM `2017-01-02T03:04:05+07:08`)       -- 7
EXTRACT(TIMEZONE_MINUTE FROM `2017-01-02T03:04:05+07:08`)     -- 8
```

TO_STRING

考量到時間戳記和格式模式，TO_STRING 會傳回以指定格式的時間戳記字串呈現。

語法

```
TO_STRING ( timestamp time_format_pattern )
```

參數

timestamp

該函數對其運作的目標時間戳記。

time_format_pattern

字串，其中包含下列特殊字元的解釋。

格式	範例	描述
yy	69	2 位數年份
y	1969	4 位數年份
yyyy	1969	填補零之 4 位數年份
M	1	某年某月
MM	01	填補零的某年某月
MMM	Jan	精簡的月年名稱
MMMM	January	完整的月年名稱
MMMMM	J	某年某月的第一個字母 (注意：此格式搭配 TO_TIMESTAMP 函數使用無效。)
d	2	某月某日 (1-31)

格式	範例	描述
dd	02	填補零的某月某日 (01-31)
a	AM	一天的早上或下午
h	3	一天的幾時 (1-12)
hh	03	填補零的一天的幾時 (01-12)
H	3	一天的幾時 (0-23)
HH	03	填補零的一天的幾時 (00-23)
m	4	幾分 (0-59)
mm	04	填補零的小時中的分鐘 (00-59)
s	5	幾秒 (0-59)
ss	05	填補零的分鐘中的幾秒 (00-59)
S	0	幾分之幾秒 (精確度：0.1，範圍：0.0-0.9)

格式	範例	描述
SS	6	幾分之幾秒 (精確度：0.01，範圍：0.0-0.99)
SSS	60	幾分之幾秒 (精確度：0.001，範圍：0.0-0.999)
…	…	…
SSSSSSSSS	60000000	幾分之幾秒 (精確度上限：1 奈米秒，範圍：0.0-0.999999999)
n	60000000	奈米秒
X	+07 或 Z	如果位移為 0，則在小時或 Z 內位移
XX 或 XXXX	+0700 或 Z	如果位移為 0，則在小時和分鐘或 Z 內位移
XXX 或 XXXXX	+07:00 或 Z	如果位移為 0，則在小時和分鐘或 Z 內位移
x	7	在小時內位移

格式	範例	描述
xx 或 xxxx	700	在小時和分鐘內位移
xxx 或 xxxxx	+07:00	在小時和分鐘內位移

範例

```
TO_STRING(`1969-07-20T20:18Z`,  'MMMM d, y')                    -- "July 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMM d, yyyy')                   -- "Jul 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'M-d-yy')                        -- "7-20-69"
TO_STRING(`1969-07-20T20:18Z`, 'MM-d-y')                        -- "07-20-1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMMM d, y h:m a')               -- "July 20, 1969 8:18
 PM"
TO_STRING(`1969-07-20T20:18Z`, 'y-MM-dd''T''H:m:ssX')           --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00Z`, 'y-MM-dd''T''H:m:ssX')     --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXX')   --
 "1969-07-20T20:18:00+0800"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXXX')  --
 "1969-07-20T20:18:00+08:00"
```

TO_TIMESTAMP

給定一個字串，TO_TIMESTAMP 將其轉換為時間戳記。TO_TIMESTAMP 是 TO_STRING 的反轉操作。

語法

```
TO_TIMESTAMP ( string )
```

參數

string

該函數對其運作的目標字串。

範例

```
TO_TIMESTAMP('2007T')                          -- `2007T`
TO_TIMESTAMP('2007-02-23T12:14:33.079-08:00') -- `2007-02-23T12:14:33.079-08:00`
```

UTCNOW

UTCNOW 傳回 UTC 目前時間做為時間戳記。

語法

```
UTCNOW()
```

參數

UTCNOW 未取得參數。

範例

```
UTCNOW() -- 2017-10-13T16:02:11.123Z
```

字串函數

Amazon S3 Select 支援下列字串函數。

主題

- CHAR_LENGTH, CHARACTER_LENGTH
- LOWER
- SUBSTRING
- TRIM
- UPPER

CHAR_LENGTH, CHARACTER_LENGTH

CHAR_LENGTH (或 CHARACTER_LENGTH) 計數指定字串內的字元數。

> **Note**
>
> CHAR_LENGTH 和 CHARACTER_LENGTH 為同義詞。

語法

```
CHAR_LENGTH ( string )
```

參數

string

該函數對其運作的目標字串。

範例

```
CHAR_LENGTH('')          -- 0
CHAR_LENGTH('abcdefg')   -- 7
```

LOWER

考量到字串，LOWER 將所有大寫字元轉換為小寫字元。任何非大寫的字元則保持不變。

語法

```
LOWER ( string )
```

參數

string

該函數對其運作的目標字串。

範例

```
LOWER('AbCdEfG!@#$') -- 'abcdefg!@#$'
```

SUBSTRING

考量到字串、開始索引，與選擇性地長度，SUBSTRING 傳回從開始索引到高達字串結尾，或高達提供長度的子字串。

> **Note**
>
> 輸入字串的第一種字元索引位置為 1。

- 如果 start < 1，沒有指定長度，則索引位置設定為 1。
- 如果 start < 1，有指定長度，則索引位置設定為 start + length -1。
- 如果start + length -1 < 0，則會傳回一個空字串。
- 如果 start + length -1 > = 0，將被傳回從索引位置 1 開始的子字串，長度為 start + length - 1。

語法

```
SUBSTRING( string FROM start [ FOR length ] )
```

參數

string

該函數對其運作的目標字串。

start

字串的開始位置。

length

要傳回的子字串長度。如果不存在，請移至字串結尾。

範例

```
SUBSTRING("123456789", 0)      -- "123456789"
SUBSTRING("123456789", 1)      -- "123456789"
SUBSTRING("123456789", 2)      -- "23456789"
SUBSTRING("123456789", -4)     -- "123456789"
SUBSTRING("123456789", 0, 999) -- "123456789"
SUBSTRING("123456789", 1, 5)   -- "12345"
```

TRIM

從字串裁剪字首或字尾字元。要移除的預設字元是空間 (' ')。

語法

```
TRIM ( [[LEADING | TRAILING | BOTH remove_chars] FROM] string )
```

參數

string

該函數對其運作的目標字串。

LEADING | TRAILING | BOTH

此參數表示裁剪的是字首或字尾字元，或字首和字尾字元兩者。

remove_chars

要移除的一組字元。*remove_chars* 可以是長度 > 1 的字串。此函數會從遭移除之字串的開始或結束時找到的 *remove_chars* 傳回含任何字元的字串。

範例

```
TRIM('       foobar         ')               -- 'foobar'
TRIM('      \tfoobar\t         ')            -- '\tfoobar\t'
TRIM(LEADING FROM '       foobar         ')  -- 'foobar         '
TRIM(TRAILING FROM '       foobar         ') -- '       foobar'
TRIM(BOTH FROM '       foobar         ')     -- 'foobar'
TRIM(BOTH '12' FROM '1112211foobar22211122') -- 'foobar'
```

UPPER

考量到字串，UPPER 將所有小寫字元轉換為大寫字元。任何非小寫的字元則保持不變。

語法

```
UPPER ( string )
```

參數

string

該函數對其運作的目標字串。

範例

```
UPPER('AbCdEfG!@#$') -- 'ABCDEFG!@#$'
```

在 Amazon S3 物件上執行大規模批次操作

您可以使用 S3 批次作業對 Amazon S3 物件執行大規模的批次作業。S3 批次作業可以對您指定的 Amazon S3 物件清單執行單一作業。單一任務可在包含數 EB 資料的數以億計物件上執行指定的操作。Amazon S3 會追蹤進度、傳送通知，並存放所有動作的詳細完成報告，提供完整受管、可稽核、無伺服器的體驗。您可以透過 AWS Management Console、AWS CLI、Amazon SDK 或 REST API 使用 S3 批次操作。

使用 S3 批次作業來複製物件並設定物件標籤或存取控制清單 (ACL)。您也可以起始從 S3 Glacier Flexible Retrieval 還原物件，或叫用 AWS Lambda 函數來使用物件執行自訂動作。您可以對自訂物件清單執行這些操作，也可以使用 Amazon S3 庫存報告輕鬆產生物件清單。Amazon S3 批次作業使用的 Amazon S3 API 與您已在 Amazon S3 中使用的 API 相同，因此您會覺得介面很熟悉。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱 什麼是 S3 Express One Zone？ 和 目錄值區。如需使用 Batch Operations 搭配 S3 Express One Zone 和目錄儲存貯體的詳細資訊，請參閱 使用 Batch Operations 搭配 S3 Express One Zone。

S3 批次作業基礎知識

您可以使用 S3 批次作業對 Amazon S3 物件執行大規模的批次作業。S3 批次作業可以對您指定的 Amazon S3 物件清單執行單一作業或動作。

術語

本節使用術語 jobs (任務)、operations (操作) 和 tasks (任務)，這些術語的定義如下：

任務 (Job)

任務是 S3 批次作業的基本工作單位。任務包含所有針對資訊清單中所列出物件執行指定操作時的必要資訊。當您提供此資訊並請求開始任務後，任務會針對資訊清單中的每個物件執行操作。

操作

作業是您希望批次作業任務執行的 API 動作類型，例如複製物件。每個任務會在資訊清單中指定的所有物件上執行單一類型的操作。

任務

任務 (task) 是任務 (job) 的執行單位。任務代表呼叫一次 Amazon S3 或 AWS Lambda API 操作，對單一物件執行任務的操作。在任務的存留期間，S3 批次作業會為資訊清單中指定的每個物件各自建立一個任務。

S3 批次作業任務的運作方式

任務是 S3 批次作業的基本工作單位。任務包含所有針對物件清單執行指定操作的必要資訊。若要建立任務，您可以給予 S3 批次作業一個物件清單，然後指定要在這些物件上執行的動作。

如需 S3 批次操作支援之操作的相關資訊，請參閱S3 批次操作支援的操作。

批次作業會在資訊清單中包含的每個物件上執行指定操作。資訊清單會列出您希望批次任務處理的物件，並做為物件儲存在儲存貯體中。您可以使用逗號分隔值 (CSV) 格式的 Amazon S3 清查 報告做為資訊清單，輕鬆為儲存貯體中的物件建立大型清單。您也可以以簡易的 CSV 格式指定資訊清單，讓您在單一儲存貯體中包含的物件自訂清單上執行批次操作。

建立任務後，Amazon S3 便會處理資訊清單中的物件清單，並針對每個物件執行指定的作業。任務執行期間，您可以透過程式設計方式或 Amazon S3 主控台來監控進度。您也可以設定任務，在其完成時產生完成報告。完成報告會描述任務 (job) 所執行的每個任務 (task) 結果。如需監控任務的詳細資訊，請參閱 管理 S3 批次作業任務。

S3 批次操作教學課程

下列教學課 end-to-end 程介紹某些 Batch 作業工作的完整程序。

- 教學課程：使用 S3 Batch 操作進行 Batch 轉碼影片，以及 AWS LambdaAWS Elemental MediaConvert

授予 Amazon S3 批次操作的許可

建立和執行 S3 批次操作任務之前，您必須授予必要的權限。若要建立 Amazon S3 批次操作任務，則必須具備 `s3:CreateJob` 使用者許可。建立工作的同一個實體也必須具有將為工作指定的 AWS Identity and Access Management (IAM) 角色傳遞給 Batch 作業的`iam:PassRole`權限。

有關指定 IAM 資源的一般資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策、資源元素。以下各節提供有關建立 IAM 角色和連接政策的資訊。

主題

- 建立 S3 批次操作 IAM 角色
- 連接許可政策

建立 S3 批次操作 IAM 角色

Amazon S3 必須擁有代表您執行 S3 批次操作的許可。您可以透過 AWS Identity and Access Management (IAM) 角色授予這些許可。此區段提供您在建立 IAM 角色時使用的信任和許可政策的範例。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 角色。如需範例，請參閱 使用任務標籤控制 S3 批次作業的許可 和 使用 S3 批次操作複製物件。

在您的 IAM 政策中，您也可以使用條件金鑰來篩選 S3 批次操作任務的存取許可。如需詳細資訊和 Amazon S3 特定條件金鑰的完整清單，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

信任政策

若要允許 S3 批次操作服務主體擔任 IAM 角色，請將下列信任政策連接到該角色。

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

連接許可政策

根據操作類型，您可以附加下列其中一種政策。

在設定許可之前，請注意下列事項：

- 無論是哪一種操作，Amazon S3 都需要許可才能從 S3 儲存貯體中讀取資訊清單物件，並選擇性地將報告寫入儲存貯體。因此，所有下列政策都包含這些許可。
- 針對 Amazon S3 庫存報告資訊清單，S3 批次操作需要讀取 manifest.json 物件與所有相關聯 CSV 資料檔案的許可。
- 只有在指定物件的版本 ID 時才需要版本特定的許可 (如 `s3:GetObjectVersion`)。
- 如果您在加密物件上執行 S3 Batch 操作，IAM 角色也必須能夠存取用於加密物件的 AWS KMS 金鑰。
- 如果您提交使用加密的庫存報告資訊清單 AWS KMS，您的 IAM 政策必須包含許可，以`"kms:Decrypt"`及`"kms:GenerateDataKey"`清單 .json 物件和所有相關聯的 CSV 資料檔案。
- 如果 Batch 作業工作在已啟用 ACL 且位於不同 AWS 帳戶的儲存貯體中產生資訊清單，則您必須在針對批次任務設定的 IAM 角色的 IAM 政策中授予`s3:PutObjectAcl`權限。如果您未包含此權限，批次工作會失敗並顯示錯誤`Error occurred when preparing manifest: Failed to write manifest`。

複製物件： PutObject

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectTagging"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DestinationBucket/*"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::SourceBucket",
                "arn:aws:s3:::SourceBucket/*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

取代物件標記： PutObjectTagging

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::TargetResource/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
```

```
      "Resource": [
        "arn:aws:s3:::ManifestBucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::ReportBucket/*"
      ]
    }
  ]
}
```

刪除物件標記： DeleteObjectTagging

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "s3:DeleteObjectTagging",
              "s3:DeleteObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::TargetResource/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

取代存取控制清單： PutObjectAcl

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectAcl",
        "s3:PutObjectVersionAcl"
      ],
      "Resource": "arn:aws:s3:::TargetResource/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ManifestBucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::ReportBucket/*"
      ]
    }
  ]
```

```
}
```

還原物件： RestoreObject

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
          "s3:RestoreObject"
      ],
      "Resource": "arn:aws:s3:::TargetResource/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ManifestBucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::ReportBucket/*"
      ]
    }
  ]
}
```

套用物件鎖定保留： PutObjectRetention

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::TargetResource"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectRetention",
                "s3:BypassGovernanceRetention"
            ],
            "Resource": [
                "arn:aws:s3:::TargetResource/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

套用物件鎖定法律保留： PutObjectLegalHold

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

```
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::TargetResource"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "s3:PutObjectLegalHold",
            "Resource": [
                "arn:aws:s3:::TargetResource/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

複寫現有物件： InitiateReplication 使用 S3 產生的資訊清單

如果使用和存放 S3 產生的資訊清單，請使用此政策。如需使用批次操作來複寫現有物件的詳細資訊，請參閱使用 S3 批次複寫來複寫現有物件。

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

```
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
         ]
      },
      {
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:PutInventoryConfiguration"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***"
         ]
      },
      {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** manifest bucket ***/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::*** completion report bucket ****/*",
            "arn:aws:s3:::*** manifest bucket ****/*"
         ]
      }
   ]
}
```

複製現有物件： InitiateReplication 使用使用者資訊清單

如果使用使用者提供的資訊清單，請使用此政策。如需使用批次操作來複寫現有物件的詳細資訊，請參閱使用 S3 批次複寫來複寫現有物件。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
         ]
      },
            {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** manifest bucket ***/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::*** completion report bucket ****/*"
         ]
      }
   ]
}
```

建立 S3 批次操作任務

您可以使用 Amazon S3 Batch Operations，在特定 Amazon S3 物件清單上執行大規模的批次操作。本節說明建立 S3 批次操作任務所需的資訊，以及 `CreateJob` 要求的結果，它也提供使用 Amazon S3 主控台 AWS Command Line Interface (AWS CLI) 和建立 Batch 操作任務的指示 AWS SDK for Java。

建立 S3 批次操作任務時，您可以請求所有任務或僅限失敗任務的完成報告。只要順利叫用至少一個任務，S3 Batch Operations 就會產生已完成、失敗或已取消任務的報告。如需詳細資訊，請參閱 範例：S3 批次操作完成報告。

主題

批次操作任務請求元素

若要建立 S3 批次操作任務，您必須提供下列資訊：

操作

請指定希望 S3 批次操作針對資訊清單中物件執行的操作。每個操作類型都接受該操作特有的參數。使用 Batch 作業，您可以大量執行作業，其結果與您對每個物件執行該作業時 one-by-one 的結果相同。

清單檔案

清單檔案是您希望 S3 Batch Operations 對其執行指定操作的所有物件的清單。您可以使用下列方法指定 Batch Operations 任務的清單檔案：

- 手動建立自己的自訂 CSV 格式物件清單。
- 選擇現有的 CSV 格式 Amazon S3 清查 報告。
- 指示 Batch Operations 根據您在建立任務時指定的物件篩選條件自動產生清單檔案。此選項適用於您在 Amazon S3 主控台中建立的批次複寫任務，或使用 AWS 開發套件或 Amazon S3 REST API 建立的任何任務類型。 AWS CLI

> **Note**
>
> - 無論您如何指定清單檔案，此清單本身都必須儲存在一般用途儲存貯體中。Batch Operations 無法從目錄儲存貯體匯入現有的清單檔案，或將產生的清單檔案儲存到目錄儲存貯體。不過，清單檔案內描述的物件可以儲存在目錄儲存貯體中。如需詳細資訊，請參閱目錄儲存貯體。

- 如果您清單檔案中的物件位於已進行版本控制的儲存貯體中，指定物件的版本 ID 將會指示 Batch Operations 對特定版本執行操作。如果未指定版本 ID，則 Batch Operations 會對物件的最新版本執行操作。如果您的清單檔案包含版本 ID 欄位，您必須為資訊清單中的所有物件提供一個版本 ID。

如需詳細資訊，請參閱「指定資訊清單」。

優先順序

請使用任務優先順序，指出此任務與您帳戶中執行之其他任務的相對優先順序。數字越大表示優先順序越高。

任務優先順序僅在相對於為同一帳戶和區域中的其他任務所設定的優先順序有意義。您可以選擇任何適合您的編號系統。例如，您可能想要對所有還原 (`RestoreObject`) 任務指派優先順序 1，對所有複製 (`CopyObject`) 任務指派優先順序 2，以及對所有取代存取控制清單 (ACL) (`PutObjectAcl`) 任務指派優先順序 3。

S3 Batch Operations 會根據優先順序編號來排序任務的優先順序，但不保證嚴格排序。因此，請不要使用任務優先順序來確保其中任何一個任務會在其他任務之前啟動或完成。若您必須確保嚴格排序，請等待一個任務完成之後，再啟動下一個任務。

RoleArn

指定要執行工作的 AWS Identity and Access Management (IAM) 角色。您使用的 IAM 角色必須擁有足夠的許可，來執行任務中指定的操作。例如，若要執行 `CopyObject` 任務，IAM 角色必須具備來源儲存貯體的 `s3:GetObject` 許可，以及目的地儲存貯體的 `s3:PutObject` 許可。角色也需要讀取資訊清單，以及寫入任務完成報告的許可。

如需 IAM 角色的相關資訊，請參閱《IAM 使用者指南》中的 IAM 角色。

如需 Amazon S3 許可的詳細資訊，請參閱 Amazon S3 的政策動作。

> **Note**
>
> 對目錄儲存貯體執行動作的 Batch Operations 任務須具備特定許可。如需詳細資訊，請參閱適用於 S3 Express One Zone 的AWS Identity and Access Management (IAM)。

報告

指定是否希望 S3 批次操作產生完整報告。若您要請求任務完成報告，您也必須在此元素中提供報告的參數。必要資訊包括：

- 您希望在其中存放報告的儲存貯體

> **Note**
>
> 報告必須儲存在一般用途儲存貯體中。Batch Operations 無法將報告儲存至目錄儲存貯體。如需詳細資訊，請參閱目錄儲存貯體。

- 報告的格式
- 您希望報告包含所有任務的詳細資訊，還是僅限失敗的任務
- 選擇性的字首字串

> **Note**
>
> 完成報告一律使用 Amazon S3 受管金鑰 (SSE-S3) 加密。

標籤 (選用)

您可以透過新增標籤，標示和控制對 S3 批次操作任務的存取權。您可以使用標籤來識別負責 Batch Operations 任務的人員，或控制使用者與 Batch Operations 任務互動的方式。任務標籤的存在可以授與或限制使用者的以下能力：取消任務、啟動處於確認狀態的任務或變更任務的優先順序層級。例如，假設建立的任務具有標籤 `"Department=Finance"`，您就可以授予使用者調用 `CreateJob` 操作的許可。

您可以建立已連接標籤的任務，也可以在建立任務後將標籤新增至任務。

如需詳細資訊，請參閱「the section called “使用標籤”」。

描述 (選用)

若要追蹤和監控任務，您也可以提供最多 256 個字元的說明。每當 Amazon S3 在 Amazon S3 主控台上傳回任務的相關資訊或顯示任務的詳細資訊時，都會包含此說明。您可以輕鬆地根據指派的描述來排序和篩選任務。描述不需要是唯一的，因此您可以使用描述作為類別 (例如「每週複製日誌任務」) 來協助您追蹤相似任務的群組。

指定資訊清單

清單檔案是 Amazon S3 物件，其中包含您希望 Amazon S3 採取行動的物件索引鍵。您可透過下列其中一種方式供應清單檔案：

- 手動建立新的清單檔案。
- 使用現有的清單檔案。
- 指示 Batch Operations 根據您在建立任務時指定的物件篩選條件自動產生清單檔案。此選項適用於您在 Amazon S3 主控台中建立的批次複寫任務，或使用 AWS 開發套件或 Amazon S3 REST API 建立的任何任務類型。 AWS CLI

> **Note**
>
> 無論您如何指定清單檔案，此清單本身都必須儲存在一般用途儲存貯體中。Batch Operations 無法從目錄儲存貯體匯入現有的清單檔案，或將產生的清單檔案儲存到目錄儲存貯體。不過，清單檔案內描述的物件可以儲存在目錄儲存貯體中。如需詳細資訊，請參閱目錄儲存貯體。

建立清單檔案

若要手動建立清單檔案，您可指定清單檔案物件索引鍵、ETag (實體標籤) 和選用的版本 ID (使用 CSV 格式清單)。資訊清單的內容必須為 URL 編碼。

根據預設，Amazon S3 會自動使用採用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密，來加密上傳至 S3 儲存貯體的清單檔案。不支援使用以客戶提供之金鑰 (SSE-C) 進行伺服器端加密的資訊清單。只有 AWS Key Management Service 當您使用 CSV 格式的庫存報告時，才支援使用伺服器端加密 () 金鑰 (SSE-KMS) 的資訊清單。AWS KMS不支援使用手動建立 AWS KMS 的資訊清單。

您的資訊清單必須包含儲存貯體名稱、物件金鑰，也可選擇納入各物件的物件版本。S3 批次作業不會使用資訊清單中的任何其他欄位。

> **Note**
>
> 如果您清單檔案中的物件位於已進行版本控制的儲存貯體中，指定物件的版本 ID 將會指示 Batch Operations 對特定版本執行操作。如果未指定版本 ID，則 Batch Operations 會對物件的最新版本執行操作。如果您的清單檔案包含版本 ID 欄位，您必須為資訊清單中的所有物件提供一個版本 ID。

以下為不帶版本 ID 的 CSV 格式資訊清單範例。

```
Examplebucket,objectkey1
Examplebucket,objectkey2
Examplebucket,objectkey3
Examplebucket,photos/jpgs/objectkey4
Examplebucket,photos/jpgs/newjersey/objectkey5
Examplebucket,object%20key%20with%20spaces
```

以下是包含版本 ID 的 CSV 格式清單檔案範例。

```
Examplebucket,objectkey1,PZ9ibn9D5lP6p298B7S9_ceqx1n5EJ0p
Examplebucket,objectkey2,YY_ouuAJByNW1LRBfFMfxMge7XQWxMBF
Examplebucket,objectkey3,jbo9_jhdPEyB4RrmOxWS0kU0EoNrU_oI
Examplebucket,photos/jpgs/objectkey4,6EqlikJJxLTsHsnbZbSRffn24_eh5Ny4
Examplebucket,photos/jpgs/newjersey/objectkey5,imHf3FAiRsvBW_EHB8GOu.NHunHO1gVs
Examplebucket,object%20key%20with%20spaces,9HkPvDaZY5MVbMhn6TMn1YTb5ArQAo3w
```

指定現有的清單檔案

您可以使用下列兩種格式之一指定建立任務請求的清單檔案：

- Amazon S3 庫存清單報告：必須是 CSV 格式的 Amazon S3 庫存清單報告。您必須指定與庫存報告關聯的 `manifest.json` 檔案。如需庫存報告的詳細資訊，請參閱 Amazon S3 清查。如果庫存報告包含版本 ID，則 S3 批次作業會對特定物件版本執行操作。

  > **Note**
  >
  > - S3 Batch Operations 支援使用 SSE-KMS 加密的 CSV 庫存報告。
  > - 如果您提交使用 SSE-KMS 加密的庫存報告清單檔案，您的 IAM 政策必須包含 `manifest.json` 物件及所有相關聯 CSV 資料檔案的許可 `"kms:GenerateDataKey"` 和 `"kms:Decrypt"`。

- CSV 檔案：檔案中的每一列都必須包含儲存貯體名稱、物件索引鍵及選用的物件版本。物件金鑰必須使用 URL 編碼，如下列範例所示。資訊清單必須包含所有物件的版本 ID，或省略所有物件的版本 ID。如需 CSV 清單檔案格式的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 JobManifestSpec。

> **Note**
>
> S3 Batch Operations 不支援使用 SSE-KMS 加密的 CSV 清單檔案。

> **Important**
>
> 如果您使用手動建立的清單檔案和版本控制的儲存貯體時，我們建議您指定物件的版本 ID。建立任務時，S3 批次作業會在執行任務之前剖析整個資訊清單。但是，它不會為儲存貯體的狀態拍攝「快照」。
>
> 由於清單檔案可能包含數十億個物件，因此任務可能需要很長的時間來執行，這樣可能會影響任務採取行動的物件版本。假設您在任務執行時，以新版本覆寫物件，而您未指定該物件的版本 ID。在此情況下，Amazon S3 會對物件的最新版本執行操作，而非您建立任務時已存在的版本。避免此行為最簡單的方式，便是為資訊清單中列出的物件指定版本 ID。

自動產生清單檔案

您可以指示 Amazon S3 根據您在建立任務時指定的物件篩選條件自動產生清單檔案。此選項適用於您在 Amazon S3 主控台中建立的批次複寫任務，或使用 AWS 開發套件或 Amazon S3 REST API 建立的任何任務類型。 AWS CLI如需批次複寫的詳細資訊，請參閱 使用 S3 批次複寫來複寫現有物件。

若要自動產生清單檔案，請在任務建立請求中指定下列元素：

- 包含來源物件的儲存貯體相關資訊，包括儲存貯體擁有者和 Amazon Resource Name (ARN)
- 清單檔案輸出的相關資訊，包括建立清單檔案的旗標、輸出儲存貯體擁有者、ARN、字首、檔案格式及加密類型
- 依建立日期、索引鍵名稱、大小、儲存類別和標籤篩選物件的選用條件標準

物件篩選條件

若要篩選要包含在自動產生的清單檔案中的物件清單，您可以指定下列篩選條件。如需詳細資訊，請參閱 Amazon S3 API 參考中的 JobManifestGeneratorFilter。

CreatedAfter

如有提供，則產生的清單檔案只會包含在此時間之後建立的來源儲存貯體物件。

CreatedBefore

如有提供，則產生的清單檔案只會包含在此時間之前建立的來源儲存貯體物件。

EligibleForReplication

如有提供，則產生的清單檔案只會包含根據來源儲存貯體上的複寫組態，符合複寫資格的物件。

KeyNameConstraint

如果有提供，產生的資訊清單只會包含其物件索引鍵符合
為MatchAnySubstring、MatchAnyPrefix和指定的字串條件約束的來源值區物件MatchAnySuffix。

MatchAnySubstring— 如果提供，如果指定的字串出現在物件索引鍵字串中的任何位置，則產生的資訊清單會包含物件

MatchAnyPrefix— 如果提供，如果指定的字串出現在物件索引鍵字串的開頭，則產生的資訊清單會包含物件。

MatchAnySuffix— 如果提供，如果指定的字串出現在物件索引鍵字串的結尾，則產生的資訊清單會包含物件。

MatchAnyStorageClass

如有提供，則產生的清單檔案只會包含以指定的儲存類別儲存的來源儲存貯體物件。

ObjectReplicationStatuses

如有提供，則產生的清單檔案只會包含具有其中一種指定複寫狀態的來源儲存貯體物件。

ObjectSizeGreaterThanBytes

如有提供，則產生的清單檔案只會包含檔案大小大於所指定位元組數目的來源儲存貯體物件。

ObjectSizeLessThanBytes

如有提供，則產生的清單檔案只會包含檔案大小小於所指定位元組數目的來源儲存貯體物件。

> **Note**
>
> 您無法複製大部分已自動產生清單檔案的任務。除非批次複寫任務使用 `KeyNameConstraint`、`MatchAnyStorageClass`、`ObjectSizeGreaterThanBytes` 或 `ObjectSizeLessThanBytes` 清單檔案篩選條件，否則可以複製這些任務。

指定清單檔案條件的語法會根據您用來建立任務的方法而有所不同。如需範例，請參閱 建立任務。

建立任務

您可以使用 Amazon S3 主控台、 AWS CLI AWS 開發套件或 Amazon S3 REST API 來建立 S3 Batch 操作任務。

如需建立任務請求的詳細資訊，請參閱 批次操作任務請求元素。

必要條件

建立 Batch Operations 任務之前，請先確認您已設定相關許可。如需詳細資訊，請參閱「授予 Amazon S3 批次操作的許可」。

使用 S3 主控台

建立批次任務

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在頁面頂端的導覽列中，選擇目前顯示的名稱 AWS 區域。接下來，選擇您要在其中建立工作的地區。

   > **Note**
   >
   > 針對複製作業，您必須在與目的地時段相同的「區域」中建立工單。對於所有其他作業，您必須在與資訊清單中的物件相同的「區域」中建立工作。

3. 在 Amazon S3 主控台的左側導覽窗格中選擇「Batch 操作」。
4. 選擇建立作業。
5. 檢視您AWS 區域要建立工作的位置。
6. 在 Manifest format (資訊清單格式) 下，選擇要使用的資訊清單物件類型。

   - 如果您選擇 S3 inventory report (S3 庫存報告)，請輸入 Amazon S3 在 CSV 格式庫存報告中所產生 manifest.json 物件的路徑。若要使用最新版本以外的版本，則可選擇輸入資訊清單物件的版本 ID。
   - 如果您選擇 CSV，請輸入 CSV 格式資訊清單物件的路徑。資訊清單物件必須遵循主控台中所描述的格式。如果要使用最新版本以外的版本，則可以選擇包含資訊清單物件的版本 ID。

> **Note**
>
> Amazon S3 主控台僅支援針對批次複寫任務自動產生清單檔案。對於所有其他任務類型，如果您希望 Amazon S3 根據您指定的篩選條件自動產生資訊清單，則必須使用 AWS 開發套件或 Amazon S3 REST API 來設定您的任務。 AWS CLI

7. 選擇 Next (下一步)。
8. 在 Operation (操作) 底下，選擇要對資訊清單上所有物件執行的操作。填寫您選擇的操作資訊，然後選擇 Next (下一步)。
9. 填寫 Configure additional options (設定其他選項) 的資訊，然後選擇 Next (下一步)。
10. 在 Review (檢閱) 中，確認您的設定。如需變更，請選擇 Previous (上一步)。否則，請選擇 Create Job (建立任務)。

使用 AWS CLI

Specify manifest

下列範例顯示如何建立 S3 Batch Operations `S3PutObjectTagging` 任務，讓它對現有清單檔案中列出的物件執行動作。

建立批次操作 **`S3PutObjectTagging`** 任務

1. 使用下列命令建立 AWS Identity and Access Management (IAM) 角色，然後建立 IAM 政策以指派相關許可。下列角色和政策會授予 Amazon S3 許可來新增物件標籤，您在後續步驟中建立任務時將會需要這些標籤。

   a. 使用下列範例命令來建立要供 Batch Operations 使用的 IAM 角色。若要使用此範例命令，請將 *`S3BatchJobRole`* 取代為您要為角色指定的名稱。

```
aws iam create-role \
 --role-name S3BatchJobRole \
 --assume-role-policy-document '{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
```

```
        },
        "Action":"sts:AssumeRole"
      }
   ]
}'
```

記錄角色的 Amazon Resource Name (ARN)。在建立任務時，您將需要 ARN。

b. 使用下列範例命令來建立具有必要許可的 IAM 政策，並將其連接到上一個步驟中建立的 IAM 角色。如需有關必要許可的詳細資訊，請參閱 授予 Amazon S3 批次操作的許可。

> **Note**
>
> 對目錄儲存貯體執行動作的 Batch Operations 任務須具備特定許可。如需詳細資訊，請參閱適用於 S3 Express One Zone 的AWS Identity and Access Management (IAM)。

若要使用此範例命令，請取代 *user input placeholders*，如下所示：

- 將 *S3BatchJobRole* 取代為您的 IAM 角色名稱。確定此名稱與您之前使用的名稱相符。
- 將 *PutObjectTaggingBatchJobPolicy* 取代為您要為 IAM 政策指定的名稱。
- 將 *example-s3-destination-bucket* 取代為包含您要套用標籤之物件的儲存貯體名稱。
- 將 *DOC-EXAMPLE-MANIFEST-BUCKET* 取代為包含清單檔案的儲存貯體名稱。
- 將 *DOC-EXAMPLE-REPORT-BUCKET* 取代為要接收完成報告的儲存貯體名稱。

```
aws iam put-role-policy \
  --role-name S3BatchJobRole \
  --policy-name PutObjectTaggingBatchJobPolicy \
  --policy-document '{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectTagging",
```

```
        "s3:PutObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::example-s3-destination-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-BUCKET/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:GetBucketLocation"
      ],
      "Resource":[
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET/*"
      ]
    }
  ]
}'
```

2. 使用下列範例命令來建立 S3PutObjectTagging 任務。

   manifest.csv 檔案提供儲存貯體和物件金鑰值的清單。該任務會將指定的標籤套用至清單檔案中識別的物件。ETag 是 manifest.csv 物件的 ETag (您可以從 Amazon S3 主控台取得該物件)。此請求會指定 no-confirmation-required 參數，因此您不需使用 update-job-status 命令進行確認就可以執行任務。如需詳細資訊，請參閱 AWS CLI 命令參考中的 create-job。

   若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。將 *IAM-role* 取代為您先前建立之 IAM 角色的 ARN。

```
aws s3control create-job \
    --region us-west-2 \
```

```
    --account-id acct-id \
    --operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"keyOne",
 "Value":"ValueOne"}] }}' \
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":
{"ObjectArn":"arn:aws:s3:::my_manifests/
manifest.csv","ETag":"60e460c9d1046e73f7dde5043ac3ae85"}}' \
    --report '{"Bucket":"arn:aws:s3:::DOC-EXAMPLE-REPORT-
BUCKET","Prefix":"final-reports",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
    --priority 42 \
    --role-arn IAM-role \
    --client-request-token $(uuidgen) \
    --description "job description" \
    --no-confirmation-required
```

為了回應，Amazon S3 會傳回任務 ID (例如 `00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c`)。您將需要任務 ID 來識別、監控和修改任務。

Generate manifest

下列範例顯示如何建立 S3 Batch Operations `S3DeleteObjectTagging` 任務，讓它根據您的物件篩選條件自動產生清單檔案。此條件包括建立日期、索引鍵名稱、大小、儲存類別和標籤。

建立批次操作 `S3DeleteObjectTagging` 任務

1. 使用下列命令建立 AWS Identity and Access Management (IAM) 角色，然後建立 IAM 政策以指派許可。下列角色和政策會授予 Amazon S3 許可來刪除物件標籤，您在後續步驟中建立任務時將會需要這些標籤。

   a. 使用下列範例命令來建立要供 Batch Operations 使用的 IAM 角色。若要使用此範例命令，請將 *S3BatchJobRole* 取代為您要為角色指定的名稱。

   ```
   aws iam create-role \
    --role-name S3BatchJobRole \
    --assume-role-policy-document '{
      "Version":"2012-10-17",
      "Statement":[
         {
            "Effect":"Allow",
            "Principal":{
   ```

```
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}'
```

記錄角色的 Amazon Resource Name (ARN)。在建立任務時，您將需要 ARN。

b. 使用下列範例命令來建立具有必要許可的 IAM 政策，並將其連接到上一個步驟中建立的 IAM 角色。如需有關必要許可的詳細資訊，請參閱 授予 Amazon S3 批次操作的許可。

> **Note**
>
> 對目錄儲存貯體執行動作的 Batch Operations 任務須具備特定許可。如需詳細資訊，請參閱適用於 S3 Express One Zone 的AWS Identity and Access Management (IAM)。

若要使用此範例命令，請取代 *user input placeholders*，如下所示：

- 將 *S3BatchJobRole* 取代為您的 IAM 角色名稱。確定此名稱與您之前使用的名稱相符。
- 將 *DeleteObjectTaggingBatchJobPolicy* 取代為您要為 IAM 政策指定的名稱。
- 將 *example-s3-destination-bucket* 取代為包含您要套用標籤之物件的儲存貯體名稱。
- 將 *DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET* 取代為要在其中儲存清單檔案的儲存貯體名稱。
- 將 *DOC-EXAMPLE-REPORT-BUCKET* 取代為要接收完成報告的儲存貯體名稱。

```
aws iam put-role-policy \
  --role-name S3BatchJobRole \
  --policy-name DeleteObjectTaggingBatchJobPolicy \
  --policy-document '{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
```

```
        "s3:DeleteObjectTagging",
        "s3:DeleteObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::example-s3-destination-bucket/*"
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutInventoryConfiguration"
      ],
      "Resource": "arn:aws:s3:::example-s3-destination-bucket"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:ListBucket"
      ],
      "Resource":[
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET/*",
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET/*"
      ]
    }
  ]
}'
```

2. 使用下列範例命令來建立 S3DeleteObjectTagging 任務。

   在此範例中，--report 區段中的值會指定將產生之任務報告的儲存貯體、字首、格式和範圍。此 --manifest -generator 區段會指定包含任務將對其執行動作之物件的來源儲存貯

體的相關資訊、將為任務產生的清單檔案輸出清單的相關資訊，以及用來縮小要納入清單檔案中之物件範圍的篩選條件，包括建立日期、名稱限制、大小和儲存類別。此命令也會指定任務的優先順序、IAM 角色和 AWS 區域。

如需詳細資訊，請參閱 AWS CLI 命令參考中的 create-job。

若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。將 *IAM-role* 取代為您先前建立之 IAM 角色的 ARN。

```
aws s3control create-job \
    --account-id 012345678901 \
    --operation '{
        "S3DeleteObjectTagging": {}
    }' \
    --report '{
        "Bucket":"arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET",
        "Prefix":"reports",
        "Format":"Report_CSV_20180820",
        "Enabled":true,
        "ReportScope":"AllTasks"
    }' \
    --manifest-generator '{
        "S3JobManifestGenerator": {
          "ExpectedBucketOwner": "012345678901",
          "SourceBucket": "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
          "EnableManifestOutput": true,
          "ManifestOutputLocation": {
            "ExpectedManifestBucketOwner": "012345678901",
            "Bucket": "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET",
            "ManifestPrefix": "prefix",
            "ManifestFormat": "S3InventoryReport_CSV_20211130"
          },
          "Filter": {
            "CreatedAfter": "2023-09-01",
            "CreatedBefore": "2023-10-01",
            "KeyNameConstraint": {
              "MatchAnyPrefix": [
                "prefix"
              ],
              "MatchAnySuffix": [
                "suffix"
              ]
            },
```

```
            "ObjectSizeGreaterThanBytes": 100,
            "ObjectSizeLessThanBytes": 200,
            "MatchAnyStorageClass": [
              "STANDARD",
              "STANDARD_IA"
            ]
          }
        }
      }' \
    --priority 2 \
    --role-arn IAM-role \
    --region us-east-1
```

為了回應，Amazon S3 會傳回任務 ID (例如 `00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c`)。您將需要此任務 ID 來識別、監控或修改任務。

使用 AWS SDK for Java

Specify manifest

下列範例顯示如何建立 S3 Batch Operations `S3PutObjectTagging` 任務，讓它對現有清單檔案中列出的物件執行動作。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

Example

```
package aws.example.s3control;



import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.*;

import java.util.UUID;
import java.util.ArrayList;

import static com.amazonaws.regions.Regions.US_WEST_2;
```

```
public class CreateJob {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String iamRoleArn = "IAM Role ARN";
        String reportBucketName = "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET";
        String uuid = UUID.randomUUID().toString();

        ArrayList tagSet = new ArrayList<S3Tag>();
        tagSet.add(new S3Tag().withKey("keyOne").withValue("ValueOne"));


        try {
            JobOperation jobOperation = new JobOperation()
                    .withS3PutObjectTagging(new S3SetObjectTaggingOperation()
                            .withTagSet(tagSet)
                    );

            JobManifest manifest = new JobManifest()
                    .withSpec(new JobManifestSpec()
                            .withFormat("S3BatchOperations_CSV_20180820")
                            .withFields(new String[]{
                                    "Bucket", "Key"
                            }))
                    .withLocation(new JobManifestLocation()
                            .withObjectArn("arn:aws:s3:::my_manifests/manifest.csv")
                            .withETag("60e460c9d1046e73f7dde5043ac3ae85"));
            JobReport jobReport = new JobReport()
                    .withBucket(reportBucketName)
                    .withPrefix("reports")
                    .withFormat("Report_CSV_20180820")
                    .withEnabled(true)
                    .withReportScope("AllTasks");

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.createJob(new CreateJobRequest()
                    .withAccountId(accountId)
                    .withOperation(jobOperation)
                    .withManifest(manifest)
                    .withReport(jobReport)
```

```
                    .withPriority(42)
                    .withRoleArn(iamRoleArn)
                    .withClientRequestToken(uuid)
                    .withDescription("job description")
                    .withConfirmationRequired(false)
            );

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Generate manifest

下列範例顯示如何建立 S3 Batch Operations `s3PutObjectCopy` 任務，讓它根據包括建立日期、索引鍵名稱和大小等物件篩選條件自動產生清單檔案。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

Example

```
        package aws.example.s3control;

    import com.amazonaws.AmazonServiceException;
    import com.amazonaws.SdkClientException;
    import com.amazonaws.auth.profile.ProfileCredentialsProvider;
    import com.amazonaws.services.s3control.AWSS3Control;
    import com.amazonaws.services.s3control.AWSS3ControlClient;
    import com.amazonaws.services.s3control.model.CreateJobRequest;
    import com.amazonaws.services.s3control.model.CreateJobResult;
    import com.amazonaws.services.s3control.model.JobManifestGenerator;
    import com.amazonaws.services.s3control.model.JobManifestGeneratorFilter;
    import com.amazonaws.services.s3control.model.JobOperation;
    import com.amazonaws.services.s3control.model.JobReport;
    import com.amazonaws.services.s3control.model.KeyNameConstraint;
    import com.amazonaws.services.s3control.model.S3JobManifestGenerator;
    import com.amazonaws.services.s3control.model.S3ManifestOutputLocation;
```

```
    import com.amazonaws.services.s3control.model.S3SetObjectTaggingOperation;
    import com.amazonaws.services.s3control.model.S3Tag;

    import java.time.Instant;
    import java.util.Date;
    import java.util.UUID;
    import java.util.ArrayList;

    import static com.amazonaws.regions.Regions.US_WEST_2;

    public class test {
        public static void main(String[] args) {
            String accountId = "012345678901";
            String iamRoleArn = "arn:aws:iam::012345678901:role/ROLE";
            String sourceBucketName = "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET";
            String reportBucketName = "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET";
            String manifestOutputBucketName = "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-
OUTPUT-BUCKET";
            String uuid = UUID.randomUUID().toString();
            long minimumObjectSize = 100L;

            ArrayList<S3Tag> tagSet = new ArrayList<>();
            tagSet.add(new S3Tag().withKey("keyOne").withValue("ValueOne"));

            ArrayList<String> prefixes = new ArrayList<>();
            prefixes.add("s3KeyStartsWith");

            try {
                JobOperation jobOperation = new JobOperation()
                        .withS3PutObjectTagging(new S3SetObjectTaggingOperation()
                                .withTagSet(tagSet)
                        );
                S3ManifestOutputLocation manifestOutputLocation = new
 S3ManifestOutputLocation()
                        .withBucket(manifestOutputBucketName)
                        .withManifestPrefix("manifests")
                        .withExpectedManifestBucketOwner(accountId)
                        .withManifestFormat("S3InventoryReport_CSV_20211130");

                JobManifestGeneratorFilter jobManifestGeneratorFilter = new
 JobManifestGeneratorFilter()
                        .withEligibleForReplication(true)
                        .withKeyNameConstraint(
                                new KeyNameConstraint()
```

```
                                        .withMatchAnyPrefix(prefixes))
                        .withCreatedBefore(Date.from(Instant.now()))
                        .withObjectSizeGreaterThanBytes(minimumObjectSize);

                S3JobManifestGenerator s3JobManifestGenerator = new
S3JobManifestGenerator()
                        .withEnableManifestOutput(true)
                        .withManifestOutputLocation(manifestOutputLocation)
                        .withFilter(jobManifestGeneratorFilter)
                        .withSourceBucket(sourceBucketName);

                JobManifestGenerator jobManifestGenerator = new
JobManifestGenerator()
                        .withS3JobManifestGenerator(s3JobManifestGenerator);

                JobReport jobReport = new JobReport()
                        .withBucket(reportBucketName)
                        .withPrefix("reports")
                        .withFormat("Report_CSV_20180820")
                        .withEnabled(true)
                        .withReportScope("AllTasks");

                AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                        .withCredentials(new ProfileCredentialsProvider())
                        .withRegion(US_WEST_2)
                        .build();

                CreateJobResult createJobResult = s3ControlClient.createJob(new
CreateJobRequest()
                        .withAccountId(accountId)
                        .withOperation(jobOperation)
                        .withManifestGenerator(jobManifestGenerator)
                        .withReport(jobReport)
                        .withPriority(42)
                        .withRoleArn(iamRoleArn)
                        .withClientRequestToken(uuid)
                        .withDescription("job description")
                        .withConfirmationRequired(true)
                );

                System.out.println("Created job " + createJobResult.getJobId());

            } catch (AmazonServiceException e) {
```

```
                // The call was transmitted successfully, but Amazon S3 couldn't
process
                // it and returned an error response.
                e.printStackTrace();
            } catch (SdkClientException e) {
                // Amazon S3 couldn't be contacted for a response, or the client
                // couldn't parse the response from Amazon S3.
                e.printStackTrace();
            }
        }
    }
```

使用 REST API

您可以使用 REST API 建立批次操作任務。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 CreateJob。

任務回應

如果 `CreateJob` 請求成功，Amazon S3 會傳回任務 ID。任務 ID 是 Amazon S3 自動產生的唯一識別符，讓您可以識別批次操作並監控其狀態。

當您透過 AWS CLI、開 AWS 發套件或 REST API 建立任務時，您可以設定 S3 Batch 操作以自動開始處理任務。任務會在準備好後立即開始執行，而不會等待優先順序更高的任務。

當您透過 Amazon S3 主控台建立任務時，您必須檢閱任務詳細資訊，並確認您希望在 Batch Operations 開始處理任務之前執行該任務。如果任務保持在暫停狀態的時間超過 30 天，則該任務會失敗。

S3 批次操作支援的操作

S3 批次作業支援幾個不同的作業。本節中的主題將說明每種作業。

複製物件

Copy (複製) 操作會複製資訊清單中指定的每個物件。您可以將物件複製到相同 AWS 區域中的儲存貯體，或複製到不同區域中的儲存貯體。S3 批次作業支援 Amazon S3 中可用來複製物件的大多數選項。這些選項包含設定物件中繼資料、設定許可，以及變更物件的儲存體方案。

您也可以使用 Copy (複製) 操作來複製現有的未加密物件，並將其作為加密物件寫入相同的儲存貯體中。如需詳細資訊，請參閱使用 Amazon S3 批次操作來加密現有物件。

複製物件時，您可以變更物件計算檢查總和的檢查總和演算法。如果物件沒有額外計算檢查總和，您也可以透過指定 Amazon S3 使用檢查總和演算法新增一個。如需詳細資訊，請參閱 檢查物件完整性。

如需有關在 Amazon S3 中複製物件以及必要和選用參數的詳細資訊，請參閱本指南中的 複製、移動和重新命名物件 以及《Amazon Simple Storage Service API 參考》中的 CopyObject。

法規與限制

- 所有來源物件必須位於一個儲存貯體中。
- 所有目的地物件必須位於一個儲存貯體中。
- 您必須具有來源儲存貯體的讀取許可，和目的地儲存貯體的寫入許可。
- 複製的物件大小上限可達 5 GB。
- 如果嘗試將物件從 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 類別複製到 S3 Standard 儲存類別，必須先還原這些物件。如需詳細資訊，請參閱 還原已封存的物件。
- 複製任務必須在目的地區域中建立，即您要將物件複製至其中的區域。
- 支援所有複製選項，不包括對 ETag 執行條件式檢查，以及使用客戶提供的加密金鑰 (SSE-C) 進行伺服器端加密。
- 如果未對儲存貯體進行版本控制，您將使用相同的金鑰名稱來覆寫物件。
- 物件不一定按照它們在資訊清單中出現的相同順序進行複製。對於使用版本控制的儲存貯體，如果保留當前/非當前版本順序很重要，則應該首先複製所有非當前版本。然後，在第一個任務完成後，在接下來的任務中複製目前版本。
- 不支援將物件複製到低冗餘儲存 (RRS) 類別。

使用 S3 批次操作複製物件

您可以使用 S3 批次操作來建立 PUT 複製任務，將相同帳戶內的物件複製到另一個目的地帳戶。以下各節包含如何存放及使用不同帳戶中之資訊清單的範例。在第一節中，您可以使用 Amazon S3 庫存將庫存報告交付至目的地帳戶，以供建立任務時使用，或者您也可以使用第二個範例顯示的來源或目的地帳戶中的逗號分隔值 (CSV) 資訊清單。第三個範例顯示如何使用 Copy (複製) 操作以在現有物件上啟動 S3 儲存貯體金鑰加密。

複製操作範例

- 使用傳遞至目的地帳戶的清查報告，在 AWS 帳戶 間複製物件
- 使用儲存在來源帳戶中的 CSV 資訊清單，以在 AWS 帳戶 間複製物件
- 使用 S3 批次操作以透過 S3 儲存貯體金鑰來加密物件

使用傳遞至目的地帳戶的清查報告，在 AWS 帳戶 間複製物件

您可以使用 Amazon S3 清查功能建立清查報告，並使用該報告建立要使用 S3 批次操作複製的物件清單。如需更多有關在來源帳戶或目標帳戶中使用 CSV 資訊清單的資訊，請參閱「the section called “使用 CSV 資訊清單在 AWS 帳戶 之間複製物件”」。

Amazon S3 庫存會在儲存貯體中產生物件的庫存。結果清單將推送至一個輸出檔案。已進行庫存儲存貯體稱作來源儲存貯體，存放庫存報告檔案的儲存貯體則稱作目的地儲存貯體。

Amazon S3 庫存報告可設定為交付至另一個 AWS 帳戶。當任務係建立於目的地帳戶時，這可讓 S3 批次操作能讀取庫存報告。

如需 Amazon S3 清查來源和目的地儲存貯體的詳細資訊，請參閱「來源與目的地儲存貯體」。

設定最簡單的方法是使用 AWS Management Console，但您也可以使用 REST API、AWS Command Line Interface (AWS CLI) 或 AWS 開發套件。

下列主控台程序包含設定 S3 批次操作任務許可的高階步驟。在此程序中，您從來源帳戶複製物件至目的地帳戶，並將清查報告存放於目的地帳戶中。

設定來源與目標儲存貯體分屬於不同帳戶時的 Amazon S3 庫存

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 選擇存放庫存報告的目標儲存貯體。

   決定存放庫存報告的目的地資訊清單儲存貯體。在此程序中，目的地帳戶是擁有目的地資訊清單儲存貯體和物件複製的目的儲存貯體的帳戶。
3. 設定清查以列出來源儲存貯體的物件，並將清單發佈至目標資訊清單儲存貯體。

   設定來源儲存貯體的庫存清單。在執行此操作時，您會指定欲存放清單目的地儲存貯體。來源儲存貯體的庫存報告會推送至目的地儲存貯體。在此程序中，來源帳戶為擁有來源儲存貯體的帳戶。

   如需如何使用主控台設定清查或如何加密清查清單檔案的相關資訊，請參閱 設定 Amazon S3 清查。

   選擇 CSV 為輸出格式。

   在輸入目的地儲存貯體資訊時，請選擇 Buckets in another account (另一個帳戶中的儲存貯體)。然後，輸入該目的地資訊清單儲存貯體的名稱。(選用) 您可輸入目的地帳戶的帳戶 ID。

在儲存庫存組態設定後，主控台會顯示類似以下的訊息：

Amazon S3 無法在目的地儲存貯體建立儲存貯體政策。請要求目的地儲存貯體擁有者新增以下儲存貯體政策，以允許 Amazon S3 在該儲存貯體中放置資料。

主控台隨後會顯示您可用於目的地儲存貯體的儲存貯體政策。

4. 複製該顯示於主控台中的目的地儲存貯體政策。
5. 在該目的地帳戶中，新增複製的儲存貯體政策至存放庫存報告的目的地資訊清單儲存貯體。
6. 在以 S3 批次操作信任政策為基礎的目的地帳戶中建立角色。如需信任政策的詳細資訊，請參閱「信任政策」。

   如需建立角色的詳細資訊，請參閱《IAM 使用者指南》中的建立角色以將許可委派給 AWS 服務。

   為角色輸入名稱 (範例角色使用 `BatchOperationsDestinationRoleCOPY` 為名稱)。選擇 S3 (S3) 服務，然後選擇 S3 bucket Batch Operations (S3 儲存貯體批次操作) 使用案例 (用於套用信任政策至角色)。

   然後選擇 Create policy (建立政策) 以連接以下政策至角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsDestinationObjectCOPY",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectVersionAcl",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectTagging",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::ObjectDestinationBucket/*",
```

```
        "arn:aws:s3:::ObjectSourceBucket/*",
        "arn:aws:s3:::ObjectDestinationManifestBucket/*"
      ]
    }
  ]
}
```

該角色會使用政策來授予 batchoperations.s3.amazonaws.com 權限以讀取目的地儲存貯體中的資訊清單。它也會授予權限給來源物件儲存貯體中的 GET 物件、存取控制清單 (ACL)、標籤和版本。它也會授予 PUT 物件、ACL、標籤和版本權限至目的地物件儲存貯體中。

7. 在來源帳戶中建立來源儲存貯體的儲存貯體政策，以在來源儲存貯體中授予角色至您上一步驟中的 GET 物件、ACL、標籤和版本。此步驟可允許 S3 批次操作透過信任的角色從來源儲存貯體取得物件。

   以下為來源帳戶的儲存貯體政策範例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceObjectCOPY",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::DestinationAccountNumber:role/
BatchOperationsDestinationRoleCOPY"
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::ObjectSourceBucket/*"
    }
  ]
}
```

8. 庫存報告可供使用後，請在目的地帳戶中建立一個 S3 批次操作 PUT 物件複製任務，並從目的地資訊清單儲存貯體選擇庫存報告。您需要在目的地帳戶中建立的角色 ARN。

如需建立任務的一般資訊，請參閱「建立 S3 批次操作任務」。

如需使用主控台建立任務的資訊，請參閱「建立 S3 批次操作任務」。

使用儲存在來源帳戶中的 CSV 資訊清單，以在 AWS 帳戶 間複製物件

您可以使用儲存在不同 AWS 帳戶 中的 CSV 檔案作為 S3 批次操作任務的資訊清單。若要使用 S3 庫存報告，請參閱the section called “使用清查報告在 AWS 帳戶 之間複製物件”。

以下程序顯示當使用 S3 批次操作任務將物件從來源帳戶複製到目的地帳戶時，如何使用來源帳戶中存放的 CSV 資訊清單檔案設定許可。

設定存放在不同 AWS 帳戶 的 CSV 資訊清單

1. 在以 S3 批次操作信任政策為基礎的目的地帳戶中建立角色。在此程序中，destination account (目的地帳戶) 為物件複製到的目的地帳戶。

   如需信任政策的詳細資訊，請參閱「信任政策」。

   如需建立角色的詳細資訊，請參閱《IAM 使用者指南》中的建立角色以將許可委派給 AWS 服務。

   若您使用主控台建立角色，請為角色輸入名稱 (此範例角色使用名稱 `BatchOperationsDestinationRoleCOPY`)。選擇 S3 (S3) 服務，然後選擇 S3 bucket Batch Operations (S3 儲存貯體批次操作) 使用案例 (用於套用信任政策至角色)。

   然後選擇 Create policy (建立政策) 以連接以下政策至角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsDestinationObjectCOPY",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectVersionAcl",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectTagging",
        "s3:GetObject",
        "s3:GetObjectVersion",
```

```
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::ObjectDestinationBucket/*",
        "arn:aws:s3:::ObjectSourceBucket/*",
        "arn:aws:s3:::ObjectSourceManifestBucket/*"
      ]
    }
  ]
}
```

透過使用政策，該角色可授予 batchoperations.s3.amazonaws.com 權限以讀取來源資訊清單儲存貯體中的資訊清單。它也會授予權限給來源物件儲存貯體中的 GET 物件、ACL、標籤和版本。它也會授予 PUT 物件、ACL、標籤和版本權限至目的地物件儲存貯體中。

2. 在來源帳戶中建立儲存貯體的儲存貯體政策，以在來源資訊清單儲存貯體中授予角色至您上一步驟中建立的 GET 物件、ACL、標籤和版本。

   此步驟可讓 S3 批次操作使用信任的角色讀取資訊清單。套用儲存貯體政策至包含資訊清單的儲存貯體。

   以下為套用至來源資訊清單儲存貯體的儲存貯體政策範例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceManfiestRead",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::DestinationAccountNumber:user/ConsoleUserCreatingJob",
          "arn:aws:iam::DestinationAccountNumber:role/
BatchOperationsDestinationRoleCOPY"
        ]
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
```

```
      "Resource": "arn:aws:s3:::ObjectSourceManifestBucket/*"
    }
  ]
}
```

此政策也將授予權限，以允許在目的地帳戶中建立任務的主控台使用者，可透過相同的儲存貯體政策，以在來源資訊清單儲存貯體中具有相同的權限。

3. 在來源帳戶中建立來源儲存貯體的儲存貯體政策，該原則會將您建立的角色授予來源儲存貯體中的 GET 物件、ACL、標籤和版本。然後 S3 批次操作可以透過信任的角色從來源儲存貯體取得物件。

   以下為包含來源物件的儲存貯體之儲存貯體政策範例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceObjectCOPY",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::DestinationAccountNumber:role/
BatchOperationsDestinationRoleCOPY"
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::ObjectSourceBucket/*"
    }
  ]
}
```

4. 在目的地帳戶中建立 S3 批次操作任務。您需要在目標帳戶中建立的角色 Amazon Resource Name (ARN)。

   如需建立任務的一般資訊，請參閱「建立 S3 批次操作任務」。

   如需使用主控台建立任務的資訊，請參閱「建立 S3 批次操作任務」。

使用 S3 批次操作以透過 S3 儲存貯體金鑰來加密物件

在本節中，您可以使用 Amazon S3 批次操作的複製操作，識別和啟用現有物件上的 S3 儲存貯體金鑰加密。如需 S3 儲存貯體金鑰的詳細資訊，請參閱「使用 Amazon S3 儲存貯體金鑰降低 SSE-KMS 的成本」和「設定您的儲存貯體以使用具有 SSE-KMS 的 S3 儲存貯體來獲取新物件」。

此範例涵蓋的主題如下：

主題

必要條件

若要遵循此程序中的步驟，您必須有 AWS 帳戶 和至少一個 S3 儲存貯體，以保留您的工作檔案和加密結果。您也可能會發現許多現有的 S3 批次操作文件極為有用，包括下列主題：

- S3 批次作業基礎知識
- 建立 S3 批次操作任務
- S3 批次操作支援的操作
- 管理 S3 批次作業任務

步驟 1：使用 Amazon S3 庫存取得您的物件清單

若要開始使用，請確認包含要加密之物件的 S3 儲存貯體，並取得其內容清單。Amazon S3 清查報告是執行此操作最方便且經濟實惠的方式。報告會提供儲存貯體中的物件清單，以及相關聯的中繼資料。source bucket (來源儲存貯體) 代表已進行清查的儲存貯體，destination bucket (目的地儲存貯體) 則表示您存放清查報告檔案的儲存貯體。如需 Amazon S3 清查來源和目的地儲存貯體的詳細資訊，請參閱「Amazon S3 清查」。

設定庫存最簡單的方式就是使用 AWS Management Console。但您也可以使用 REST API、AWS Command Line Interface (AWS CLI) 或 AWS 開發套件。在執行這些步驟前，請務必先登入主控台，然後開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。如果您遇到權限遭拒絕的

錯誤情況，請將儲存貯體政策新增至目的地儲存貯體。如需詳細資訊，請參閱「授予 S3 清查與 S3 分析的許可」。

使用 S3 庫存取得您的物件清單

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在導覽窗格中，選擇 Buckets (儲存貯體)，然後選擇含有要加密物件的儲存貯體。
3. 在 Management (管理) 索引標籤上，導覽至 Inventory configurations (庫存組態) 區段，然後選擇 Create inventory configuration (建立庫存組態)。
4. 為您的新清查命名、輸入目的地 S3 儲存貯體的名稱，然後選擇是否要建立 Amazon S3 的目的地字首以指派物件至儲存貯體中。
5. 在 Output format (輸出格式) 中，選擇 CSV。
6. (選用) 在其他欄位 - 選用區段中，選擇加密以及其他任何您感興趣的報告欄位。將報告傳送的頻率設定為 Daily (每日)，以縮短第一份報告傳送至您儲存貯體的時間。
7. 選擇 Create (建立) 以儲存您的組態。

Amazon S3 提供首份報告最多可能需要 48 小時，因此，請在第一份報告送達時再回來查看。收到第一份報告之後，請前往下一區段以篩選 S3 清查報告的內容。如果您不想再收到此儲存貯體的詳細清查報告，請刪除您的 S3 清查組態。否則 S3 會依每日或每週排程傳送報告。

詳細目錄清單並非所有物件的單一 point-in-time 檢視。庫存清單是儲存貯體項目的輪換快照，最終會趨於一致 (例如，清單可能不包含最近新增或刪除的物件)。結合 S3 庫存和 S3 批次操作，會在您使用靜態物件或使用兩天或更多天前建立的物件集時達到最佳效果。若要處理更新的資料，請使用 ListObjectsV2 (GET 值區) API 作業來手動建立物件清單。如有需要，請在接下來的幾天內重複此程序，或直到您的清查報告顯示出所有金鑰的所需狀態為止。

第 2 步：使用 S3 Select 來篩選物件列表

在收到 S3 清查報告之後，您可以篩選報告內容，以僅列出未使用 S3 儲存貯體金鑰加密的物件。如果您希望使用 S3 儲存貯體金鑰加密所有儲存貯體的物件，則可以忽略此步驟。不過，在此階段篩選 S3 清查報告，可為您節省重新加密先前加密物件所需的時間和費用。

雖然下列步驟顯示如何使用 Amazon S3 Select 來篩選，但您也可以使用 Amazon Athena。若要決定要使用哪個工具，請查看 S3 清查報告的 `manifest.json` 檔案。此檔案會列出與該報告相關聯的資料檔案數量。如果數量龐大，請使用 Amazon Athena，因為其會在多個 S3 物件上運行，而 S3 Select 一次僅可在一個物件上運行。如需搭配使用 Amazon S3 和 Athena 的詳細資訊，請參閱「使用

Amazon Athena 查詢 Amazon S3 庫存」以及部落格文章《使用 Amazon S3 批次操作來加密物件》中的使用 Athena。

使用 S3 Select 來篩選 S3 清查報告

1. 開啟您清查報告的 manifest.json 檔案，並查看 JSON 的 fileSchema 區段。這會通知您在資料上執行的查詢。

   下面的 JSON 是範例 manifest.json 檔案，示範儲存貯體上已啟用版本控制的 CSV 格式庫存。依據您設定清查報告的方式，資訊清單可能會有所不同。

   ```
   {
     "sourceBucket": "batchoperationsdemo",
     "destinationBucket": "arn:aws:s3:::testbucket",
     "version": "2021-05-22",
     "creationTimestamp": "1558656000000",
     "fileFormat": "CSV",
     "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker,
  BucketKeyStatus",
     "files": [
       {
         "key": "demoinv/batchoperationsdemo/DemoInventory/data/009a40e4-
  f053-4c16-8c75-6100f8892202.csv.gz",
         "size": 72691,
         "MD5checksum": "c24c831717a099f0ebe4a9d1c5d3935c"
       }
     ]
   }
   ```

   如果未在儲存貯體上啟動版本控制，或者如果您選擇執行最新版本的報告，則 fileSchema 是 Bucket、Key 以及 BucketKeyStatus。

   如果版本控制已啟動，依據您設定清查報告的方式，fileSchema 可能包含以下項目：Bucket、Key、VersionId、IsLatest、IsDeleteMarker、BucketKeyStatus。因此，在您執行查詢時，請注意第 1、2、3 和 6 欄。

   除了要搜尋的欄位 (即 BucketKeyStatus) 之外，S3 批次操作還需要輸入儲存貯體、金鑰及版本 ID，以執行任務。您不需要版本 ID 欄位，但在啟用了版本控制的儲存貯體上操作時，該欄位有助於識別它。如需詳細資訊，請參閱「使用已啟用版本控制之儲存貯體中的物件」。

2. 找出清查報告的資料檔案。manifest.json 物件會列出 files (檔案) 下的資料檔案。

3. 在 S3 主控台中找出並選取資料檔案後，請選擇 Actions (動作)，然後選擇 Query with S3 Select (使用 S3 Select 查詢)。
4. 保留選取預設 CSV、Comma (逗號) 以及 GZIP 等欄位，然後選擇 Next (下一步)。
5. 若要在繼續之前檢閱您的清查報告格式，請選擇 Show file preview (顯示檔案預覽)。
6. 在 SQL 表達式欄位中輸入要參照的資料欄，然後選擇 Run SQL (執行 SQL)。下列運算式會傳回未設定 S3 儲存貯體金鑰之所有物件的第 1 至 3 欄。

   ```
   select s._1, s._2, s._3 from s3object s where s._6 = 'DISABLED'
   ```

   以下為結果範例。

   ```
         batchoperationsdemo,0100059%7Ethumb.jpg,lsrtIxksLu0R0ZkYPL.LhgD5caTYn6vu
         batchoperationsdemo,0100074%7Ethumb.jpg,sd2M60g6Fdazoi6D5kNARIE7KzUibmHR
         batchoperationsdemo,0100075%7Ethumb.jpg,TLYESLnl1mXD5c4BwiOIinqFrktddkoL
         batchoperationsdemo,0200147%7Ethumb.jpg,amufzfMi_fEw0Rs99rxR_HrDFlE.l3Y0
         batchoperationsdemo,0301420%7Ethumb.jpg,9qGU2SEscL.C.c_sK89trmXYIwooABSh
         batchoperationsdemo,0401524%7Ethumb.jpg,ORnEWNuB1QhHrrYAGFsZhbyvEYJ3DUor
         batchoperationsdemo,200907200065HQ
   %7Ethumb.jpg,d8LgvIVjbDR5mUVwW6pu9ahTfReyn5V4
         batchoperationsdemo,200907200076HQ
   %7Ethumb.jpg,XUT25d7.gK40u_GmnupdaZg3BVx2jN40
         batchoperationsdemo,201103190002HQ
   %7Ethumb.jpg,z.2sVRh0myqVi0BuIrngWlsRPQdb7qOS
   ```

7. 下載結果，將結果儲存為 CSV 格式，然後將結果上傳到 Amazon S3 作為 S3 批次操作任務的物件清單。
8. 如果您有多個清單檔案，也請在其上執行 Query with S3 Select (使用 S3 Select 查詢)。根據結果大小，您可以合併清單並執行單一 S3 批次操作任務，或將每個清單當成個別任務來執行。

   在您決定要執行的任務數量時，請考量執行每個 S3 批次操作任務的價格。

步驟 3：設定並執行 S3 批次操作任務

此時，您已擁有經過篩選的 S3 物件的 CSV 清單，可以開始 S3 批次操作任務，以使用 S3 儲存貯體金鑰來加密物件。

任務指的是所提供物件的列表 (清單)、執行的操作以及指定的參數。加密這組物件的最簡單方式就是使用 PUT 複製操作，並指定與資訊清單中所列物件相同的目的地字首。這會覆寫無版本控制的儲存貯體中的現有物件，或是在開啟版本控制的情況下，建立較新的物件加密版本。

在複製物件時，請指定 Amazon S3 應使用 SSE-KMS 加密和 S3 來加密物件。此任務會複製物件，因此無論您將物件新增至 S3 的最初時間為何，所有物件都會在完成時顯示更新的建立日期。也可以為您的物件集指定其他屬性，作為 S3 批次操作任務的一部分，包括物件標籤和儲存類別。

子步驟

- 設定 IAM 政策
- 設定批次操作 IAM 角色
- 開啟現有儲存貯體的 S3 儲存貯體金鑰
- 建立批次操作任務
- 執行批次操作任務
- 須知事項

設定 IAM 政策

1. 在 https://console.aws.amazon.com/iam/ 中開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Policies (政策)，然後選擇 Create Policy (建立政策)。
3. 請選擇 JSON 索引標籤。選擇 Edit policy (編輯政策)，然後新增出現在下列程式碼區塊中的 IAM 政策範例。

   將政策範例複製到 IAM 主控台後，請取代下列項目：

   a. 用您的來源儲存貯體名稱取代 *SOURCE_BUCKET_FOR_COPY*。
   b. 用您的目的地儲存貯體名稱取代 *DESTINATION_BUCKET_FOR_COPY*。
   c. 以您的資訊清單物件名稱取代 *MANIFEST_KEY*。
   d. 以您要儲存報告的儲存貯體名稱取代 *REPORT_BUCKET*。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CopyObjectsToEncrypt",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectTagging",
        "s3:PutObjectAcl",
```

```
          "s3:PutObjectVersionTagging",
          "s3:PutObjectVersionAcl",
          "s3:GetObject",
          "s3:GetObjectAcl",
          "s3:GetObjectTagging",
          "s3:GetObjectVersion",
          "s3:GetObjectVersionAcl",
          "s3:GetObjectVersionTagging"
        ],
        "Resource": [
          "arn:aws:s3:::SOURCE_BUCKET_FOR_COPY/*",
          "arn:aws:s3:::DESTINATION_BUCKET_FOR_COPY/*"
        ]
      },
      {
        "Sid": "ReadManifest",
        "Effect": "Allow",
        "Action": [
          "s3:GetObject",
          "s3:GetObjectVersion"
        ],
        "Resource": "arn:aws:s3:::MANIFEST_KEY"
      },
      {
        "Sid": "WriteReport",
        "Effect": "Allow",
        "Action": [
          "s3:PutObject"
        ],
        "Resource": "arn:aws:s3:::REPORT_BUCKET/*"
      }
    ]
}
```

4. 選擇 Next: Tags (下一步：標籤)。
5. 新增您想要的標籤 (選用)，然後選擇 Next: Review (下一步：檢閱)。
6. 新增政策名稱並選擇性輸入描述，然後選擇 Create policy (建立政策)。
7. 選擇 Review policy (檢閱政策)，然後選擇 Save changes (儲存變更)。
8. 現在已完成 S3 批次操作政策，主控台會將您導回 IAM Policies (政策) 頁面。篩選政策名稱，選擇政策名稱左側的按鈕，再選擇 Policy actions (政策動作)，然後選擇 Attach (連接)。

若要將新建立的政策附加到 IAM 角色，請選取帳戶中適當的使用者、群組或角色，然後選擇 Attach policy (連接政策)。這會引導您返回 IAM 主控台。

設定批次操作 IAM 角色

1. 在 IAM 主控台的導覽窗格中，選擇 [角色]，然後選擇 [建立角色]。
2. 選擇 AWS 服務、S3 以及 S3 批次操作。然後選擇 Next: Permissions (下一步：許可)。
3. 開始輸入您剛建立的 IAM 政策名稱。選取政策名稱出現時的核取方塊，然後選擇 Next: Tags (下一步：標籤)。
4. (選用) 在本練習中新增標籤，或保持金鑰和值欄位空白。選擇 Next:Review (下一步：檢閱)。
5. 輸入角色名稱，然後接受預設描述或新增自己的描述。選擇 Create role (建立角色)。
6. 請確定建立任務的使用者具有以下範例中的權限。

   以您的 AWS 帳戶 ID 取代 *{ACCOUNT-ID}*，並以您要套用至 IAM 角色的名稱 (稍後將在 Batch 批次操作建立步驟中建立該名稱) 來取代 *{IAM_ROLE_NAME}*。如需詳細資訊，請參閱 授予 Amazon S3 批次操作的許可。

```
{
"Sid": "AddIamPermissions",
"Effect": "Allow",
"Action": [
"iam:GetRole",
"iam:PassRole"
],
"Resource": "arn:aws:iam::ACCOUNT-ID:role/IAM_ROLE_NAME"
}
```

開啟現有儲存貯體的 S3 儲存貯體金鑰

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在儲存貯體清單中，選擇您要開啟 S3 儲存貯體金鑰的儲存貯體。
3. 選擇 Properties (屬性)。
4. 在 Default encryption (預設加密) 底下，選擇 Edit (編輯)。
5. 在加密類型下，您可以選擇 Amazon S3 受管金鑰 (SSE-S3) 或 AWS Key Management Service 金鑰 (SSE-KMS)。

6. 若您選擇 AWS Key Management Service 金鑰 (SSE-KMS)，在 AWS KMS key 之下，您可以透過下列其中一個選項指定 AWS KMS 金鑰。

   - 若要從可用 KMS 金鑰清單中選擇，請選擇從您的 AWS KMS 金鑰中選擇。從可用金鑰清單中，然後選擇與您儲存貯體相同區域中的對稱加密 KMS 金鑰。AWS 受管金鑰 (aws/s3) 和您的客戶受管金鑰都會出現在清單中。
   - 若要輸入 KMS 金鑰 ARN，請選擇輸入 AWS KMS 金鑰 ARN，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。
   - 若要在 AWS KMS 主控台建立新的客戶受管金鑰，請選擇建立 KMS 金鑰。

7. 在 Bucket Key (儲存貯體金鑰) 底下，選擇 Enable (啟用)，然後選擇 Save changes (儲存變更)。

現在，S3 儲存貯體金鑰已在儲存貯體層級開啟，上傳、修改或複製到此儲存貯體的物件將預設會繼承此加密組態。其中就包含使用 Amazon S3 批次操作複製的物件。

建立批次操作任務

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在導覽窗格中，選擇 Batch Operations (批次操作)，然後選擇 Create Job (建立任務)。
3. 選擇您存放物件的 Region (區域)，然後選擇 CSV 作為資訊清單類型。
4. 輸入路徑或導覽至您先前從 S3 Select (或 Athena) 結果建立的 CSV 資訊清單檔案。如果您的資訊清單包含版本 ID，請選取該方塊。選擇 Next (下一步)。
5. 選擇 Copy (複製) 操作，然後選擇複製目的地儲存貯體。您可以保持停用伺服器端加密。只要儲存貯體目的地已啟用 S3 儲存貯體金鑰，複製操作便會在目的地儲存貯體中套用 S3 儲存貯體金鑰。
6. (選用) 視需要選擇儲存類別和其他參數。您在此步驟中指定的參數會套用至對資訊清單中所列物件執行的所有操作。選擇下一步。
7. 若要設定伺服器端加密，請執行下列步驟：

   a. 在伺服器端加密下，選擇下列其中一項：

      - 若要在將物件存放到 Amazon S3 時，保留預設伺服器端加密的儲存貯體設定，請選擇不指定加密金鑰。只要儲存貯體目的地已啟用 S3 儲存貯體金鑰，複製操作便會在目的地儲存貯體中套用 S3 儲存貯體金鑰。

> **Note**
>
> 若指定目的地的儲存貯體政策要求物件先加密，再存放到 Amazon S3，則您必須指定加密金鑰。否則，便無法將物件複製到目的地。

- 若要先加密物件再存放到 Amazon S3，請選擇指定加密金鑰。

b. 在加密設定下，若您選擇指定加密金鑰，則必須選擇使用目的地儲存貯體設定做為預設加密或覆寫預設加密的目的地儲存貯體設定。

c. 若您選擇覆寫預設加密的目的地儲存貯體設定，則您必須設定下列加密設定。

i. 在加密類型下，您必須選擇 Amazon S3 受管金鑰 (SSE-S3) 或 AWS Key Management Service 金鑰 (SSE-KMS)。SSE-S3 使用目前最強大的其中一種區塊加密法，也就是 256 位元進階加密標準 (AES-256)，來加密每個物件。SSE-KMS 可以讓您更完善地控制金鑰。如需詳細資訊，請參閱 使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密 及 使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS)。

ii. 若您選擇 AWS Key Management Service 金鑰 (SSE-KMS)，在 AWS KMS key 之下，您可以透過下列其中一個選項指定 AWS KMS key。

- 從可用 KMS 金鑰清單中進行選擇，選擇從您的 AWS KMS keys 中選擇，然後選擇與您儲存貯體相同區域中的對稱加密 KMS 金鑰。AWS 受管金鑰 (aws/s3) 和您的客戶受管金鑰都會出現在清單中。
- 若要輸入 KMS 金鑰 ARN，請選擇輸入 AWS KMS 金鑰 ARN，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。
- 若要在 AWS KMS 主控台建立新的客戶受管金鑰，請選擇建立 KMS 金鑰。

iii. 在 Bucket Key (儲存貯體金鑰) 底下，選擇 Enable (啟用)。複製操作會在目的地儲存貯體套用至 S3 儲存貯體金鑰。

8. 為您的任務提供描述 (或保留預設值)、設定其優先順序層級、選擇報告類型，並指定 Path to completion report destination (完成報告目的地的路徑)。
9. 在 Permissions (許可) 區段中，請務必選擇您先前定義的批次操作 IAM 角色。選擇 Next (下一步)。
10. 請在 Review (檢閱) 中驗證設定。如需變更，請選擇 Previous (上一步)。確認批次操作設定後，請選擇 Create job (建立任務)。

    如需詳細資訊，請參閱「建立 S3 批次操作任務」。

執行批次操作任務

設定精靈會自動返回 Amazon S3 主控台的 S3 Batch Operations (S3 批次操作) 區段。在 S3 開始該過程後，您的新任務會從 New (全新) 狀態轉換為 Preparing (正在準備) 狀態。在 Preparing (準備) 狀態期間，S3 會讀取工作的資訊清單、檢查工作是否有錯誤，並計算物件數量。

1. 選擇 Amazon S3 主控台中的重新整理按鈕，以檢查進度。依資訊清單大小而定，讀取可能需要幾分鐘或幾小時。
2. 在 S3 完成讀取任務的資訊清單後，任務會移 至Awaiting your confirmation (等待確認) 狀態。選擇任務 ID 左側的選項按鈕，然後選擇 Run job (執行任務)。
3. 檢查任務的設定，然後選擇右下角的 Run job (執行任務)。

   在任務開始執行後，您可以選擇重新整理按鈕，透過主控台儀表板檢視或選取特定任務來檢查進度。
4. 在任務完成時，您可以檢視 Successful (成功) 和 Failed (失敗) 的物件計數，以確認所有內容都如預期般執行。如果您已啟用任務報告，請檢查任務報告，以了解任何操作失敗的確切原因。

   您也可以使用 AWS CLI、AWS 開發套件或 Amazon S3 REST API 來執行這些步驟。如需追蹤任務狀態和完成報告的詳細資訊，請參閱「追蹤任務狀態和完成報告」。

須知事項

在您使用 S3 批次操作透過 S3 儲存貯體金鑰來加密物件時，請考慮下列問題：

- 除了與 S3 批次操作代表您執行之作業相關聯的任何費用之外，還需支付 S3 批次操作任務、物件及請求之費用，包括資料傳輸、請求及其他費用。如需詳細資訊，請參閱 Amazon S3 定價。
- 如果您使用無版本控制的儲存貯體，則執行的每個 S3 批次操作任務都會建立物件的新加密版本。同時也會保留先前沒有配置 S3 儲存貯體金鑰的版本。若要刪除舊版本，請如 生命週期組態元素 之描述，設定非最新版本 S3 生命週期過期政策。
- 複製操作會建立具有新建立日期的新物件，這會影響封存等生命週期動作。若您複製儲存貯體中的所有物件，則所有新副本皆會有相同或類似的建立日期。若要進一步識別這些物件，並為多個資料子集建立不同的生命週期規則，請考慮使用物件標籤。

Summary

在此區段中，您已排序現有物件，以篩選出已加密的資料。然後，您使用 S3 批次操作將現有資料複製到啟動的 S3 儲存貯體金鑰中，以將 S3 儲存貯體金鑰功能套用至未加密物件。此程序可以節省您的時間和金錢，同時允許您完成如加密所有現有物件等相關操作。

如需 S3 批次操作的詳細資訊，請參閱「在 Amazon S3 物件上執行大規模批次操作」。

如需使用 AWS CLI 和 AWS SDK for Java 的標籤顯示複製操作範例，請參閱「建立具有任務標籤 (用於標示) 的批次作業任務」。

調用 AWS Lambda 函數

In voke AWS Lambda 函數會啟動 AWS Lambda 函數，以對資訊清單中列出的物件執行自訂動作。本節說明如何建立 Lambda 函數以搭配 S3 批次作業使用，以及如何建立任務來叫用函數。S3 批次操作任務會使用 `LambdaInvoke` 操作，對資訊清單中列出的每個物件執行 Lambda 函數。

您可以使用 AWS Management Console、 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 REST API，針對 Lambda 使用 S3 Batch 操作。如需有關使用 Lambda 的詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 AWS Lambda入門。

下列各節說明如何開始搭配 Lambda 使用 S3 批次作業。

主題

- 搭配 Amazon S3 批次作業使用 Lambda
- 建立 Lambda 函數以搭配 S3 批次作業使用
- 建立 S3 批次作業任務以叫用 Lambda 函數
- 在 Lambda 資訊清單中提供工作層級的資訊
- 從 S3 批次操作教學課程學習

搭配 Amazon S3 批次作業使用 Lambda

搭配使用 S3 Batch 操作時 AWS Lambda，您必須建立專門用於 S3 Batch 操作的新 Lambda 函數。您無法搭配 S3 批次作業重複使用現有基於 Amazon S3 事件的函數。事件函數只能接收訊息；不能傳回訊息。搭配 S3 批次作業使用的 Lambda 函數必須接受並傳回訊息。如需將 Lambda 與 Amazon S3 事件搭配使用的詳細資訊，請參閱AWS Lambda 開發人員指南中的AWS Lambda 與 Amazon S3 搭配使用。

您建立 S3 批次作業任務來叫用 Lambda 函數。此任務對資訊清單中列出的所有物件執行相同的 Lambda 函數。您可以控制在處理清單中的物件時，使用 Lambda 函數的哪些版本。S3 批次作業支援不合格的 Amazon Resource Name (ARN)、別名和特定版本。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的介紹 AWS Lambda 版本控制。

如果您為 S3 批次作業任務提供的函數 ARN 使用別名或 `$LATEST` 限定詞，而且您更新其中任一項所指向的版本，則 S3 批次作業會開始呼叫 Lambda 函數的新版本。當您想要隨著大型工作更

新部分功能，此功能會很有用。如果您不希望 S3 批次作業變更所使用的版本，請在建立任務時在 FunctionARN 參數中提供特定版本。

使用 Lambda 和 Amazon S3 Batch Operations 搭配目錄儲存貯體

目錄儲存貯體是一種 Amazon S3 儲存貯體，這是專為需要一致的個位數毫秒延遲的工作負載或效能關鍵應用程式所設計的類型。如需詳細資訊，請參閱目錄儲存貯體。

使用 Amazon S3 Batch Operations 調用對目錄儲存貯體執行動作的 Lambda 函數時，須遵循特殊需求。例如，您必須使用更新的 JSON 結構描述來建構 Lambda 請求，並在建立任務時指定 `InvocationSchemaVersion` **`2.0`**。此更新的結構描述可讓您為 `UserArguments` 指定選用的索引鍵值配對，您可以用它來修改現有 Lambda 函數的特定參數。如需詳細資訊，請參閱使用 S3 Batch 操作和AWS 儲存部落格中的自動化 Amazon S3 目錄儲存貯體 AWS Lambda中的物件處理。

回應代碼和結果代碼

S3 Batch 操作會使用一或多個索引鍵叫用 Lambda 函數，每個索引鍵都有`TaskID`關聯。S3 Batch 操作需要 Lambda 函數提供每個索引鍵的結果程式碼。在要求中傳送的任何工作 ID，但未隨每個索引鍵結果碼傳回，都會從`treatMissingKeysAs`欄位中提供結果代碼。 `treatMissingKeysAs`是選擇性的要求欄位，且預設為`TemporaryFailure`。下表包含欄位的其他可能結果代碼和`treatMissingKeysAs`值。

回應代碼	描述
`Succeeded`	任務正常完成。如果您請求工作完成報告，即會在報告中包含任務的結果字串。
`TemporaryFailure`	任務遇到暫時性的失敗，並且將在工作完成之前重新推動。會忽略結果字串。如果這是最後一次的重新推動，即會在最終報告中包含錯誤訊息。
`PermanentFailure`	任務遇到永久的失敗。如果您請求工作完成報告，即會將任務標示為 `Failed`，並包含錯誤訊息字串。會忽略來自失敗任務的結果字串。

建立 Lambda 函數以搭配 S3 批次作業使用

本節提供您必須搭配 Lambda 函數使用的範例 AWS Identity and Access Management (IAM) 許可。還包含一個搭配 S3 批次作業使用的 Lambda 函數範例。如果您之前從未建立過 Lambda 函數，請參閱AWS Lambda 開發人員指南中的教學課程： AWS Lambda 搭配 Amazon S3 使用。

您必須建立專門搭配 S3 批次作業使用的 Lambda 函數。您無法重複使用現有基於 Amazon S3 事件的 Lambda 函數。這是因為用於 S3 批次作業的 Lambda 函數必須接受並傳回特殊的資料欄位。

> **Important**
>
> AWS Lambda 用 Java 編寫的函數接受RequestHandler或RequestStreamHandler處理程序接口。但是，若要支援 S3 Batch 操作請求和回應格式， AWS Lambda 需要用於自訂序列化和請求和回應還原序列化的`RequestStreamHandler`介面。該接口允許 Lambda 將 InputStream 和 OutputStream 傳遞給 Java `handleRequest` 方法。
>
> 搭配 S3 批次作業使用 Lambda 函數時，請務必使用 `RequestStreamHandler` 界面。如果您使用 `RequestHandler` 界面，批次任務會失敗，完成報告中出現「Lambda 承載中傳回無效的 JSON」。
>
> 如需詳細資訊，請參閱《AWS Lambda 使用者指南》中的處理常式界面。

IAM 權限範例

以下是搭配 S3 批次作業使用 Lambda 函數所需的 IAM 權限範例。

Example — S3 批次作業信任政策

以下是可用於批次作業 IAM 角色的信任政策範例。您在建立任務時指定此 IAM 角色，以准許批次作業擔任 IAM 角色。

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

```
}
```

Example — Lambda IAM 政策

以下 IAM 政策範例准許 S3 批次作業叫用 Lambda 函數和讀取輸入資訊清單。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "BatchOperationsLambdaPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject",
                "lambda:InvokeFunction"
            ],
            "Resource": "*"
        }
    ]
}
```

範例請求和回應

本節提供 Lambda 函數的請求和回應範例。

Example 請求

以下是 Lambda 函數請求的 JSON 範例。

```
{
    "invocationSchemaVersion": "1.0",
    "invocationId": "YXNkbGZqYWRmaiBhc2RmdW9hZHNmZGpmaGFzbGtkaGZza2RmaAo",
    "job": {
        "id": "f3cc4f60-61f6-4a2b-8a21-d07600c373ce"
    },
    "tasks": [
        {
            "taskId": "dGFza2lkZ29lc2hlcmUK",
            "s3Key": "customerImage1.jpg",
            "s3VersionId": "1",
            "s3BucketArn": "arn:aws:s3:us-east-1:0123456788:awsexamplebucket1"
        }
```

```
    ]
}
```

Example 回應

以下是 Lambda 函數回應的 JSON 範例。

```
{
  "invocationSchemaVersion": "1.0",
  "treatMissingKeysAs" : "PermanentFailure",
  "invocationId" : "YXNkbGZqYWRmaiBhc2RmdW9hZHNmZGpmaGFzbGtkaGZza2RmaAo",
  "results": [
    {
      "taskId": "dGFza2lkZ29lc2hlcmUK",
      "resultCode": "Succeeded",
      "resultString": "[\"Mary Major", \"John Stiles\"]"
    }
  ]
}
```

S3 批次作業的 Lambda 函數範例

以下範例 Python Lambda 從版本控制的物件中移除了刪除標記。

如範例所示，來自 S3 批次作業的金鑰以 URL 編碼。若要將 Amazon S3 與其他 AWS 服務搭配使用，請務必將從 S3 Batch 操作傳遞的金鑰進行 URL 解碼。

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
```

```
                to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a delete
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete marker."
            )

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]["HTTPHeaders"].get(
                "x-amz-delete-marker", "false"
            )
```

```
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
                    # Mark request timeout as a temporary failure so it will be
retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
                        )
                        logger.info(result_string)
                    else:
                        raise
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
                    f"present or is not 'true'."
                )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
```

```
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

建立 S3 批次作業任務以叫用 Lambda 函數

建立 S3 批次作業任務來叫用 Lambda 函數時，您必須提供下列項目：

- Lambda 函數的 ARN (可能包含函數別名或特定版本號碼)
- 獲准叫用函數的 IAM 角色
- 動作參數 `LambdaInvokeFunction`

如需有關建立 S3 批次作業任務的詳細資訊，請參閱建立 S3 批次操作任務和S3 批次操作支援的操作。

下列範例建立 S3 批次操作任務來使用 AWS CLI叫用 Lambda 函數。

```
aws s3control create-job
    --account-id <AccountID>
    --operation  '{"LambdaInvoke": { "FunctionArn":
 "arn:aws:lambda:Region:AccountID:function:LambdaFunctionName" } }'
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":
{"ObjectArn":"arn:aws:s3:::ManifestLocation","ETag":"ManifestETag"}}'
    --report
 '{"Bucket":"arn:aws:s3:::awsexamplebucket1","Format":"Report_CSV_20180820","Enabled":true,"Pre
    --priority 2
    --role-arn arn:aws:iam::AccountID:role/BatchOperationsRole
    --region Region
    --description “Lambda Function"
```

在 Lambda 資訊清單中提供工作層級的資訊

當您將 AWS Lambda 函數與 S3 Batch 操作搭配使用時，您可能需要額外的資料隨附於所操作的每個工作/金鑰。例如，您可能希望同時提供來源物件金鑰與全新的物件金鑰。這樣 Lambda 函數就能以新名稱，將來源金鑰複製到新的 S3 儲存貯體。依預設，在任務的輸入資訊清單中，Amazon S3 批次作業只允許您指定目的地儲存貯體和來源金鑰清單。下列說明如何在資訊清單中包含其他資料，以便執行更複雜的 Lambda 函數。

若要在 S3 批次作業資訊清單中指定每個金鑰的參數，以用於 Lambda 函數程式碼中，請使用以下 URL 編碼的 JSON 格式。`key` 欄位視同 Amazon S3 物件金鑰傳遞給 Lambda 函數。但 Lambda 函數可以轉譯此欄位，以包含其他值或多個金鑰，如下所示。

> **Note**
>
> 資訊清單中 `key` 欄位的字元數上限是 1,024。

Example — 以 JSON 字串取代「Amazon S3 金鑰」的資訊清單

必須提供以 URL 編碼的版本給 S3 批次作業。

```
my-bucket,{"origKey": "object1key", "newKey": "newObject1Key"}
my-bucket,{"origKey": "object2key", "newKey": "newObject2Key"}
my-bucket,{"origKey": "object3key", "newKey": "newObject3Key"}
```

Example — 以 URL 編碼的資訊清單

必須提供這個以 URL 編碼的版本給 S3 批次作業。非 URL 編碼的版本不會運作。

```
my-bucket,%7B%22origKey%22%3A%20%22object1key%22%2C%20%22newKey%22%3A
%20%22newObject1Key%22%7D
my-bucket,%7B%22origKey%22%3A%20%22object2key%22%2C%20%22newKey%22%3A
%20%22newObject2Key%22%7D
my-bucket,%7B%22origKey%22%3A%20%22object3key%22%2C%20%22newKey%22%3A
%20%22newObject3Key%22%7D
```

Example — 以資訊清單格式將結果寫入任務報告的 Lambda 函數

這個 URL 編碼的資訊清單範例包含以管道分隔的物件索引鍵，供下列 Lambda 函數進行剖析。

```
my-bucket,object1key%7Clower
```

```
my-bucket,object2key%7Cupper
my-bucket,object3key%7Creverse
my-bucket,object4key%7Cdelete
```

此 Lambda 函數示範如何剖析已編碼成 S3 批次作業資訊清單的管道分隔任務。任務會指出要套用至指定物件的修訂版作業。

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.resource("s3")


def lambda_handler(event, context):
    """
    Applies the specified revision to the specified object.

    :param event: The Amazon S3 batch event that contains the ID of the object to
                  revise and the revision type to apply.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]
    # The revision type is packed with the object key as a pipe-delimited string.
    obj_key, revision = parse.unquote(task["s3Key"], encoding="utf-8").split("|")
    bucket_name = task["s3BucketArn"].split(":")[-1]

    logger.info("Got task: apply revision %s to %s.", revision, obj_key)
```

```
    try:
        stanza_obj = s3.Bucket(bucket_name).Object(obj_key)
        stanza = stanza_obj.get()["Body"].read().decode("utf-8")
        if revision == "lower":
            stanza = stanza.lower()
        elif revision == "upper":
            stanza = stanza.upper()
        elif revision == "reverse":
            stanza = stanza[::-1]
        elif revision == "delete":
            pass
        else:
            raise TypeError(f"Can't handle revision type '{revision}'.")

        if revision == "delete":
            stanza_obj.delete()
            result_string = f"Deleted stanza {stanza_obj.key}."
        else:
            stanza_obj.put(Body=bytes(stanza, "utf-8"))
            result_string = (
                f"Applied revision type '{revision}' to " f"stanza {stanza_obj.key}."
            )

        logger.info(result_string)
        result_code = "Succeeded"
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            result_code = "Succeeded"
            result_string = (
                f"Stanza {obj_key} not found, assuming it was deleted "
                f"in an earlier revision."
            )
            logger.info(result_string)
        else:
            result_code = "PermanentFailure"
            result_string = (
                f"Got exception when applying revision type '{revision}' "
                f"to {obj_key}: {error}."
            )
            logger.exception(result_string)
    finally:
        results.append(
            {
```

```
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

從 S3 批次操作教學課程學習

下列教學課 end-to-end 程介紹使用 Lambda 的某些 Batch 作業工作的完整程序。

- 教學課程：使用 S3 Batch 操作進行 Batch 轉碼影片，以及 AWS LambdaAWS Elemental MediaConvert

取代所有物件標籤

取代所有物件標籤操作會取代資訊清單中列出的每個物件上的 Amazon S3 物件標籤。Amazon S3 物件標籤是一組鍵值字串，可用於存放物件的中繼資料。

若要建立「取代所有物件標籤」任務，請提供要套用的標籤組。S3 批次操作會將相同的標籤集套用至每個物件。您提供的標籤集會取代已與資訊清單中的物件相關聯的任何標籤組。S3 批次操作不支援將標籤新增至物件又保留現有標籤。

若您資訊清單中的物件位於啟用版本控制的儲存貯體，您可以將標籤組套用至每個物件的特定版本。您可以透過為資訊清單中的每個物件指定版本 ID 來執行此操作。如果您沒有包含任何物件的版本 ID，則 S3 批次操作會將標籤組套用至每個物件的最新版本。

法規與限制

- 您指定用於執行批次操作任務的 AWS Identity and Access Management (IAM) 角色，必須具有執行基礎 Amazon S3 取代所有物件標籤操作的許可。如需有關所需許可的詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的 PutObjectTagging。

- S3 批次操作使用 Amazon S3 PutObjectTagging 操作，將標籤套用至資訊清單中的每個物件。套用至基礎操作的所有約束與限制也適用於 S3 批次操作任務。

如需使用主控台建立任務的詳細資訊，請參閱建立 S3 批次操作任務。

如需有關物件標記的資訊，請參閱本指南中的使用標籤分類儲存空間，並參閱 Amazon Simple Storage Service API 參考中的 PutObjectTagging、GetObjectTagging 和 DeleteObjectTagging。

刪除所有物件標籤

刪除所有物件標籤操作會移除目前與資訊清單中列出的物件關聯的所有 Amazon S3 物件標籤組。S3 批次操作不支援從物件中刪除標籤，同時保留其他標籤。

如果資訊清單中的物件位於已建立版本的儲存貯體中，您可以從特定版本的物件中移除標籤組。可以透過為資訊清單中的每個物件指定版本 ID 來執行此操作。如果您未包含物件的版本 ID，S3 批次操作會從每個物件的最新版本中移除標籤組。

如需批次操作資訊清單的詳細資訊，請參閱指定資訊清單。

> **Warning**
>
> 執行此任務會移除資訊清單中列出的每個物件上的所有物件標籤組。

法規與限制

- 您指定用於執行任務的 AWS Identity and Access Management (IAM) 角色必須具有執行基礎 Amazon S3 刪除物件標記操作的許可。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的 DeleteObjectTagging。
- S3 批次操作使用 Amazon S3 DeleteObjectTagging，從資訊清單中的每個物件中移除標籤組。套用至基礎操作的所有約束與限制也適用於 S3 批次操作任務。

如需有關建立任務的詳細資訊，請參閱建立 S3 批次操作任務。

如需有關物件標記的詳細資訊，請參閱本指南中的取代所有物件標籤，以及 Amazon Simple Storage Service API 參考中的 PutObjectTagging、GetObjectTagging 和 DeleteObjectTagging。

取代存取控制清單

取代存取控制清單 (ACL) 操作會取代資訊清單中列出的每個物件的 Amazon S3 存取控制清單 (ACL)。您可以使用 ACL 定義可存取物件的人員，以及他們可執行的操作。

S3 批次作業支援您定義的自訂 ACL，以及 Amazon S3 提供的固定 ACL (隨附一組預先定義的存取許可)。

若您資訊清單中的物件位於啟用版本控制的儲存貯體，您可以將 ACL 套用至每個物件的特定版本。您可以透過為資訊清單中的每個物件指定版本 ID 來執行此操作。如果您沒有包含任何物件的版本 ID，則 S3 批次作業會將 ACL 套用至物件的最新版本。

有關 Amazon S3 中 ACL 的更多資訊，存取控制清單 (ACL) 概觀。

S3 封鎖公有存取權

若要限制公開存取儲存貯體中的所有物件，您應該使用 Amazon S3 封鎖公開存取，而不是 S3 批次操作。封鎖公用存取是一項快速生效的單一簡單操作，能夠限制每個儲存貯體或帳戶範圍內的公用存取。當您的目標是控制儲存貯體或帳戶中所有物件的公用存取時，這項做法較好。當您需要將自訂 ACL 套用至資訊清單中的每個物件時，請使用 S3 批次操作。如需 S3 封鎖公有存取的詳細資訊，請參閱封鎖對 Amazon S3 儲存體的公開存取權。

S3 物件擁有權

如果資訊清單中物件位於使用儲存貯體擁有者強制執行之「物件擁有權」設定的儲存貯體中，則 Replace access control list (ACL) (取代存取控制清單 (ACL)) 操作只能指定向儲存貯體擁有者授予完全控制權的物件 ACL。操作無法將物件 ACL 許可授予其他 AWS 帳戶 或群組。如需詳細資訊，請參閱「控制物件的擁有權並停用儲存貯體的 ACL」。

法規與限制

- 您指定用於執行取代存取控制清單任務的角色，必須具有執行基礎 Amazon S3 `PutObjectAcl` 操作的許可。如需有關所需許可的詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的 PutObjectAcl。
- S3 批次操作使用 Amazon S3 `PutObjectAcl` 操作，將指定的 ACL 套用至資訊清單中的每個物件。因此，套用至基礎 PutObjectAcl 操作的所有約束和限制，也適用於 S3 批次操作取代存取控制清單任務。

使用批次操作還原物件

還原操作會針對您的清單檔案中列出的已封存 Amazon S3 物件，啟動還原請求。下列封存的物件必須先還原，才能供即時存取：

- 在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別中封存的物件
- 透過 Archive Access 或 Deep Archive Access 方案中的 S3 Intelligent-Tiering 儲存類別封存的物件

在 S3 批次操作任務中使用 S3 起始還原物件操作，會導致資訊清單中指定的每個物件的還原請求。

> **Important**
>
> S3 起始還原物件任務只會提出請求以還原物件。在對物件提出請求之後，S3 批次操作會向每個物件報告任務完成。還原物件之後，Amazon S3 不會更新任務或另外通知您。不過，使用 S3 事件通知即可在 Amazon S3 中出現物件時收到通知。如需詳細資訊，請參閱 Amazon S3 事件通知。

若要建立 S3 啟動還原物件任務，可使用下列引數：

ExpirationInDays

此引數指定 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 物件在 Amazon S3 中保持可用的時長。以 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 物件為目標的啟動還原物件任務，會要求您將 `ExpirationInDays` 設定為 1 或更大數字。

> **Important**
>
> 在建立以 S3 Intelligent-Tiering Archive Access 和 Deep Archive Access 層物件為目標的 S3 啟動還原物件操作任務時，請勿設定 `ExpirationInDays`。S3 Intelligent-Tiering Archive Access 層中的物件不受還原過期的約束，因此指定 `ExpirationInDays` 會造成還原請求失敗。

GlacierJobTier

Amazon S3 可以使用三種不同的擷取層級之一來還原物件：`EXPEDITED`、`STANDARD` 和 `BULK`。不過，S3 Batch 操作功能僅支援`STANDARD`擷取層。如需擷取層級之間差異的詳細資訊，請參閱 封存擷取選項。

如需每個層級定價的詳細資訊，請參閱 Amazon S3 定價頁面上的請求與資料擷取區段。

從 S3 Glacier 和 S3 Intelligent-Tiering 還原的差異

從 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 儲存類別還原封存檔案，與從 Archive Access 或 Deep Archive Access 中的 S3 Intelligent-Tiering 儲存類別中還原檔案不同。

- 當您從 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 還原時，會建立物件的暫時複本。Amazon S3 會在您於 `ExpirationInDays` 引數中指定的值經過之後刪除此複本。刪除此暫時複本之後，您必須提交額外的還原請求才能存取物件。
- 還原封存的 S3 Intelligent-Tiering 物件時，請勿指定 `ExpirationInDays` 引數。當您從 S3 Intelligent-Tiering Archive Access 或 Deep Archive Access 層還原物件時，物件會移回 S3 Intelligent-Tiering Frequent Access 層。在至少連續 90 天無存取的情況下，物件會自動移至 Archive Access 層。在至少連續 180 天無存取的情況下，物件會自動移至 Deep Archive Access 層。
- 批次操作任務可以在 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 儲存類別物件，或者 S3 Intelligent-Tiering Archive Access 和 Deep Archive Access 儲存層物件上進行。批次操作無法在同一任務中同時對兩種類型的封存物件進行操作。若要還原這兩種類型的物件，您必須建立單獨批次操作任務。

重疊還原

如果您的 S3 起始還原物件任務嘗試還原已在還原程序中的物件，S3 批次操作會依下列方式繼續進行。

如果滿足以下任一條件，則物件的還原操作將會成功：

- 與已在進行中的還原請求相比，此任務的 `ExpirationInDays` 值相同，且其 `GlacierJobTier` 值較快。
- 之前的還原請求已完成，且該物件目前可用。在此情況下，批次操作會更新還原物件的到期日期，以符合正在進行的還原請求中指定的 `ExpirationInDays` 值。

如果滿足以下任一條件，則物件的還原操作將會失敗：

- 正在進行的還原請求尚未完成，而此任務的還原持續時間 (由 `ExpirationInDays` 值指定) 與正在進行的還原請求中指定的還原持續時間不同。
- 此任務的還原層 (由 `GlacierJobTier` 值指定) 等於或慢於正在進行的還原請求中指定的還原持續時間。

限制

S3 起始還原物件任務具有下列限制：

- 您必須在與封存物件相同的區域中建立任務。
- S3 批次操作不支援 EXPEDITED 擷取方案。

如需還原物件的詳細資訊，請參閱還原已封存的物件。

S3 物件鎖定保留

物件鎖定保留操作可讓您使用控管模式或合規模式，為物件套用保留日期。這些保留模式可套用不同層級的保護。您可以將任一保留模式套用至任何物件版本。保留日期 (如法務保存) 可防止覆寫或刪除物件。Amazon S3 會將指定的「保留截止日期」儲存在物件的中繼資料中，並保護物件版本的指定版本，直到保留期間到期為止。

您可以搭配物件鎖定來使用 S3 批次作業，一次管理許多 Amazon S3 物件的保留日期。您可以在資訊清單中指定目標物件的清單，並提交至批次操作以便完成。如需詳細資訊，請參閱 S3 物件鎖定the section called “保留期”。

搭配保留日期的 S3 批次作業任務會執行直到完成、取消或失敗狀態為止。當您想要透過單一請求新增、變更或移除多個物件的保留日期時，您應該使用 S3 批次作業和 S3 物件鎖定保留。

批次作業會驗證儲存貯體已啟用物件鎖定，然後才處理資訊清單中的任何金鑰。為了執行作業和驗證，批次作業需要 IAM 角色具有 `s3:GetBucketObjectLockConfiguration` 和 `s3:PutObjectRetention` 許可，以允許批次作業代表您呼叫物件鎖定。如需詳細資訊，請參閱「the section called “物件鎖定的考量事項”」。

如需有關搭配 REST API 使用此作業的資訊，請參閱《Amazon Simple Storage Service API 參考》中 CreateJob 作業的 `S3PutObjectRetention`。

如需使用此作業的 AWS Command Line Interface 範例，請參閱 the section called “使用具有物件鎖定保留的批次操作”。如需 AWS SDK for Java 範例，請參閱the section called “使用具有物件鎖定保留的批次操作”。

法規與限制

- S3 批次作業絕不會進行儲存貯體層級的變更。
- 執行任務的儲存貯體上必須設定版本控制和 S3 物件鎖定。
- 資訊清單中列出的所有物件都必須位於同一個儲存貯體中。

- 除非在資訊清單中明確指定版本，否則作業會適用於物件的最新版本。
- 您的 IAM 角色需要 `s3:PutObjectRetention` 許可才能使用此作業。
- `s3:GetBucketObjectLockConfiguration`需要 IAM 許可，才能確認 S3 儲存貯體已啟用物件鎖定。
- 您只能延長套用 `COMPLIANCE` 模式保留日期的物件的保留期間，而且無法縮短。

S3 物件鎖定法務保存

物件鎖定法務保存操作可讓您對物件版本進行法務保存。就像設定保留期一樣，法務保存可避免物件版本遭到覆寫或刪除。不過，法務保存不具相關聯的保留期，除非將其移除，否則會持續有效。

您可以搭配物件鎖定使用 S3 批次作業，一次為許多 Amazon S3 物件新增法務保存。作法是在資訊清單中列出目標物件，再將該清單提交給批次作業。搭配物件鎖定法務保存的 S3 批次作業任務會一直執行到完成、取消或失敗狀態為止。

S3 批次作業會驗證 S3 儲存貯體已啟用物件鎖定，然後才處理資訊清單中的任何金鑰。若要執行物件操作和儲存貯體層級的驗證，S3 批次操作需要 IAM 角色具有 `s3:PutObjectLegalHold` 和 `s3:GetBucketObjectLockConfiguration`，以允許 S3 批次操作代表您呼叫 S3 物件鎖定。

當您建立 S3 批次作業任務以移除法務保存時，只需將法務保存狀態指定為 Off 即可。如需詳細資訊，請參閱「the section called “物件鎖定的考量事項”」。

如需有關如何搭配 REST API 使用此操作的資訊，請參閱 Amazon Simple Storage Service API 參考中 CreateJob 操作的 `S3PutObjectLegalHold`。

如需此操作的範例，請參閱使用適用於 Java 的 AWS 開發套件。

法規與限制

- S3 批次作業絕不會進行儲存貯體層級的變更。
- 資訊清單中列出的所有物件都必須位於同一個儲存貯體中。
- 執行任務的儲存貯體上必須設定版本控制和 S3 物件鎖定。
- 除非在資訊清單中明確指定版本，否則作業會適用於物件的最新版本。
- `s3:PutObjectLegalHold`您的 IAM 角色需要 許可，才能新增或移除物件的法務保存。
- `s3:GetBucketObjectLockConfiguration`需要 IAM 許可，才能確認 S3 儲存貯體已啟用 S3 物件鎖定。

- 複製物件

- 調用 AWS Lambda 函數
- 取代所有物件標籤
- 刪除所有物件標籤
- 取代存取控制清單
- 使用批次操作還原物件
- S3 物件鎖定保留
- S3 物件鎖定法務保存
- 使用 S3 批次複寫來複寫現有物件

管理 S3 批次作業任務

Amazon S3 提供一組工具，協助您在建立 S3 批次操作任務之後進行管理。本節描述您可以使用 AWS Management Console、AWS CLI、AWS SDK 或 REST API 來管理和追蹤任務的操作。

主題

- 使用 Amazon S3 主控台管理您的 S3 批次操作任務
- 列出任務
- 檢視任務詳細資訊
- 指派任務優先順序

使用 Amazon S3 主控台管理您的 S3 批次操作任務

使用主控台，您可管理您的 S3 批次操作任務。例如，您可以：

- 檢視作用中和排入佇列的任務
- 變更任務的優先順序
- 確認並執行任務
- 複製任務
- 取消任務

使用主控台管理批次操作

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

2. 在左側導覽窗格中，選擇 Batch Operations (批次操作)。
3. 選擇您要管理的特定任務。

列出任務

您可以擷取 S3 批次作業任務的清單。該清單包含尚未完成的任務，以及在過去 90 天內完成的任務。任務清單包含每個任務的資訊，例如 ID、描述、優先順序、目前狀態，以及完成及失敗的任務數。您可以依狀態篩選任務清單。透過主控台擷取任務清單時，您還可以根據描述或 ID 來搜尋任務，並依照 AWS 區域 對其進行篩選。

取得作用中和已完成任務的清單

下列 AWS CLI 範例會取得 `Active` 和 `Complete` 任務的清單。

```
aws s3control list-jobs \
    --region us-west-2 \
    --account-id acct-id \
    --job-statuses '["Active","Complete"]' \
    --max-results 20
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 list-jobs。

檢視任務詳細資訊

若您希望取得比列出任務更詳細的任務詳細資訊，您可以檢視單一任務的所有詳細資訊。您可以查看尚未完成的任務，或過去 90 天內完成的任務的詳細資訊。除了任務清單中傳回的資訊之外，單一任務的詳細資訊還包含其他項目，例如：

- 操作參數
- 資訊清單的詳細資訊
- 有關完成報告的資訊 (如果您在建立任務時有設定)
- 您指派用於執行任務之使用者角色的 Amazon 資源名稱 (ARN)

透過檢視個別任務的詳細資訊，您可以存取任務的整個組態。若要檢視任務的詳細資訊，您可以使用 Amazon S3 主控台或 AWS Command Line Interface (AWS CLI)。

在 Amazon S3 主控台取得 S3 批次操作的任務說明

使用主控台檢視批次操作任務說明

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Batch Operations (批次操作)。
3. 選擇特定任務的 ID 以檢視其詳細資訊。

在 AWS CLI 取得 S3 批次操作的任務說明

下列範例使用 AWS CLI 取得 S3 批次操作任務的說明。若要使用下列範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control describe-job \
--region us-west-2 \
--account-id acct-id \
--job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

如需詳細資訊和範例，請參閱《AWS CLI 命令參考》中的 describe-job。

指派任務優先順序

您可以為每個任務指定數值優先順序，可以是任何正整數。S3 批次作業會根據指定的優先順序，排列任務的優先順序。優先順序較高的任務 (或優先順序參數較高的數值) 會先進行評估。優先順序會依遞減順序決定。例如，優先順序值為 10 的任務佇列的排程優先順序會高於優先順序值為 1 的任務佇列。

您可以在任務執行時變更任務的優先順序。如果您在任務執行時提交較高優先順序的新任務，則系統會暫停較低優先順序的任務，以允許優先順序較高之任務的執行。

變更任務優先順序不會影響任務處理速度。

> **Note**
>
> S3 批次作業會盡力遵循任務優先順序。雖然優先順序較高的任務一般會優先於優先順序較低的任務，但 Amazon S3 不保證任務的嚴格排序。

使用 S3 主控台

如何在AWS Management Console中更新任務優先順序

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Batch Operations (批次操作)。
3. 選擇您要管理的特定任務。
4. 選擇 Action (動作)。在下拉式清單中選擇 Update priority (更新優先順序)。

使用 AWS CLI

下列範例會使用 AWS CLI 更新任務優先順序。數字越高表示執行優先順序越高。

```
aws s3control update-job-priority \
    --region us-west-2 \
    --account-id acct-id \
    --priority 98 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

使用 AWS SDK for Java

下列範例使用 AWS SDK for Java 更新 S3 批次操作任務的優先順序。

如需任務優先順序的詳細資訊，請參閱 指派任務優先順序。

Example

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.UpdateJobPriorityRequest;
```

```
import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateJobPriority {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String jobId = "00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.updateJobPriority(new UpdateJobPriorityRequest()
                    .withAccountId(accountId)
                    .withJobId(jobId)
                    .withPriority(98));

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

追蹤任務狀態和完成報告

使用 S3 批次操作，您可以檢視和更新任務狀態、新增通知和日誌記錄、追蹤任務失敗狀況，以及產生完成報告。

主題

- 範例：透過 AWS CloudTrail 追蹤在 Amazon EventBridge 中的 S3 批次操作任務
- 範例：S3 批次操作完成報告

任務狀態

建立和執行任務後，該任務會根據進度顯示一系列狀態。下表描述狀態及其之間可能的轉換。

狀態	描述	轉換
New	建立任務時，任務的起始狀態為 New。	當 Amazon S3 開始處理資訊清單物件時，任務會自動進入 Preparing 狀態。
Preparing	Amazon S3 正在處理資訊清單物件和其他任務參數，以設定並執行任務。	當 Amazon S3 完成處理資料清單和其他參數後，任務會自動變成 Ready 狀態。然後，任務便可開始對資訊清單中列出的物件執行指定操作。 如果任務在執行前需要確認，例如當您使用 Amazon S3 主控台建立任務時，任務將從 Preparing 轉換為 Suspended 。其將保持在 Suspended 狀態，直到您確認要執行為止。
Suspended	任務需要確認，但您尚未確認是否要執行。只有使用 Amazon S3 主控台建立的任務才需要確認。使用主控台建立的任務會在 Suspended 狀態之後，立即進入 Preparing 狀態。當您確認要執行任務且任務變為 Ready 狀態後，其將永遠不會返回 Suspended 狀態。	確認要執行任務後，其狀態將變更為 Ready。

狀態	描述	轉換
Ready	Amazon S3 已準備開始執行要求的物件操作。	當 Amazon S3 開始執行任務時，任務會自動變成 Active 狀態。任務保持在 Ready 狀態的時間，取決於您是否已執行優先順序更高的任務，以及這些任務需要多長時間才能完成。
Active	Amazon S3 正在對資訊清單中列出的物件執行請求的操作。執行任務時Active，您可以使用 Amazon S3 主控台或透過 REST API 或 AWS 開發套件監控DescribeJob 操作進度。AWS CLI	當任務不再對物件執行操作時，任務將移出 Active 狀態。系統會自動執行此動作，例如當任務成功完成或失敗時。這也可能因為使用者的動作而發生，例如取消任務。任務移至的狀態依轉換原因而定。
Pausing	任務正在從其他狀態轉換為 Paused。	當 Paused 階段結束時，任務將自動移到 Pausing 狀態。
Paused	如果在目前任務執行時提交優先順序更高的其他任務，該任務會變更為 Paused 狀態。	當有任何優先順序較高的任務，使 Paused 任務執行無法完成、失敗，或遭到暫停時，該任務將自動返回 Active 狀態。
Complete	任務已完成對資訊清單中所有物件執行的請求操作。每個物件的操作可能成功或失敗。如果您已設定任務以產生完成報告，則當任務為 Complete 時，系統就會立即提供報告。	Complete 是最終狀態。一旦任務達到 Complete，便不會轉換到任何其他狀態。
Cancelling	任務正在轉換為 Cancelled 狀態。	當 Cancelled 階段結束時，任務將自動移到 Cancelling 狀態。

狀態	描述	轉換
Cancelled	您已請求取消任務，且 S3 批次操作已成功取消任務。該任務不會向 Amazon S3 提交任何新請求。	Cancelled 是最終狀態。任務達到 Cancelled 後，便不會轉換到任何其他狀態。
Failing	任務正在轉換為 Failed 狀態。	當 Failed 階段結束時，任務將自動移到 Failing 狀態。
Failed	任務已失敗，不再繼續執行。如需任務失敗的詳細資訊，請參閱追蹤任務失敗。	Failed 是最終狀態。任務達到 Failed 後，便不會轉換到任何其他狀態。

更新任務狀態

以下 AWS CLI 和 SDK for Java 範例會更新「Batch 作業」工作的狀態。如需使用 S3 主控台管理批次操作任務的詳細資訊，請參閱「使用 Amazon S3 主控台管理您的 S3 批次操作任務」。

使用 AWS CLI

- 如果您在先前的 `--no-confirmation-required` 範例中未指定 `create-job` 參數，任務會保持暫停狀態，直到您將狀態設定為 `Ready` 以確認任務為止。然後 Amazon S3 會使該任務符合執行資格。

```
aws s3control update-job-status \
    --region us-west-2 \
    --account-id 181572960644 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
    --requested-job-status 'Ready'
```

- 將任務狀態設定為 `Cancelled` 來取消任務。

```
aws s3control update-job-status \
     --region us-west-2 \
     --account-id 181572960644 \
     --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
     --status-update-reason "No longer needed" \
```

```
    --requested-job-status Cancelled
```

使用適用於 Java 的 AWS 開發套件

下列範例會使用 AWS SDK for Java更新 S3 批次操作任務的狀態。

如需任務狀態的詳細資訊，請參閱 追蹤任務狀態和完成報告。

Example

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.UpdateJobStatusRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateJobStatus {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String jobId = "00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.updateJobStatus(new UpdateJobStatusRequest()
                    .withAccountId(accountId)
                    .withJobId(jobId)
                    .withRequestedJobStatus("Ready"));

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
```

```
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

通知和記錄日誌

除了請求完成報告之外，您還可以使用 AWS CloudTrail擷取、檢閱和稽核批次操作活動。由於批次作業會使用現有的 Amazon S3 API 來執行任務，因此這些任務也會發出與您直接呼叫時的相同事件。因此，您可以使用您已運用在 Amazon S3 的相同通知、日誌記錄及稽核工具和程序，以追蹤和記錄任務及其所有工作的進度。如需詳細資訊，請參閱下列段落中的範例。

> **Note**
>
> Amazon S3 Batch 操作會在任務執行 CloudTrail 期間產生管理和資料事件。這些事件的磁碟區會隨著每個任務資訊清單中的金鑰數目來調整。如需詳細資訊，請參閱CloudTrail 定價頁面，其中包括定價如何根據您在帳戶中設定的追蹤數量而變更的範例。若要了解如何設定和記錄事件以符合您的需求，請參閱《AWS CloudTrail 使用者指南》中的建立您的第一個追蹤。

如需 Amazon S3 事件的詳細資訊，請參閱Amazon S3 事件通知。

追蹤任務失敗

若 S3 批次作業任務發生問題而無法成功執行 (例如無法讀取指定的資訊清單)，任務便會失敗。任務失敗時會產生一個或多個失敗代碼或失敗原因。S3 批次作業會將失敗代碼和原因連同任務一起儲存，以便您可以請求任務的詳細資訊來檢視。若您請求任務的完成報告，即會在該處顯示故障代碼和原因。

為了防止任務執行大量的失敗操作，Amazon S3 會對每個批次作業任務施加任務失敗閾值。當任務執行至少 1000 個任務時，Amazon S3 會監控工作失敗率。若在任何一個時間點，失敗率 (失敗的任務數，以相對於已執行任務總數的比例呈現) 超過 50%，任務便會失敗。如果您的任務因超出任務失敗閾值而失敗，您可以判定失敗的原因。例如，您可能不小心在資訊清單中包含了一些在指定儲存貯體中不存在的物件。修復錯誤後，您可以重新提交任務。

> **Note**
>
> S3 批次作業以非同步方式運作，工作不一定按照資訊清單列出物件的順序執行。因此，您無法使用資訊清單排序來確定哪些物件的任務成功，以及哪些物件的任務失敗。相反地，您可以檢查工作的完成報告 (如果您要求) 或檢視 AWS CloudTrail 事件記錄檔，以協助判斷失敗的來源。

完成報告

當您建立任務時，您可以請求完成報告。只要 S3 批次作業成功呼叫至少一個工作，Amazon S3 便會在完成執行工作、失敗或取消後產生完成報告。您可以設定完成報告來包含所有任務或僅限失敗的任務。

完成報告包含任務組態，以及每個任務的狀態和資訊，包含物件鍵和版本、狀態、錯誤代碼，以及任何錯誤的描述。完成報告可讓您以合併形式輕鬆檢視任務結果，無需進行額外設定。完成報告會使用 Amazon S3 受管金鑰 (SSE-S3) 加密。如需完成報告的範例，請參閱範例：S3 批次操作完成報告。

如果您未設定完成報告，您仍然可以使用和 Amazon 監控和稽核您的任務 CloudTrail 及其任務 CloudWatch。如需詳細資訊，請參閱下一節。

主題

- 範例：透過 AWS CloudTrail 追蹤在 Amazon EventBridge 中的 S3 批次操作任務
- 範例：S3 批次操作完成報告

範例：透過 AWS CloudTrail 追蹤在 Amazon EventBridge 中的 S3 批次操作任務

Amazon S3 批次操作任務活動會記錄為 AWS CloudTrail 中的事件。您可以在 Amazon EventBridge 中建立自訂規則，並將這些事件傳送到您選擇的目標通知資源，例如 Amazon Simple Notification Service (Amazon SNS)。

> **Note**
>
> Amazon EventBridge 是管理活動的首選方式。Amazon CloudWatch Events 和 EventBridge 是相同的基礎服務和 API，但 EventBridge 提供了更多功能。您在 CloudWatch 或 EventBridge 中所做的變更都會出現在每個主控台中。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》。

追蹤範例

- CloudTrail 中記錄的 S3 批次操作事件
- 追蹤 S3 批次操作任務事件的 EventBridge 規則

CloudTrail 中記錄的 S3 批次操作事件

建立批次操作任務時，該任務會記錄為 CloudTrail 中的 `JobCreated` 事件。當任務執行時，它會在處理期間變更狀態，並在 CloudTrail 中記錄其他 `JobStatusChanged` 事件。您可以在 CloudTrail 主控台上檢視這些事件。如需 CloudTrail 的詳細資訊，請參閱《AWS CloudTrail 使用者指南》。

> **Note**
>
> CloudTrail 中只會記錄 S3 批次操作任務 `status-change` 事件。

Example CloudTrail 所記錄的 S3 批次操作任務完成事件

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "accountId": "123456789012",
        "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2020-02-05T18:25:30Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "JobStatusChanged",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "s3.amazonaws.com",
    "userAgent": "s3.amazonaws.com",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "f907577b-bf3d-4c53-b9ed-8a83a118a554",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "123412341234",
    "serviceEventDetails": {
        "jobId": "d6e58ec4-897a-4b6d-975f-10d7f0fb63ce",
        "jobArn": "arn:aws:s3:us-west-2:181572960644:job/
d6e58ec4-897a-4b6d-975f-10d7f0fb63ce",
        "status": "Complete",
```

```
        "jobEventId": "b268784cf0a66749f1a05bce259804f5",
        "failureCodes": [],
        "statusChangeReason": []
    }
}
```

追蹤 S3 批次操作任務事件的 EventBridge 規則

下列範例顯示如何在 Amazon EventBridge 中建立規則，將 AWS CloudTrail 記錄的 S3 批次操作事件擷取到您選擇的目標。

若要執行此動作，請遵循建立對事件進行反應的 EventBridge 規則中的所有步驟，以建立規則。您可以在情況適用時貼上下列 S3 批次操作自訂事件模式政策，然後選擇您所選的目標服務。

S3 批次操作自訂事件模式政策

```
{
    "source": [
        "aws.s3"
    ],
    "detail-type": [
        "AWS Service Event via CloudTrail"
    ],
    "detail": {
        "eventSource": [
            "s3.amazonaws.com"
        ],
        "eventName": [
            "JobCreated",
            "JobStatusChanged"
        ]
    }
}
```

下列範例是從 EventBridge 事件規則傳送至 Amazon Simple Queue Service (Amazon SQS) 的兩個批次操作事件。批次操作任務處理時，會經歷許多不同的狀態 (`New`、`Preparing`、`Active` 等)，因此您可以預期每個任務都會收到幾則訊息。

Example JobCreated 範例事件

```
{
```

```
    "version": "0",
    "id": "51dc8145-541c-5518-2349-56d7dffdf2d8",
    "detail-type": "AWS Service Event via CloudTrail",
    "source": "aws.s3",
    "account": "123456789012",
    "time": "2020-02-27T15:25:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventVersion": "1.05",
        "userIdentity": {
            "accountId": "11112223334444",
            "invokedBy": "s3.amazonaws.com"
        },
        "eventTime": "2020-02-27T15:25:49Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "JobCreated",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "s3.amazonaws.com",
        "userAgent": "s3.amazonaws.com",
        "eventID": "7c38220f-f80b-4239-8b78-2ed867b7d3fa",
        "readOnly": false,
        "eventType": "AwsServiceEvent",
        "serviceEventDetails": {
            "jobId": "e849b567-5232-44be-9a0c-40988f14e80c",
            "jobArn": "arn:aws:s3:us-east-1:181572960644:job/
e849b567-5232-44be-9a0c-40988f14e80c",
            "status": "New",
            "jobEventId": "f177ff24f1f097b69768e327038f30ac",
            "failureCodes": [],
            "statusChangeReason": []
        }
    }
}
```

Example JobStatusChanged 任務完成事件

```
{
  "version": "0",
  "id": "c8791abf-2af8-c754-0435-fd869ce25233",
  "detail-type": "AWS Service Event via CloudTrail",
  "source": "aws.s3",
  "account": "123456789012",
```

```
  "time": "2020-02-27T15:26:42Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "eventVersion": "1.05",
    "userIdentity": {
      "accountId": "1111222233334444",
      "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2020-02-27T15:26:42Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "JobStatusChanged",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "s3.amazonaws.com",
    "userAgent": "s3.amazonaws.com",
    "eventID": "0238c1f7-c2b0-440b-8dbd-1ed5e5833afb",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "serviceEventDetails": {
      "jobId": "e849b567-5232-44be-9a0c-40988f14e80c",
      "jobArn": "arn:aws:s3:us-east-1:181572960644:job/
e849b567-5232-44be-9a0c-40988f14e80c",
      "status": "Complete",
      "jobEventId": "51f5ac17dba408301d56cd1b2c8d1e9e",
      "failureCodes": [],
      "statusChangeReason": []
    }
  }
}
```


範例：S3 批次操作完成報告

建立 S3 批次操作任務時，您可以請求所有任務或僅限失敗任務的完成報告。只要順利叫用至少一個任務，S3 批次操作就會產生已完成、失敗或已取消任務的報告。

完成報告包含每個任務的其他資訊，包含物件鍵名稱和版本、狀態、錯誤代碼，以及任何錯誤的描述。每個失敗任務的錯誤描述可用來診斷任務建立期間發生的問題，例如許可。

> **Note**
>
> 完成報告一律使用 Amazon S3 受管金鑰 (SSE-S3) 加密。

Example 最上層資訊清單結果檔案

最上層 manifest.json 檔案包含每個成功報告的位置，以及 (如果任務包含任何失敗) 失敗報告的位置，如下範例所示。

```
{
    "Format": "Report_CSV_20180820",
    "ReportCreationDate": "2019-04-05T17:48:39.725Z",
    "Results": [
        {
            "TaskExecutionStatus": "succeeded",
            "Bucket": "my-job-reports",
            "MD5Checksum": "83b1c4cbe93fc893f54053697e10fd6e",
            "Key": "job-f8fb9d89-a3aa-461d-bddc-ea6a1b131955/
results/6217b0fab0de85c408b4be96aeaca9b195a7daa5.csv"
        },
        {
            "TaskExecutionStatus": "failed",
            "Bucket": "my-job-reports",
            "MD5Checksum": "22ee037f3515975f7719699e5c416eaa",
            "Key": "job-f8fb9d89-a3aa-461d-bddc-ea6a1b131955/results/
b2ddad417e94331e9f37b44f1faf8c7ed5873f2e.csv"
        }
    ],
    "ReportSchema": "Bucket, Key, VersionId, TaskStatus, ErrorCode, HTTPStatusCode,
 ResultMessage"
}
```

Example 失敗的任務報告

失敗的任務報告包含所有失敗任務的下列資訊：

- Bucket
- Key
- VersionId
- TaskStatus
- ErrorCode
- HTTPStatusCode
- ResultMessage

下列範例報告顯示 AWS Lambda 函數逾時，導致失敗超過失敗臨界值的情況。接著會將此報告標示為 PermanentFailure。

```
awsexamplebucket1,image_14975,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:21.155Z 2845ca0d-38d9-4c4b-
abcf-379dc749c452 Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_15897,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:29.610Z 2d0a330b-de9b-425f-
b511-29232fde5fe4 Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_14819,,failed,200,PermanentFailure,"Lambda returned function
 error: {""errorMessage"":""2019-04-05T17:35:22.362Z fcf5efde-74d4-4e6d-b37a-
c7f18827f551 Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_15930,,failed,200,PermanentFailure,"Lambda returned function
 error: {""errorMessage"":""2019-04-05T17:35:29.809Z 3dd5b57c-4a4a-48aa-8a35-
cbf027b7957e Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_17644,,failed,200,PermanentFailure,"Lambda
 returned function error: {""errorMessage"":""2019-04-05T17:35:46.025Z
 10a764e4-2b26-4d8c-9056-1e1072b4723f Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_17398,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:44.661Z 1e306352-4c54-4eba-
aee8-4d02f8c0235c Task timed out after 3.00 seconds""}"
```

Example 成功的任務報告

成功工作報告包含成功工作的下列項目：

- Bucket
- Key
- VersionId
- TaskStatus
- ErrorCode
- HTTPStatusCode
- ResultMessage

在下列範例中，Lambda 函數已順利將 Amazon S3 物件複製到另一個儲存貯體。傳回的 Amazon S3 回應會傳回至 S3 批次操作，然後寫入最終完成報告。

```
awsexamplebucket1,image_17775,,succeeded,200,,"{u'CopySourceVersionId':
 'xVR78haVKlRnurYofbTfYr3ufYbktF8h', u'CopyObjectResult': {u'LastModified':
```

```
 datetime.datetime(2019, 4, 5, 17, 35, 39, tzinfo=tzlocal()), u'ETag':
 '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata': {'HTTPStatusCode':
 200, 'RetryAttempts': 0, 'HostId': 'nXNaClIMxEJzWNmeMNQV2KpjbaCJLn0OGoXWZpuVOFS/
iQYWxb3QtTvzX9SVfx2lA3oTKLwImKw=', 'RequestId': '3ED5852152014362', 'HTTPHeaders':
 {'content-length': '234', 'x-amz-id-2': 'nXNaClIMxEJzWNmeMNQV2KpjbaCJLn0OGoXWZpuVOFS/
iQYWxb3QtTvzX9SVfx2lA3oTKLwImKw=', 'x-amz-copy-source-version-id':
 'xVR78haVKlRnurYofbTfYr3ufYbktF8h', 'server': 'AmazonS3', 'x-amz-request-id':
 '3ED5852152014362', 'date': 'Fri, 05 Apr 2019 17:35:39 GMT', 'content-type':
 'application/xml'}}}"
awsexamplebucket1,image_17763,,succeeded,200,,"{u'CopySourceVersionId':
 '6HjOUSim4Wj6BTcbxToXW44pSZ.40pwq', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 39, tzinfo=tzlocal()),
 u'ETag': '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata':
 {'HTTPStatusCode': 200, 'RetryAttempts': 0, 'HostId': 'GiCZNYr8LHd/
Thyk6beTRP96IGZk2sYxujLe13TuuLpq6U2RD3we0YoluuIdm1PRvkMwnEW1aFc=', 'RequestId':
 '1BC9F5B1B95D7000', 'HTTPHeaders': {'content-length': '234', 'x-amz-id-2':
 'GiCZNYr8LHd/Thyk6beTRP96IGZk2sYxujLe13TuuLpq6U2RD3we0YoluuIdm1PRvkMwnEW1aFc=', 'x-
amz-copy-source-version-id': '6HjOUSim4Wj6BTcbxToXW44pSZ.40pwq', 'server': 'AmazonS3',
 'x-amz-request-id': '1BC9F5B1B95D7000', 'date': 'Fri, 05 Apr 2019 17:35:39 GMT',
 'content-type': 'application/xml'}}}"
awsexamplebucket1,image_17860,,succeeded,200,,"{u'CopySourceVersionId':
 'm.MDD0g_QsUnYZ8TBzVFrp.TmjN8PJyX', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 40, tzinfo=tzlocal()), u'ETag':
 '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata': {'HTTPStatusCode':
 200, 'RetryAttempts': 0, 'HostId': 'F9ooZOgpE5g9sNgBZxjdiPHqB4+0DNWgj3qbsir
+sKai4fv7rQEcF2fBN1VeeFc2WH45a9ygb2g=', 'RequestId': '8D9CA56A56813DF3', 'HTTPHeaders':
 {'content-length': '234', 'x-amz-id-2': 'F9ooZOgpE5g9sNgBZxjdiPHqB4+0DNWgj3qbsir
+sKai4fv7rQEcF2fBN1VeeFc2WH45a9ygb2g=', 'x-amz-copy-source-version-id':
 'm.MDD0g_QsUnYZ8TBzVFrp.TmjN8PJyX', 'server': 'AmazonS3', 'x-amz-request-id':
 '8D9CA56A56813DF3', 'date': 'Fri, 05 Apr 2019 17:35:40 GMT', 'content-type':
 'application/xml'}}}"
```

使用標籤控制存取和標記任務

您可以透過新增標籤，標示和控制對 S3 批次操作任務的存取權。標籤可用來識別負責批次操作任務的人員。任務標籤的存在可以授與或限制使用者的以下能力：取消任務、啟動處於確認狀態的任務或變更任務的優先順序層級。您可以建立已連接標籤的任務，也可以在建立任務後將標籤新增至任務。每個標籤都是一個鍵值對，可以在建立任務時包含或稍後更新。

> **Warning**
>
> 任務標籤不應包含任何機密資訊或個人資料。

請考慮下列標記範例：假設您希望財務部門建立批次作業任務。您可以撰寫 AWS Identity and Access Management (IAM) 政策，允許使用者叫用 `CreateJob`，只要該任務是使用指定數值 `Finance` 的 `Department` 標籤建立的即可。此外，您可以將該政策連接到屬於財務部門成員的所有使用者。

您可以繼續使用此範例撰寫政策，允許使用者更新具有所需標籤之任何任務的優先順序，或取消任何具有該些標籤的任務。如需詳細資訊，請參閱the section called “控制許可”。

您可以在建立新的 S3 批次作業任務時新增標籤，或將新增至現有任務。

請注意以下標籤限制：

- 您最多可以將 50 個標籤與一項任務建立關聯，只要它們具有唯一的標籤鍵。
- 標籤金鑰最長可包含 128 個 Unicode 字元，標籤值最長可包含 256 個 Unicode 字元。
- 金鑰與值皆會區分大小寫。

如需標籤限制的詳細資訊，請參閱《AWS Billing and Cost Management 使用者指南》中的使用者定義的標籤限制。

與 S3 批次作業任務標記相關的 API 作業

Amazon S3 支援以下專屬於 S3 批次作業任務標記的 API 操作：

- GetJobTagging — 傳回與批次作業任務相關聯的標籤組。
- PutJobTagging — 取代與任務相關聯的標籤組。使用此 API 動作進行 S3 批次作業任務標籤管理時，有兩種不同的情況：
  - 任務沒有標籤 — 您可以為任務新增一組標籤 (任務先前沒有標籤)。
  - 任務有一組現有的標籤 — 若要修改現有的標籤組，您可以完全取代現有的標籤組，或使用 GetJobTagging 擷取現有的標籤組，修改該標籤組，然後使用此 API 動作將標籤組換成您修改過的標籤組。

> **Note**
>
> 若傳送此請求時附上空的標籤組，則 S3 批次作業會刪除物件上的現有標籤組。如果您使用此方法，系統會向您收取 Tier 1 Request (PUT) 的費用。如需詳細資訊，請參閱 Amazon S3 定價。
>
> 若要刪除批次作業任務的現有標籤，建議採取 `DeleteJobTagging` 動作，因為這可以達到相同的結果，而不會產生費用。

- DeleteJobTagging — 刪除與批次作業任務相關聯的標籤組。

建立具有任務標籤 (用於標示) 的批次作業任務

您可以透過新增標籤，標示和控制對 S3 批次操作任務的存取權。標籤可用來識別負責批次操作任務的人員。您可以建立已連接標籤的任務，也可以在建立任務後將標籤新增至任務。如需詳細資訊，請參閱「the section called “使用標籤”」。

使用 AWS CLI

下列 AWS CLI 範例會建立 S3 批次操作任務 S3PutObjectCopy，並使用任務標籤作為該任務的標籤。

1. 選取您要批次操作任務執行的動作或 OPERATION，然後選擇您的 TargetResource。

```
read -d '' OPERATION <<EOF
{
  "S3PutObjectCopy": {
    "TargetResource": "arn:aws:s3:::destination-bucket"
  }
}
EOF
```

2. 識別您要用於任務的 TAGS。在本例中，您套用兩個標籤，department 和FiscalYear，分別具有數值 Marketing 和 2020。

```
read -d '' TAGS <<EOF
[
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
  }
]
EOF
```

3. 指定批次操作任務的 MANIFEST。

```
read -d '' MANIFEST <<EOF
```

```
{
  "Spec": {
    "Format": "EXAMPLE_S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::example-bucket/example_manifest.csv",
    "ETag": "example-5dc7a8bfb90808fc5d546218"
  }
}
EOF
```

4. 設定批次操作任務的 REPORT。

```
read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::example-report-bucket",
  "Format": "Example_Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/copy-with-replace-metadata",
  "ReportScope": "AllTasks"
}
EOF
```

5. 執行 `create-job` 動作，使用上述步驟中設定的輸入來建立您的批次操作任務。

```
aws \
    s3control create-job \
    --account-id 123456789012 \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn arn:aws:iam::123456789012:role/batch-operations-role \
    --tags "${TAGS//$'\n'/}" \
    --client-request-token "$(uuidgen)" \
    --region us-west-2 \
    --description "Copy with Replace Metadata";
```

使用適用於 Java 的 AWS 開發套件

Example

下列範例會使用 AWS SDK for Java 建立具有標籤的 S3 批次操作任務。

```
public String createJob(final AWSS3ControlClient awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::example-manifest-bucket/
manifests/10_manifest.csv";
    final String manifestObjectVersionId = "example-5dc7a8bfb90808fc5d546218";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new
 JobManifestSpec().withFormat(JobManifestFormat.S3InventoryReport_CSV_20161130);

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::example-report-bucket";
    final String jobReportPrefix = "example-job-reports";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final String lambdaFunctionArn = "arn:aws:lambda:us-
west-2:123456789012:function:example-function";

    final JobOperation jobOperation = new JobOperation()
            .withLambdaInvoke(new
 LambdaInvokeOperation().withFunctionArn(lambdaFunctionArn));

    final S3Tag departmentTag = new
 S3Tag().withKey("department").withValue("Marketing");
    final S3Tag fiscalYearTag = new S3Tag().withKey("FiscalYear").withValue("2020");
```

```
    final String roleArn = "arn:aws:iam::123456789012:role/example-batch-operations-
role";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Test lambda job")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withTags(departmentTag, fiscalYearTag)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

從 S3 批次作業任務中刪除標籤

您可以使用這些範例，從批次作業任務刪除標籤。

使用 AWS CLI

下列範例會使用 AWS CLI 從批次操作任務中刪除標籤。

```
aws \
    s3control delete-job-tagging \
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

刪除批次操作任務的任務標籤

Example

下列範例會使用 AWS SDK for Java 刪除 S3 批次操作任務的標籤。

```
public void deleteJobTagging(final AWSS3ControlClient awss3ControlClient,
                             final String jobId) {
```

```
    final DeleteJobTaggingRequest deleteJobTaggingRequest = new
 DeleteJobTaggingRequest()
            .withJobId(jobId);

    final DeleteJobTaggingResult deleteJobTaggingResult =
                awss3ControlClient.deleteJobTagging(deleteJobTaggingRequest);
}
```

為現有的 S3 批次作業任務放置任務標籤

您可以使用 PutJobTagging 將任務標籤新增至現有的 S3 批次操作任務。如需詳細資訊，請參閱下列範例。

使用 AWS CLI

以下是使用 AWS CLI，以使用 `s3control put-job-tagging` 將任務標籤新增至 S3 批次操作任務的範例。

> **Note**
>
> 若傳送此請求時附上空的標籤組，則 S3 批次作業會刪除物件上的現有標籤組。此外，如果您使用此方法，系統會向您收取 Tier 1 Request (PUT) 的費用。如需詳細資訊，請參閱 Amazon S3 定價。
>
> 若要刪除批次作業任務的現有標籤，建議採取 `DeleteJobTagging` 動作，因為這可以達到相同的結果，而不會產生費用。

1. 識別您要用於任務的 TAGS。在本例中，您套用兩個標籤，`department` 和`FiscalYear`，分別具有數值 `Marketing` 和 `2020`。

```
read -d '' TAGS <<EOF
[
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
  }
]
```

```
EOF
```

2. 使用必要的參數執行 put-job-tagging 動作。

```
aws \
    s3control put-job-tagging \
    --account-id 123456789012 \
    --tags "${TAGS//$'\n'/}" \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

使用適用於 Java 的 AWS 開發套件

Example

下列範例會使用 AWS SDK for Java 放置 S3 批次操作任務的標籤。

```
public void putJobTagging(final AWSS3ControlClient awss3ControlClient,
                          final String jobId) {
    final S3Tag departmentTag = new
 S3Tag().withKey("department").withValue("Marketing");
    final S3Tag fiscalYearTag = new S3Tag().withKey("FiscalYear").withValue("2020");

    final PutJobTaggingRequest putJobTaggingRequest = new PutJobTaggingRequest()
            .withJobId(jobId)
            .withTags(departmentTag, fiscalYearTag);

    final PutJobTaggingResult putJobTaggingResult =
 awss3ControlClient.putJobTagging(putJobTaggingRequest);
}
```

取得 S3 批次作業任務的標籤

您可以使用 GetJobTagging 來傳回 S3 批次操作任務的標籤。如需詳細資訊，請參閱下列範例。

使用 AWS CLI

下列範例會使用 AWS CLI 取得批次操作任務的標籤。

```
aws \
    s3control get-job-tagging \
```

```
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

使用適用於 Java 的 AWS 開發套件

Example

下列範例會使用 AWS SDK for Java 取得 S3 批次操作任務的標籤。

```
public List<S3Tag> getJobTagging(final AWSS3ControlClient awss3ControlClient,
                                 final String jobId) {
    final GetJobTaggingRequest getJobTaggingRequest = new GetJobTaggingRequest()
            .withJobId(jobId);

    final GetJobTaggingResult getJobTaggingResult =
            awss3ControlClient.getJobTagging(getJobTaggingRequest);

    final List<S3Tag> tags = getJobTaggingResult.getTags();

    return tags;
}
```

使用任務標籤控制 S3 批次作業的許可

為了協助您管理 S3 批次操作任務，您可以新增任務標籤。您可以使用任務標籤控制對批次操作任務的存取，並強制在建立任何任務時套用標籤。

您最多可以將 50 個任務標籤套用至每個批次操作任務。這可讓您設定非常細微的政策，限制可編輯任務的使用者集。任務標籤可以授與或限制使用者的以下能力：取消任務、啟動處於確認狀態的任務或變更任務的優先順序層級。此外，您可以強制將標籤套用至所有新任務，並為標籤指定允許的鍵值對。您可以使用相同的 IAM 政策語言來表達所有這些條件。如需詳細資訊，請參閱服務授權參考中適用於 Amazon S3 的動作、資源和條件金鑰。

下列範例顯示如何使用 S3 批次操作任務標籤授予使用者僅能建立和編輯在特定部門 (例如財務或合規部門) 內執行的任務的許可。您也可以根據與其相關的開發階段指派任務，例如 QA 或生產。

在此範例中，您可以在 AWS Identity and Access Management (IAM) 政策中使用 S3 Batch 操作任務標籤，授與使用者僅建立和編輯部門內執行的任務的權限。您可以根據任務相關的開發階段指派任務，例如 QA 或生產。

此範例使用下列部門，每個部門各自以不同的方式使用批次操作：

- 財務
- 合規
- 商業智慧
- 工程設計

主題

- 透過指派標籤到使用者和資源來控制存取
- 依階段標記批次操作任務，並強制執行工作優先順序限制

透過指派標籤到使用者和資源來控制存取

在本例中，系統管理員使用屬性型存取控制 (ABAC)。ABAC 是 IAM 授權策略，透過將標籤附加到使用者和 AWS 資源來定義許可。

系統會為使用者和任務指派下列其中一個部門標籤：

索引鍵：數值

- `department : Finance`
- `department : Compliance`
- `department : BusinessIntelligence`
- `department : Engineering`

> **Note**
>
> 任務標籤鍵與數值皆區分大小寫。

使用 ABAC 存取控制策略，您可以藉由將標籤 `department=Finance` 與其使用者關聯，以授予財務部門的使用者建立和管理其部門內 S3 批次操作任務的許可。

此外，您可以將受管政策連接至 IAM 使用者，以允許該公司內任何使用者建立或修改各自部門內的 S3 批次操作任務。

此範例中的政策包含三個政策陳述式：

- 政策中的第一個陳述式可讓使用者建立批次操作任務，前提是建立任務請求包含符合其各自部門的任務標籤。這是使用 "${aws:PrincipalTag/department}" 語法表示，該語法在政策評估時會以使用者的部門標籤取代。當為 ("aws:RequestTag/department") 請求中的部門標籤提供的數值符合使用者的部門時，即滿足條件。
- 政策中的第二個陳述式可讓使用者變更任務的優先順序或更新任務狀態，只要使用者正在更新的任務符合使用者的部門即可。
- 第三個陳述式可讓使用者隨時透過 PutJobTagging 請求更新批次操作任務的標籤，只要 (1) 有保留其部門標籤，(2) 他們正在更新的任務在其部門內。

```
{
      "Version": "2012-10-17",
      "Statement": [
            {
                  "Effect": "Allow",
                  "Action": "s3:CreateJob",
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:RequestTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        },
            {
                  "Effect": "Allow",
                  "Action": [
                        "s3:UpdateJobPriority",
                        "s3:UpdateJobStatus"
            ],
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        },
            {
                  "Effect": "Allow",
                  "Action": "s3:PutJobTagging",
                  "Resource": "*",
```

```
                "Condition": {
                      "StringEquals": {
                            "aws:RequestTag/department": "${aws:PrincipalTag/
department}",
                            "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
             }
        }
    ]
}
```

依階段標記批次操作任務，並強制執行工作優先順序限制

所有 S3 批次操作任務都具有數字優先順序，Amazon S3 會用來決定執行任務的順序。在此範例中，您可以限制大多數使用者可以指派給任務的最大優先順序，較高的優先順序範圍會保留給有限的特殊權限使用者，如下所示：

- QA 階段優先順序範圍 (低)：1-100
- 生產階段優先順序範圍 (高)：1-300

若要進行此動作，請引入一組新標籤來代表任務之階段：

索引鍵：數值

- `stage : QA`
- `stage : Production`

建立與更新部門內的低優先順序任務

除了以部門為基礎的限制外，此政策還會引入兩項新的 S3 批次操作任務建立與更新限制：

- 它允許使用者建立或更新部門中的任務，並有一項新條件要求任務必須包含標籤 `stage=QA`。
- 它允許使用者建立或更新任務的優先順序，新的優先順序最高為 100。

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

```
        {
        "Effect": "Allow",
        "Action": "s3:CreateJob",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/department": "${aws:PrincipalTag/department}",
                "aws:RequestTag/stage": "QA"
            },
            "NumericLessThanEquals": {
                "s3:RequestJobPriority": 100
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:UpdateJobStatus"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:UpdateJobPriority",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}",
                "aws:ResourceTag/stage": "QA"
            },
            "NumericLessThanEquals": {
                "s3:RequestJobPriority": 100
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:PutJobTagging",
        "Resource": "*",
```

```
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/department" : "${aws:PrincipalTag/department}",
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}",
                "aws:RequestTag/stage": "QA",
                "aws:ResourceTag/stage": "QA"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:GetJobTagging",
        "Resource": "*"
    }
    ]
}
```

建立與更新部門內的高優先順序任務

少數使用者可能需要在 QA 或生產中建立高優先順序的任務。若要支援這項需求，您可以建立受管政策，根據上一節中的低優先順序政策調整該政策。

此政策會執行下列動作：

- 允許使用者使用標籤 `stage=QA` 或 `stage=Production` 來建立或更新其部門中的任務。
- 允許使用者建立或更新任務的優先順序，最多可達 300。

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
                "Effect": "Allow",
                "Action": "s3:CreateJob",
                "Resource": "*",
                "Condition": {
                      "ForAnyValue:StringEquals": {
                            "aws:RequestTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                },
```

```
                    "StringEquals": {
                        "aws:RequestTag/department": "${aws:PrincipalTag/
department}"
                },
                    "NumericLessThanEquals": {
                        "s3:RequestJobPriority": 300
                }
            }
        },
          {
                "Effect": "Allow",
                "Action": [
                    "s3:UpdateJobStatus"
            ],
                "Resource": "*",
                "Condition": {
                    "StringEquals": {
                        "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        },
          {
                "Effect": "Allow",
                "Action": "s3:UpdateJobPriority",
                "Resource": "*",
                "Condition": {
                    "ForAnyValue:StringEquals": {
                        "aws:ResourceTag/stage": [
                            "QA",
                            "Production"
                    ]
                },
                    "StringEquals": {
                        "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                },
                    "NumericLessThanEquals": {
                        "s3:RequestJobPriority": 300
                }
            }
        },
          {
                "Effect": "Allow",
```

```
                "Action": "s3:PutJobTagging",
                "Resource": "*",
                "Condition": {
                      "StringEquals": {
                            "aws:RequestTag/department": "${aws:PrincipalTag/
department}",
                            "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                },
                      "ForAnyValue:StringEquals": {
                            "aws:RequestTag/stage": [
                                  "QA",
                                  "Production"
                    ],
                            "aws:ResourceTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                }
            }
        }
    ]
}
```

使用 S3 批次作業管理 S3 物件鎖定

使用 S3 物件鎖定，您可以對物件版本進行法務保存。就像設定保留期一樣，法務保存可避免物件版本遭到覆寫或刪除。不過，法務保存不具相關聯的保留期，除非將其移除，否則會持續有效。如需詳細資訊，請參閱「S3 物件鎖定法務保存」。

有關使用 S3 批次操作搭配物件鎖定，一次為多個 Amazon S3 物件新增法務保存的資訊，請參閱以下各節。

主題

- 使用 S3 批次作業啟用 S3 物件鎖定
- 使用批次作業設定物件鎖定保留
- 使用具有 S3 物件鎖定保留合規模式的 S3 批次操作
- 使用具有 S3 物件鎖定保留控管模式的 S3 批次操作
- 使用 S3 批次操作關閉 S3 物件鎖定法務保存

使用 S3 批次作業啟用 S3 物件鎖定

您可以使用具有 S3 物件鎖定的 S3 批次操作來管理保留，或同時啟用多個 Amazon S3 物件的合法保留。您可以在資訊清單中指定目標物件的清單，並提交至批次操作以便完成。如需詳細資訊，請參閱「the section called “物件鎖定保留”」和「the section called “物件鎖定合法保留”」。

下列範例顯示如何建立具有 S3 批次作業許可的 IAM 角色，以及更新角色許可，以建立會啟用物件鎖定的任務。在範例中，將任何變數值取代為符合您需求的變數值。您也必須擁有能識別用於 S3 批次操作任務之物件的 CSV 資訊清單。如需詳細資訊，請參閱 the section called “指定資訊清單”。

使用 AWS CLI

1. 建立 IAM 角色並指派要執行的 S3 批次操作許可。

   所有 S3 批次操作任務都需要此步驟。

```
export AWS_PROFILE='aws-user'

read -d '' bops_trust_policy <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "batchoperations.s3.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
aws iam create-role --role-name bops-objectlock --assume-role-policy-document
 "${bops_trust_policy}"
```

2. 設定具有 S3 物件鎖定的 S3 批次操作以便執行。

   在此步驟中，您允許角色執行下列動作：

   a. 對包含要執行批次操作的目標物件的 S3 儲存貯體執行物件鎖定。

b. 讀取資訊清單 CSV 檔案和物件所在的 S3 儲存貯體。

c. 將 S3 批次操作任務的結果寫入報告儲存貯體。

```
read -d '' bops_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::{{ManifestBucket}}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{ManifestBucket}}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{ReportBucket}}/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name object-lock-
permissions --policy-document "${bops_permissions}"
```

使用適用於 Java 的 AWS 開發套件

下列範例顯示如何建立具有 S3 批次操作許可的 IAM 角色，以及更新角色許可，以建立會啟用物件鎖定的任務 (使用 AWS SDK for Java)。在程式碼中，將任何變數值取代為符合您需求的變數值。您也必須擁有能識別用於 S3 批次操作任務之物件的 CSV 資訊清單。如需詳細資訊，請參閱「the section called “指定資訊清單”」。

您會執行以下步驟：

1. 建立 IAM 角色並指派要執行的 S3 批次操作許可。所有 S3 批次操作任務都需要此步驟。
2. 設定具有 S3 物件鎖定的 S3 批次操作以便執行。

   您允許角色執行下列動作：

   1. 對包含要執行批次操作的目標物件的 S3 儲存貯體執行物件鎖定。
   2. 讀取資訊清單 CSV 檔案和物件所在的 S3 儲存貯體。
   3. 將 S3 批次操作任務的結果寫入報告儲存貯體。

```
public void createObjectLockRole() {
    final String roleName = "bops-object-lock";

    final String trustPolicy = "{" +
            "  \"Version\": \"2012-10-17\", " +
            "  \"Statement\": [ " +
            "    { " +
            "      \"Effect\": \"Allow\", " +
            "      \"Principal\": { " +
            "        \"Service\": [" +
            "          \"batchoperations.s3.amazonaws.com\"" +
            "        ]" +
            "      }, " +
            "      \"Action\": \"sts:AssumeRole\" " +
            "    } " +
            "  ]" +
            "}";

    final String bopsPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
```

```
            "            \"Action\": \"s3:GetBucketObjectLockConfiguration\"," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket\"" +
            "            ]" +
            "        }," +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:GetObject\"," +
            "                \"s3:GetObjectVersion\"," +
            "                \"s3:GetBucketLocation\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket/*\"" +
            "            ]" +
            "        }," +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObject\"," +
            "                \"s3:GetBucketLocation\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ReportBucket/*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final CreateRoleRequest createRoleRequest = new CreateRoleRequest()
            .withAssumeRolePolicyDocument(bopsPermissions)
            .withRoleName(roleName);

    final CreateRoleResult createRoleResult = iam.createRole(createRoleRequest);

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(bopsPermissions)
            .withPolicyName("bops-permissions")
            .withRoleName(roleName);
```

```
    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

使用批次作業設定物件鎖定保留

下列範例允許規則為資訊清單儲存貯體中的物件設定 S3 物件鎖定保留。

您可以更新角色，將 `s3:PutObjectRetention` 許可納入，以便對儲存貯體中的物件執行物件鎖定保留。

使用 AWS CLI

```
export AWS_PROFILE='aws-user'

read -d '' retention_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectRetention"
            ],
            "Resource": [
                "arn:aws:s3:::{{ManifestBucket}}/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name retention-permissions
 --policy-document "${retention_permissions}"
```

使用適用於 Java 的 AWS 開發套件

```
public void allowPutObjectRetention() {
    final String roleName = "bops-object-lock";

    final String retentionPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
```

```
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObjectRetention\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(retentionPermissions)
            .withPolicyName("retention-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

使用具有 S3 物件鎖定保留合規模式的 S3 批次操作

下列範例以先前建立信任政策的範例，以及對物件設定 S3 批次作業和 S3 物件鎖定組態許可的範例為基礎。此範例會將保留模式設定為COMPLIANCE和設定retain until date為 2025 年 1 月 1 日。它會建立以資訊清單儲存貯體中物件為目標的任務，並在您識別的報告儲存貯體中報告結果。

使用 AWS CLI

Example 設定多個物件的保留合規

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
```

```
      "RetainUntilDate":"2025-01-01T00:00:00",
      "Mode":"COMPLIANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/compliance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/compliance-objects-bops",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
```

```
    --description "Set compliance retain-until to 1 Jul 2030";
```

Example 將此**COMPLIANCE**模式延長**retain until date**至 2025 年 1 月 15 日

下列範例將 COMPLIANCE 模式的 `retain until date` 延長到 2025 年 1 月 15 日。

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-15T00:00:00",
      "Mode":"COMPLIANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/compliance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/compliance-objects-bops",
```

```
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Extend compliance retention to 15 Jan 2025";
```

使用適用於 Java 的 AWS 開發套件

Example 將保留模式設定為 [規範遵循]，並將保留至 2025 年 1 月 1 日為止。

```
public String createComplianceRetentionJob(final AWSS3ControlClient awss3ControlClient)
 throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/compliance-objects-
manifest.csv";
    final String manifestObjectVersionId = "your-object-version-Id";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/compliance-objects-bops";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
```

```
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date janFirst = format.parse("01/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.COMPLIANCE)
                            .withRetainUntilDate(janFirst)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Set compliance retain-until to 1 Jan 2025")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

Example 延長 **COMPLIANCE** 模式的 **retain until date**

下列範例將 COMPLIANCE 模式的 `retain until date` 延長到 2025 年 1 月 15 日。

```
public String createExtendComplianceRetentionJob(final AWSS3ControlClient
 awss3ControlClient) throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/compliance-objects-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
```

```
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/compliance-objects-bops";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan15th = format.parse("15/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.COMPLIANCE)
                            .withRetainUntilDate(jan15th)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Extend compliance retention to 15 Jan 2025")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);
```

```
    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

使用具有 S3 物件鎖定保留控管模式的 S3 批次操作

下列範例以先前建立信任政策的範例，以及設定 S3 批次操作和 S3 物件鎖定組態許可的範例為基礎。其中顯示如何對多個物件套用 `retain until date` 為 2020 年 1 月 30 日的 S3 物件鎖定保留控管。它會建立使用資訊清單儲存貯體的批次操作任務，並在報告儲存貯體中報告結果。

使用 AWS CLI

Example 將 S3 物件鎖定保留控管套用至 2025 年 1 月 30 日為止的多個物件

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-30T00:00:00",
      "Mode":"GOVERNANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/governance-objects-manifest.csv",
```

```
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucketT",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/governance-objects",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Put governance retention";
```

Example 繞過多個物件的保留控管

下列範例以先前建立信任政策的範例，以及設定 S3 批次操作和 S3 物件鎖定組態許可的範例為基礎。其中顯示如何繞過多個物件的保留控管，並建立使用資訊清單儲存貯體的批次操作任務，以及在報告儲存貯體中報告結果。

```
export AWS_PROFILE='aws-user'

read -d '' bypass_governance_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:BypassGovernanceRetention"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name bypass-governance-
permissions --policy-document "${bypass_governance_permissions}"

export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "BypassGovernanceRetention": true,
    "Retention": {
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/governance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
```

```
{
  "Bucket": "arn:aws:s3:::REPORT_BUCKET",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/bops-governance",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Remove governance retention";
```

使用適用於 Java 的 AWS 開發套件

下列範例以先前建立信任政策的範例，以及設定 S3 批次操作和 S3 物件鎖定組態許可的範例為基礎。本文說明如何在`retain until date`設定為 2025 年 1 月 30 日的情況下，將 S3 物件鎖定保留治理套用至多個物件。它會建立使用資訊清單儲存貯體的批次操作任務，並在報告儲存貯體中報告結果。

Example 將 S3 物件鎖定保留控管套用至 2025 年 1 月 30 日為止的多個物件

```
public String createGovernanceRetentionJob(final AWSS3ControlClient awss3ControlClient)
 throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/governance-objects-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");
```

```
    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/governance-objects";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan30th = format.parse("30/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.GOVERNANCE)
                            .withRetainUntilDate(jan30th)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Put governance retention")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

Example 繞過多個物件的保留控管

下列範例以先前建立信任政策的範例，以及設定 S3 批次操作和 S3 物件鎖定組態許可的範例為基礎。其中顯示如何繞過多個物件的保留控管，並建立使用資訊清單儲存貯體的批次操作任務，以及在報告儲存貯體中報告結果。

```
public void allowBypassGovernance() {
    final String roleName = "bops-object-lock";

    final String bypassGovernancePermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:BypassGovernanceRetention\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket/*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(bypassGovernancePermissions)
            .withPolicyName("bypass-governance-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
public String createRemoveGovernanceRetentionJob(final AWSS3ControlClient
 awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/governance-objects-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
```

```
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/bops-governance";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Remove governance retention")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

使用 S3 批次操作關閉 S3 物件鎖定法務保存

下列範例以先前建立信任原則的範例，以及設定 S3 批次操作和 S3 物件鎖定組態許可的範例為基礎。其中顯示如何使用批次操作對物件停用物件鎖定法務保存。

此範例會先更新角色以授予 s3:PutObjectLegalHold 許可、建立會關閉 (移除) 從資訊清單中識別之物件的法務保存的批次操作任務，然後報告該任務。

使用 AWS CLI

Example 更新角色以授予 **s3:PutObjectLegalHold** 許可

```
export AWS_PROFILE='aws-user'

read -d '' legal_hold_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectLegalHold"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        }
    ]

EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name legal-hold-
permissions --policy-document "${legal_hold_permissions}"
```

Example 關閉法務保存

下列範例會關閉法務保存。

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'
```

```
read -d '' OPERATION <<EOF
{
  "S3PutObjectLegalHold": {
    "LegalHold": {
      "Status":"OFF"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/legalhold-object-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/legalhold-objects-bops",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
```

```
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Turn off legal hold";
```

使用適用於 Java 的 AWS 開發套件

Example 更新角色以授予 **s3:PutObjectLegalHold** 許可

```
public void allowPutObjectLegalHold() {
    final String roleName = "bops-object-lock";

    final String legalHoldPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObjectLegalHold\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket/*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(legalHoldPermissions)
            .withPolicyName("legal-hold-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

Example 關閉法務保存

如果您想要關閉法務保存，請使用以下範例。

```
public String createLegalHoldOffJob(final AWSS3ControlClient awss3ControlClient) {
```

```
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/legalhold-object-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/legalhold-objects-bops";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectLegalHold(new S3SetObjectLegalHoldOperation()
                    .withLegalHold(new S3ObjectLockLegalHold()
                            .withStatus(S3ObjectLockLegalHoldStatus.OFF)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Turn off legal hold")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
```

```
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

S3 批次操作教學課程

下列教學課程會針對部分批次操作工作，提供完整的端對端程序。

- 教學課程：使用 S3 Batch 操作進行 Batch 轉碼影片，以及 AWS LambdaAWS Elemental MediaConvert

監控 Amazon S3

監控是維護 Amazon S3 和 AWS 解決方案的可靠性、可用性和效能的重要組成部分。我們建議您從 AWS 解決方案的所有部分收集監控資料，以便在發生多點故障時更輕鬆地對多點失敗進行偵錯。開始監控 Amazon S3 之前，請建立監控計劃來回答下列問題：

- 監控目標是什麼？
- 要監控哪些資源？
- 監控這些資源的頻率為何？
- 要使用哪些監控工具？
- 誰將執行監控任務？
- 發生問題時應該通知誰？

如需 Amazon S3 記錄與監控的詳細資訊，請參閱下列主題。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱什麼是 S3 Express One Zone？ 和 目錄值區。

主題

監控工具

AWS 提供各種可用來監控 Amazon S3 的工具。您可以設定其中一些工具來進行監控，但有些工具需要手動介入。建議您盡可能自動化監控任務。

自動化監控工具

您可以使用下列自動化監控工具來監看 Amazon S3，並在發生錯誤時回報：

- Amazon A CloudWatch lar ps — 觀看您指定期間內的單一指標，並根據指定臨界值在多個時段內相對於指定閾值的指標值執行一或多個動作。動作是傳送至亞馬遜簡單通知服務 (Amazon SNS) 主題或 Amazon EC2 Auto Scaling 政策的通知。 CloudWatch 警報不會僅僅因為它們處於特定狀態而叫用動作。狀態必須已變更，且在指定的期間數內維持此狀態。如需詳細資訊，請參閱 使用 Amazon 監控指標 CloudWatch。
- AWS CloudTrail 記錄監控 — 在帳戶之間共用記錄檔、即時監控記 CloudTrail 錄檔案，方法是將記錄檔傳送至 CloudWatch 記錄檔、使用 Java 撰寫記錄處理應用程式，以及驗證您的記錄檔在傳送之後未變更 CloudTrail。如需更多詳細資訊，請參閱 使用記錄 Amazon S3 API 呼叫 AWS CloudTrail。

手動監控工具

監控 Amazon S3 的另一個重要部分是手動監控 CloudWatch 警示未涵蓋的項目。Amazon S3、CloudWatch Trusted Advisor、和其他 AWS Management Console 儀表板可提供您 AWS 環境狀態的 at-a-glance 檢視。您可能想要啟用伺服器存取記錄，以追蹤存取儲存貯體的請求。每筆存取日誌記錄都會提供有關單一存取要求的詳細資訊，例如要求者、儲存貯體名稱、要求時間、要求動作、回應狀態及錯誤代碼 (若出現錯誤)。如需詳細資訊，請參閱 使用伺服器存取記錄記錄要求。

- Amazon S3 顯示以下內容：
  - 您的儲存貯體以及其所包含的物件與屬性
- CloudWatch 首頁會顯示下列內容：
  - 目前警示與狀態
  - 警示與資源的圖表
  - 服務運作狀態

  此外，您可以使用執行 CloudWatch 以下操作：
  - 建立自訂儀表板來監控您關心的服務。
  - 用於疑難排解問題以及探索驅勢的圖形指標資料。
  - 搜尋並瀏覽所有資 AWS 源指標。
  - 建立與編輯要通知發生問題的警示。

- AWS Trusted Advisor 可協助您監控 AWS 資源，以改善效能、可靠性、安全性和成本效益。所有使用者都可以使用四項 Trusted Advisor 檢查；具有商業或企業支援計畫的使用者可以使用超過 50 項以上的檢查。如需詳細資訊，請參閱 AWS Trusted Advisor。

  Trusted Advisor 有這些與 Amazon S3 相關的檢查：

  - 檢查 Amazon S3 儲存貯體的記錄組態。
  - 對於有開放式存取許可的 Amazon S3 儲存貯體，進行安全檢查。
  - 對於未啟用版本控制或已暫停版本控制的 Amazon S3 儲存貯體，進行容錯能力檢查。

Amazon S3 的記錄選項

您可以記錄使用者、角色或 Amazon S3 資源 AWS 服務 上採取的動作，並維護日誌記錄以供稽核和合規之用。為此，您可以使用伺服器存取日誌記錄或 AWS CloudTrail 日誌記錄，或兩者並用。我們建議您使 CloudTrail 用記錄 Amazon S3 資源的儲存貯體層級和物件層級動作。如需每個選項的詳細資訊，請參閱下列各節：

- 使用伺服器存取記錄記錄要求
- 使用記錄 Amazon S3 API 呼叫 AWS CloudTrail

下表列出 CloudTrail 日誌和 Amazon S3 伺服器存取日誌的主要屬性。若要確保 CloudTrail 符合您的安全性需求，請檢閱表格和注意事項。

日誌屬性	AWS CloudTrail	Amazon S3 伺服器日誌
可以轉發到其他系統（Amazon CloudWatch 日誌，Amazon CloudWatch 事件）	是	否
將日誌交付至一個以上的目的地 (例如，將相同的日誌傳送至兩個不同的儲存儲體)	是	否
開啟物件子集的日誌 (字首)	是	否
跨帳戶日誌交付 (不同帳戶所擁有的目標和來源儲存儲體)	是	否

日誌屬性	AWS CloudTrail	Amazon S3 伺服器日誌
使用數位簽章或雜湊進行日誌檔案的完整性驗證	是	否
日誌檔案的預設加密或加密選擇	是	否
物件操作 (使用 Amazon S3 API)	是	是
儲存貯體操作 (使用 Amazon S3 API)	是	是
日誌的可搜尋 UI	是	否
物件鎖定參數的欄位、Amazon S3 Select 的日誌記錄屬性	是	否
日誌記錄的 `Object Size`、`Total Time`、`Turn-Around Time` 及 `HTTP Referer` 的欄位	否	是
生命週期轉換、逾期、還原	否	是
批次刪除操作中的金鑰記錄	否	是
身分驗證失敗[1]	否	是
交付日誌的帳戶	儲存儲體擁有者[2]，及要求者	僅儲存貯體擁有者
Performance and Cost	AWS CloudTrail	Amazon S3 Server Logs
價格	管理事件 (首次交付) 是免費的；資料事件需支付費用 (除了日誌儲存費用之外)	僅需支付日誌儲存費用
日誌交付的速度	每 5 分鐘的資料事件；每 15 分鐘的管理事件	幾小時內

日誌屬性	AWS CloudTrail	Amazon S3 伺服器日誌
記錄格式	JSON	包含空格區隔的日誌檔案、換行分隔的記錄

備註

1. CloudTrail 不會針對驗證失敗的要求傳遞記錄檔 (其中提供的認證無效)。但是，它會包含身分驗證失敗之請求的日誌 (`AccessDenied`)，以及匿名使用者提出的請求。
2. 當帳戶對請求中的物件沒有完整存取權時，S3 儲存貯體擁有者會收到 CloudTrail 日誌。如需詳細資訊，請參閱 跨帳戶案例中的 Amazon S3 物件層級動作。
3. 當 VPC 端點政策拒絕 VPC 端點請求或在評估 VPC 政策之前失敗的請求時，S3 不支援將日誌或伺服器存取日誌交付給請求者或儲存貯體擁有者。 CloudTrail

使用記錄 Amazon S3 API 呼叫 AWS CloudTrail

Amazon S3 與一項服務整合 AWS CloudTrail，可提供使用者、角色或角色所採取的動作記錄 AWS 服務。 CloudTrail 以事件形式擷取 Amazon S3 的所有 API 呼叫。擷取的呼叫包括來自 Amazon S3 主控台的呼叫，以及對 Amazon S3 API 操作的程式碼呼叫。使用收集的資訊 CloudTrail，您可以判斷向 Amazon S3 發出的請求、提出請求的來源 IP 地址、提出請求的時間以及其他詳細資訊。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是使用根使用者還是使用者憑證提出。
- 請求是否代表 IAM 身分中心使用者提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務服務提出。

CloudTrail 在您創建帳戶 AWS 帳戶 時處於活動狀態，並且您自動可以訪問 CloudTrail 事件歷史記錄。 CloudTrail 事件歷史記錄提供了過去 90 天的記錄管理事件的可查看，可搜索，可下載和不可變的記錄。 AWS 區域若要取得更多資訊，請參閱《使用指南》中的〈AWS CloudTrail 使用 CloudTrail 事件歷程〉。查看活動歷史記錄不 CloudTrail收取任何費用。

如需過 AWS 帳戶 去 90 天內持續的事件記錄，請建立追蹤或 CloudTrailLake 事件資料存放區。

CloudTrail 小徑

追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。使用建立的所有系統線 AWS Management Console 都是多區域。您可以使用建立單一區域或多區域系統線。 AWS CLI建議您建立多區域追蹤，因為您會擷取帳戶 AWS 區域 中的所有活動。如果您建立單一區域追蹤，則只能檢視追蹤記錄中的 AWS 區域事件。如需有關追蹤的詳細資訊，請參閱《AWS CloudTrail 使用指南》中的「為您的建立追蹤」 AWS 帳戶和「為組織建立追蹤」。

您可以透 CloudTrail 過建立追蹤，免費將一份正在進行的管理事件副本傳遞到 Amazon S3 儲存貯體，但是需要支付 Amazon S3 儲存費用。如需有關 CloudTrail 定價的詳細資訊，請參閱AWS CloudTrail 定價。如需 Amazon S3 定價的相關資訊，請參閱 Amazon S3 定價。

CloudTrail 湖泊事件資料存放區

CloudTrail L@@ ake 可讓您針對事件執行 SQL 型查詢。 CloudTrail 湖將基於行的 JSON 格式的現有事件轉換為 Apache ORC 格式。ORC 是一種單欄式儲存格式，針對快速擷取資料進行了最佳化。系統會將事件彙總到事件資料存放區中，事件資料存放區是事件的不可變集合，其依據為您透過套用進階事件選取器選取的條件。套用於事件資料存放區的選取器控制哪些事件持續存在並可供您查詢。若要取得有關 CloudTrail Lake 的更多資訊，請參閱使用指南中的〈AWS CloudTrail 使用 AWS CloudTrail Lake〉。

CloudTrail Lake 事件資料存放區和查詢會產生費用。建立事件資料存放區時，您可以選擇要用於事件資料存放區的定價選項。此定價選項將決定擷取和儲存事件的成本，以及事件資料存放區的預設和最長保留期。如需有關 CloudTrail 定價的詳細資訊，請參閱AWS CloudTrail 定價。

日誌檔可存放於儲存貯體任意長時間，但您也可以定義 Amazon S3 生命週期規則，自動封存或刪除日誌檔案。預設情況下，將使用 Amazon S3 伺服器端加密 (SSE) 對日誌檔案進行加密。

將 CloudTrail 日誌與 Amazon S3 伺服器存取日誌和 CloudWatch 日誌搭配使用

AWS CloudTrail 日誌可提供 Amazon S3 中使用者、角色或 AWS 服務採取的動作記錄，而 Amazon S3 伺服器存取日誌則提供對 S3 儲存貯體發出請求的詳細記錄。如需不同日誌的運作方式與其屬性、效能與成本的詳細資訊，請參閱the section called “記錄選項”。

您可以將 AWS CloudTrail 日誌與 Amazon S3 的伺服器存取日誌搭配使用。 CloudTrail 日誌為您提供 Amazon S3 儲存貯體層級和物件層級操作的詳細 API 追蹤功能。Amazon S3 的伺服器存取日誌可讓您瞭解 Amazon S3 中資料的物件層級操作。如需伺服器存取日誌的詳細資訊，請參閱「使用伺服器存取記錄記錄要求」。

您也可以將 CloudTrail 日誌與 Amazon 一起用 CloudWatch 於 Amazon S3。 CloudTrail 與 CloudWatch Logs 整合可將擷取的 S3 儲存貯體層級 API 活動提供 CloudTrail 給您指定的日 CloudWatch 誌群組中的日 CloudWatch 誌串流。您可以建立 CloudWatch 警示以監控特定 API 活動，並在特定 API 活動發生時接收電子郵件通知。如需監視特定 API 活動之 CloudWatch 警示的詳細資訊，請參閱使AWS CloudTrail 用者指南。如需 CloudWatch 搭配 Amazon S3 搭配使用的詳細資訊，請參閱使用 Amazon 監控指標 CloudWatch。

> **Note**
>
> 當 VPC 端點政策拒絕 VPC 端點請求時，S3 不支援將 CloudTrail 日誌傳遞給請求者或儲存貯體擁有者。

CloudTrail 使用 Amazon S3 SOAP API 呼叫進行追蹤

CloudTrail 追蹤 Amazon S3 SOAP API 呼叫。透過 HTTP 的 Amazon S3 SOAP 支援已被取代，但仍可透過 HTTPS 取得。如需 Amazon S3 SOAP 支援的詳細資訊，請參閱 附錄 A：使用 SOAP API。

> **Important**
>
> SOAP 不支援較新的 Amazon S3 功能。我們建議您使用其他 API 或 AWS 開發套件。

透過 CloudTrail 記錄追蹤 Amazon S3 SOAP 動作

SOAP API 名稱	CloudTrail 記錄檔中使用的 API 事件名稱
ListAllMyBuckets	`ListBuckets`
CreateBucket	`CreateBucket`
DeleteBucket	`DeleteBucket`
GetBucketAccessControlPolicy	`GetBucketAcl`
SetBucketAccessControlPolicy	`PutBucketAcl`
GetBucketLoggingStatus	`GetBucketLogging`
SetBucketLoggingStatus	`PutBucketLogging`

如需 CloudTrail 和 Amazon S3 的相關資訊，請參閱下列主題：

主題

- Amazon S3 CloudTrail 活動
- CloudTrail Outposts 上 Amazon S3 和 S3 的日誌文件條目
- 啟用 S3 儲存貯體和物件的 CloudTrail 事件記錄
- 使用識別 Amazon S3 請求 CloudTrail

Amazon S3 CloudTrail 活動

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

本節提供 S3 記錄到的事件的相關資訊 CloudTrail。

Amazon S3 資料事件 CloudTrail

資料事件提供在資源上或在資源中執行的資源操作的相關資訊 (例如，讀取或寫入 Amazon S3 物件)。這些也稱為資料平面操作。資料事件通常是大量資料的活動。依預設， CloudTrail 不會記錄資料事件。 CloudTrail 事件歷史記錄不會記錄數據事件。

資料事件需支付額外的費用。如需有關 CloudTrail 定價的詳細資訊，請參閱AWS CloudTrail 定價。

您可以使用 CloudTrail 主控台或 CloudTrail API 操作記錄 Amazon S3 資源類型的資料事件。 AWS CLI有關如何記錄資料事件的詳細資訊，請參閱AWS CloudTrail 使用《使用指南》 AWS Command Line Interface中的記錄資料事件 AWS Management Console和記錄資料事件。

下表列出您可以記錄其資料事件的 Amazon S3 資源類型。[資料事件類型 (主控台)] 欄顯示可從主控台的 [資料事件類型 CloudTrail] 清單中選擇的值。resource .type 值欄會顯示**`resources.type`**值，您可以在使用或 API 設定進階事件選取器時指定這個值。 AWS CLI CloudTrail 記錄到資料 CloudTrail欄中的資料 API 會顯示 CloudTrail 針對資源類型記錄的 API 呼叫。

資料事件類型 (主控台)	resources.type 值	記錄到的資料 API CloudTrail
S3	`AWS::S3::Object`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • DeleteObjects • GetObject • GetObjectAcl • GetObjectAttributes • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • GetObjectTorrent • HeadObject • ListMultipartUploads • ListObjectVersions • ListObjects • ListParts • PutObject • PutObjectAcl • PutObjectLegalHold • PutObjectRetention • PutObjectTagging • RestoreObject • SelectObjectContent • UploadPart • UploadPartCopy

資料事件類型 (主控台)	resources.type 值	記錄到的資料 API CloudTrail
S3 存取點	`AWS::S3::Access Point`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject (僅限同一區域的副本) • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetBucketAcl • GetBucketCors • GetBucketLocation • GetBucketNotificationConfiguration • GetBucketPolicy • GetObject • GetObjectAcl • GetObjectAttributes • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • HeadBucket • HeadObject • ListMultipartUploads • ListObjects • ListObjectsV2 • ListObjectVersions • ListParts • Presign • PutObject • PutObjectLegalHold

資料事件類型 (主控台)	resources.type 值	記錄到的資料 API CloudTrail
		• PutObjectRetention • PutObjectAcl • PutObjectTagging • RestoreObject • UploadPart • UploadPartCopy (僅限同一區域的副本)

資料事件類型 (主控台)	resources.type 值	記錄到的資料 API CloudTrail
S3 Object Lambda	`AWS::S3ObjectLambda::AccessPoint`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject (僅限同一區域的副本) • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetObject • GetObjectAcl • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • HeadObject • ListMultipartUploads • ListObjects • ListObjectVersions • ListParts • PutObject • PutObjectLegalHold • PutObjectRetention • PutObjectAcl • PutObjectTagging • RestoreObject • UploadPart • WriteGetObjectResponse

資料事件類型 (主控台)	resources.type 值	記錄到的資料 API CloudTrail
S3 Outposts	`AWS::S3Outposts::Object`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject (僅限同一區域的副本) • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetObject • GetObjectTagging • HeadObject • ListMultipartUploads • ListObjects • ListObjectsV2 • ListParts • PutObject • PutObjectTagging • UploadPart • UploadPartCopy

您可以設定進階事件選取器來篩選`eventName`、`readOnly`、和`resources.ARN`欄位，以僅記錄對您很重要的事件。如需這些欄位的詳細資訊，請參閱 AWS CloudTrail API 參考AdvancedFieldSelector中的。

Amazon S3 管理事件 CloudTrail

Amazon S3 會將所有控制平面操作記錄為管理事件。如需 S3 API 操作的詳細資訊，請參閱 Amazon S3 API 參考資料。

如何 CloudTrail 擷取對 Amazon S3 發出的請求

依預設，會 CloudTrail 記錄過去 90 天內發出的 S3 儲存貯體層級 API 呼叫，但不記錄對物件發出的請求。儲存貯體層級呼叫包括

CreateBucket、DeleteBucket、PutBucketLifecycle、PutBucketPolicy 等事件。您可以在主控台上查看值區層級事件。 CloudTrail 但是，您無法在其中檢視資料事件 (Amazon S3 物件層級呼叫)，您必須剖析或查詢這些事件的 CloudTrail日誌。

透過日誌記錄追蹤 Amazon S3 帳戶層級動作 CloudTrail

CloudTrail 記錄帳戶層級動作。Amazon S3 記錄會與日誌檔中的其他 AWS 服務 記錄一起寫入。CloudTrail 根據時間週期和檔案大小決定何時建立和寫入新檔案。

本節中的表格列出記錄支援的 Amazon S3 帳戶層級動作。 CloudTrail

透過 CloudTrail 記錄追蹤的 Amazon S3 帳戶層級 API 動作會顯示為下列事件名稱。 CloudTrail 事件名稱與 API 動作名稱不同。例如， DeletePublicAccessBlock 是 DeleteAccountPublicAccessBlock。

- DeleteAccountPublicAccessBlock
- GetAccountPublicAccessBlock
- PutAccountPublicAccessBlock

透過記錄追蹤的 Amazon S3 儲存貯體層級動作 CloudTrail

根據預設，會針對一般用途值區 CloudTrail 記錄儲存貯體層級的動作。Amazon S3 記錄會與日誌檔中的其他 AWS 服務記錄一起寫入。 CloudTrail 根據時間週期和檔案大小決定何時建立和寫入新檔案。

本節列出記錄支援的 Amazon S3 儲存貯體層級動作。 CloudTrail

透過 CloudTrail 記錄追蹤的 Amazon S3 儲存貯體層級 API 動作會顯示為下列事件名稱。在某些情況下， CloudTrail 事件名稱與 API 動作名稱不同。例如，PutBucketLifecycleConfiguration 為 PutBucketLifecycle。

- CreateBucket
- DeleteBucket
- DeleteBucketAnalyticsConfiguration
- DeleteBucketCors
- DeleteBucketEncryption
- DeleteBucketIntelligentTieringConfiguration
- DeleteBucketInventoryConfiguration

- DeleteBucketLifecycle
- DeleteBucketMetricsConfiguration
- DeleteBucketOwnershipControls
- DeleteBucketPolicy
- DeleteBucketPublicAccessBlock
- DeleteBucketReplication
- DeleteBucketTagging
- GetAccelerateConfiguration
- GetBucketAcl
- GetBucketAnalyticsConfiguration
- GetBucketCors
- GetBucketEncryption
- GetBucketIntelligentTieringConfiguration
- GetBucketInventoryConfiguration
- GetBucketLifecycle
- GetBucketLocation
- GetBucketLogging
- GetBucketMetricsConfiguration
- GetBucketNotification
- GetBucketObjectLockConfiguration
- GetBucketOwnershipControls
- GetBucketPolicy
- GetBucketPolicyStatus
- GetBucketPublicAccessBlock
- GetBucketReplication
- GetBucketRequestPayment
- GetBucketTagging
- GetBucketVersioning
- GetBucketWebsite

- HeadBucket
- ListBuckets
- PutAccelerateConfiguration
- PutBucketAcl
- PutBucketAnalyticsConfiguration
- PutBucketCors
- PutBucketEncryption
- PutBucketIntelligentTieringConfiguration
- PutBucketInventoryConfiguration
- PutBucketLifecycle
- PutBucketLogging
- PutBucketMetricsConfiguration
- PutBucketNotification
- PutBucketObjectLockConfiguration
- PutBucketOwnershipControls
- PutBucketPolicy
- PutBucketPublicAccessBlock
- PutBucketReplication
- PutBucketRequestPayment
- PutBucketTagging
- PutBucketVersioning
- PutBucketWebsite

除了這些 API 操作之外，您也可以使用 OPTIONS 物件的物件層級動作。此動作會被視為 CloudTrail 記錄時的值區層級動作，因為動作會檢查值區的 CORS 組態。

透過記錄追蹤 S3 快速單區域儲存貯體層級 (區域 API 端點) 動作 CloudTrail

依預設，會將目錄值區的儲存貯體層級動作 CloudTrail 記錄為管理事件。`eventsource`適用於 S3 快速單一區域的 CloudTrail 管理事件是`s3express.amazonaws.com`。

> **Note**
>
> 對於 S3 Express 單一區域，不支援區域端點 (物件層級或資料平面) API 操作 (例如，`PutObject`或`GetObject`) 的 CloudTrail 記錄。

下列這些區域端點 API 作業會記錄到 CloudTrail。

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy
- PutBucketPolicy
- ListDirectoryBuckets

如需詳細資訊，請參閱 S3 Express One Zone 的安全最佳實務。

跨帳戶案例中的 Amazon S3 物件層級動作

以下是涉及跨帳戶案例中物件層級 API 呼叫的特殊使用案例，以及報告 CloudTrail 記錄的方式。CloudTrail 將日誌傳遞給請求者（進行 API 調用的帳戶），但在某些拒絕訪問的情況下，日誌條目被編輯或省略。設定跨帳戶存取時，請考慮本節中的範例。

> **Note**
>
> 這些範例假設記 CloudTrail 錄已適當設定。

範例 1：將記錄 CloudTrail傳送給值區擁有者

CloudTrail 即使值區擁有者沒有相同物件 API 作業的權限，也會將記錄傳送給值區擁有者。請考慮下列跨帳戶案例：

- 帳戶 A 擁有儲存貯體。
- 帳戶 B (申請者) 嘗試存取該儲存貯體中的物件。
- 帳戶 C 擁有物件。帳戶 C 可能與 A 帳戶相同，也可能不相同。

> **Note**
>
> CloudTrail 一律將物件層級 API 記錄提供給要求者 (帳戶 B)。此 CloudTrail 外，即使值區擁有者不擁有物件 (Accounts C) 或對該物件具有相同 API 作業的權限，也會將相同的記錄傳送給值區擁有者 (帳戶 A)。

範例 2： CloudTrail 不增殖用於設定物件 ACL 的電子郵件地址

請考慮下列跨帳戶案例：

- 帳戶 A 擁有儲存貯體。
- 帳戶 B (申請者) 會傳送要求以使用電子郵件地址設定物件 ACL 授予。如需 ACL 的詳細資訊，請參閱「存取控制清單 (ACL) 概觀」。

請求者會取得日誌與電子郵件資訊。不過，值區擁有者 (如果他們有資格接收記錄檔)，如範例 1 — 取得報告事件的記 CloudTrail 錄檔。不過，儲存貯體擁有者不會取得 ACL 組態資訊，特別是被授與者電子郵件地址與授予。日誌告訴儲存貯體擁有者的唯一資訊是帳戶 B 已提出 ACL API 呼叫。

CloudTrail Outposts 上 Amazon S3 和 S3 的日誌文件條目

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態可在 AWS CloudTrail 日誌、S3 庫存、S3 儲存鏡頭、Amazon S3 主控台中使用，以及作為和 AWS 開發套件中的額外 Amazon S3 API 回應標頭。 AWS Command Line Interface 如需詳細資訊，請參閱預設加密常見問答集。

事件代表來自任何來源的單一請求，並包括有關請求的 API 操作，操作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此事件不會以任何特定順序顯示。

如需詳細資訊，請參閱下列範例。

主題

- 範例：Amazon S3 的 CloudTrail 日誌檔項目

- 範例：Amazon S3 on Outposts 日誌檔項目

範例：Amazon S3 的 CloudTrail 日誌檔項目

下列範例顯示示範 GETService、PutBucketAcl和GetBucketVersioning動作的 CloudTrail 記錄項目。


```
{
    "Records": [
    {
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
        },
        "eventTime": "2019-02-01T03:18:19Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "ListBuckets",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "[]",
        "requestParameters": {
            "host": [
                "s3.us-west-2.amazonaws.com"
            ]
        },
        "responseElements": null,
        "additionalEventData": {
            "SignatureVersion": "SigV2",
            "AuthenticationMethod": "QueryString",
            "aclRequired": "Yes"
    },
        "requestID": "47B8E8D397DCE7A6",
        "eventID": "cdc4b7ed-e171-4cef-975a-ad829d4123e8",
        "eventType": "AwsApiCall",
        "recipientAccountId": "444455556666",
        "tlsDetails": {
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "clientProvidedHostHeader": "s3.amazonaws.com"
```

```
    }
    },
    {
       "eventVersion": "1.03",
       "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
        },
      "eventTime": "2019-02-01T03:22:33Z",
      "eventSource": "s3.amazonaws.com",
      "eventName": "PutBucketAcl",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "",
      "userAgent": "[]",
      "requestParameters": {
          "bucketName": "",
          "AccessControlPolicy": {
              "AccessControlList": {
                  "Grant": {
                      "Grantee": {
                          "xsi:type": "CanonicalUser",
                          "xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
                          "ID":
"d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
                       },
                      "Permission": "FULL_CONTROL"
                   }
              },
              "xmlns": "http://s3.amazonaws.com/doc/2006-03-01/",
              "Owner": {
                  "ID":
"d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
              }
          },
          "host": [
              "s3.us-west-2.amazonaws.com"
          ],
          "acl": [
              ""
          ]
```

```
      },
      "responseElements": null,
      "additionalEventData": {
          "SignatureVersion": "SigV4",
          "CipherSuite": "ECDHE-RSA-AES128-SHA",
          "AuthenticationMethod": "AuthHeader"
      },
      "requestID": "BD8798EACDD16751",
      "eventID": "607b9532-1423-41c7-b048-ec2641693c47",
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333",
      "tlsDetails": {
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    },
    {
      "eventVersion": "1.03",
      "userIdentity": {
          "type": "IAMUser",
          "principalId": "111122223333",
          "arn": "arn:aws:iam::111122223333:user/myUserName",
          "accountId": "111122223333",
          "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
          "userName": "myUserName"
        },
      "eventTime": "2019-02-01T03:26:37Z",
      "eventSource": "s3.amazonaws.com",
      "eventName": "GetBucketVersioning",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "",
      "userAgent": "[]",
      "requestParameters": {
          "host": [
              "s3.us-west-2.amazonaws.com"
          ],
          "bucketName": "example-s3-bucket1",
          "versioning": [
              ""
          ]
      },
      "responseElements": null,
      "additionalEventData": {
```

```
            "SignatureVersion": "SigV4",
            "CipherSuite": "ECDHE-RSA-AES128-SHA",
            "AuthenticationMethod": "AuthHeader"
    },
      "requestID": "07D681279BD94AED",
      "eventID": "f2b287f3-0df1-4961-a2f4-c4bdfed47657",
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333",
      "tlsDetails": {
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    }
  ]
}
```

範例：Amazon S3 on Outposts 日誌檔項目

Outposts 上的 Amazon S3 管理事件可透過以下方式 AWS CloudTrail取得。如需詳細資訊，請參閱 使用記錄 Amazon S3 API 呼叫 AWS CloudTrail。此外，您可以選擇為 AWS CloudTrail中的資料事件啟用日誌記錄功能。

追蹤是一種組態，能讓事件以日誌檔案的形式交付至您所指定區域的 S3 儲存貯體中。 CloudTrail Outposts 值區的記錄包含新欄位`edgeDeviceDetails`，可識別指定值區所在的 Outpost。

其他記錄欄位包括請求的動作、動作的日期和時間，以及請求參數。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示示範上PutObject動作的 CloudTrail 記錄項目`s3-outposts`。

```
{
      "eventVersion": "1.08",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "111122223333",
        "arn": "arn:aws:iam::111122223333:user/yourUserName",
        "accountId": "222222222222",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "yourUserName"
      },
      "eventTime": "2020-11-30T15:44:33Z",
      "eventSource": "s3-outposts.amazonaws.com",
```

```
      "eventName": "PutObject",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "26.29.66.20",
      "userAgent": "aws-cli/1.18.39 Python/3.4.10 Darwin/18.7.0 botocore/1.15.39",
      "requestParameters": {
        "expires": "Wed, 21 Oct 2020 07:28:00 GMT",
        "Content-Language": "english",
        "x-amz-server-side-encryption-customer-key-MD5": "wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY",
        "ObjectCannedACL": "BucketOwnerFullControl",
        "x-amz-server-side-encryption": "Aes256",
        "Content-Encoding": "gzip",
        "Content-Length": "10",
        "Cache-Control": "no-cache",
        "Content-Type": "text/html; charset=UTF-8",
        "Content-Disposition": "attachment",
        "Content-MD5": "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY",
        "x-amz-storage-class": "Outposts",
        "x-amz-server-side-encryption-customer-algorithm": "Aes256",
        "bucketName": "example-s3-bucket1",
        "Key": "path/upload.sh"
      },
      "responseElements": {
        "x-amz-server-side-encryption-customer-key-MD5": "wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY",
        "x-amz-server-side-encryption": "Aes256",
        "x-amz-version-id": "001",
        "x-amz-server-side-encryption-customer-algorithm": "Aes256",
        "ETag": "d41d8cd98f00b204e9800998ecf8427f"
      },
      "additionalEventData": {
        "CipherSuite": "ECDHE-RSA-AES128-SHA",
        "bytesTransferredIn": 10,
        "x-amz-id-2": "29xXQBV2O
+xOHKItvzY1suLv1i6A52E0zOX159fpfsItYd58JhXwKxXAXI4IQkp6",
        "SignatureVersion": "SigV4",
        "bytesTransferredOut": 20,
        "AuthenticationMethod": "AuthHeader"
      },
      "requestID": "8E96D972160306FA",
      "eventID": "ee3b4e0c-ab12-459b-9998-0a5a6f2e4015",
      "readOnly": false,
      "resources": [
        {
```

```
          "accountId": "222222222222",
          "type": "AWS::S3Outposts::Object",
          "ARN": "arn:aws:s3-outposts:us-east-1:YYY:outpost/op-01ac5d28a6a232904/
bucket/path/upload.sh"
        },
        {
          "accountId": "222222222222",
          "type": "AWS::S3Outposts::Bucket",
          "ARN": "arn:aws:s3-outposts:us-east-1:YYY:outpost/op-01ac5d28a6a232904/
bucket/"
        }
      ],
      "eventType": "AwsApiCall",
      "managementEvent": false,
      "recipientAccountId": "444455556666",
      "sharedEventID": "02759a4c-c040-4758-b84b-7cbaaf17747a",
      "edgeDeviceDetails": {
        "type": "outposts",
        "deviceId": "op-01ac5d28a6a232904"
      },
      "eventCategory": "Data"
    }
```

啟用 S3 儲存貯體和物件的 CloudTrail 事件記錄

您可以使用資 CloudTrail 料事件取得 Amazon S3 中儲存貯體和物件層級請求的相關資訊。若要啟用所有值區或特定值區清單的 CloudTrail 資料事件，您必須在中手動建立追蹤 CloudTrail。

> **Note**
>
> - 的預設設定 CloudTrail 是僅尋找管理事件。檢查以確保您已為帳戶啟用資料事件。
> - 您可以使用會產生高工作負載的 S3 儲存貯體來在很短的時間內快速產生大量的日誌。請注意您選擇為忙碌值區啟用 CloudTrail 資料事件的時間長度。

CloudTrail 將 Amazon S3 資料事件日誌存放在您選擇的 S3 儲存貯體中。請考慮在單獨的值區中使用值區， AWS 帳戶 以更好地將您可能擁有的多個值區中的事件組織到集中位置，以便更輕鬆地查詢和分析。 AWS Organizations 協助您建立 AWS 帳戶 連結至擁有您正在監視之儲存貯體的帳戶的帳戶。如需詳細資訊，請參閱什麼是 AWS Organizations？ 在《AWS Organizations 使用者指南》中。

記錄追蹤的資料事件時 CloudTrail，您可以選擇使用進階事件選取器或基本事件選取器。當您使用進階事件選取器在 CloudTrail 主控台中建立追蹤時，在資料事件區段中，您可以選擇記錄選取器範本的記錄所有事件，以記錄所有物件層級事件。使用基本事件選取器在 CloudTrail 主控台中建立追蹤時，您可以在資料事件區段中選取 [選取帳戶中的所有 S3 儲存貯體] 核取方塊，以記錄所有物件層級事件。

> **Note**
>
> - 這是為您的 AWS CloudTrail 資料事件儲存貯體建立生命週期組態的最佳實務。設定生命週期組態，在經過您認為必須稽核日誌的期間之後，定期移動日誌檔案。這麼做可降低 Athena 分析每個查詢時的資料量。如需詳細資訊，請參閱 在值區上設定生命週期組態。
> - 如需關於記錄格式的詳細資訊，請參閱 使用記錄 Amazon S3 API 呼叫 AWS CloudTrail。
> - 如需如何查詢 CloudTrail 日誌的範例，請參閱AWS 大數據部落格文章使用和 Amazon Athena 分析安全性、合規 AWS CloudTrail 和操作活動。

使用主控台啟用儲存貯體中物件的記錄

您可以使用 Amazon S3 主控台設定 AWS CloudTrail 追蹤，以記錄 S3 儲存貯體中物件的資料事件。 CloudTrail 支援記錄 Amazon S3 物件層級 API 操作，例如`GetObject`、`DeleteObject`、和`PutObject`。這些事件稱為資料事件。

依預設， CloudTrail 追蹤不會記錄資料事件，但您可以設定追蹤記錄您指定的 S3 儲存貯體的資料事件，或記錄所有 Amazon S3 儲存貯體的資料事件 AWS 帳戶。如需詳細資訊，請參閱 使用記錄 Amazon S3 API 呼叫 AWS CloudTrail。

CloudTrail 不會在 CloudTrail 事件歷史記錄中填入資料事件。此外，並非所有值區層級動作都會填入事件歷史記錄中 CloudTrail 。如需透過 CloudTrail 記錄追蹤的 Amazon S3 儲存貯體層級 API 動作的詳細資訊，請參閱。透過記錄追蹤的 Amazon S3 儲存貯體層級動作 CloudTrail 如需有關如何查詢 CloudTrail 日誌的詳細資訊，請參閱 AWS 知識中心文章，瞭解如何使用 Amazon CloudWatch 日誌篩選器模式和 Amazon Athena 查詢 CloudTrail 日誌。

若要設定追蹤來記錄 S3 儲存貯體的資料事件，您可以使用 AWS CloudTrail 主控台或 Amazon S3 主控台。如果您設定追蹤記錄中所有 Amazon S3 儲存貯體的資料事件 AWS 帳戶，則使用 CloudTrail 主控台會更容易。如需使用 CloudTrail 主控台設定追蹤以記錄 S3 資料事件的相關資訊，請參閱使用AWS CloudTrail 者指南中的資料事件。

> **Important**
>
> 資料事件需支付額外的費用。如需詳細資訊，請參閱 AWS CloudTrail 定價。

下列程序顯示如何使用 Amazon S3 主控台設定 CloudTrail 追蹤，以記錄 S3 儲存貯體的資料事件。

啟用 S3 儲存貯體中物件的 CloudTrail 資料事件記錄

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇儲存貯體名稱。
3. 選擇 Properties (屬性)。
4. 在AWS CloudTrail 資料事件下，選擇在中設定 CloudTrail。

   您可以建立新 CloudTrail 追蹤或重複使用現有追蹤，並設定要記錄在追蹤中的 Amazon S3 資料事件。如需有關如何在 CloudTrail 主控台中建立追蹤的資訊，請參閱《AWS CloudTrail 使用指南》中的使用主控台建立和更新追蹤。如需如何在 CloudTrail 主控台中設定 Amazon S3 資料事件記錄的相關資訊，請參閱AWS CloudTrail 使用者指南中的記錄 Amazon S3 物件的資料事件。

   > **Note**
   >
   > 如果您使用 CloudTrail 主控台或 Amazon S3 主控台設定追蹤以記錄 S3 儲存貯體的資料事件，Amazon S3 主控台會顯示該儲存貯體已啟用物件層級記錄功能。

若要停用 S3 儲存貯體中物件的 CloudTrail 資料事件記錄

1. 請登入 AWS Management Console 並開啟 CloudTrail 主控台，網址為 https://console.aws.amazon.com/cloudtrail/。
2. 在左側導覽窗格中，選擇追蹤。
3. 選擇您已建立來記錄儲存貯體之事件的追蹤名稱。
4. 在追蹤的詳細資訊窗格上，選擇右上角的停止記錄。
5. 在出現的對話方塊中，選擇停止記錄。

如需在建立 S3 儲存貯體時啟用物件層級記錄日誌的資訊，請參閱「建立儲存貯體」。

如需使用 S3 儲存貯體 CloudTrail 記錄的相關資訊，請參閱下列主題：

- 檢視 S3 儲存貯體的屬性
- 使用記錄 Amazon S3 API 呼叫 AWS CloudTrail
- AWS CloudTrail 使用使用指南中的 CloudTrail 記錄檔

使用識別 Amazon S3 請求 CloudTrail

在 Amazon S3 中，您可以使用 AWS CloudTrail 事件日誌識別請求。 AWS CloudTrail 這是識別 Amazon S3 請求的偏好方式，但如果您使用 Amazon S3 伺服器存取日誌，請參閱the section called “識別 S3 要求”。

主題

- 在日誌中識別對 Amazon S3 發出的 CloudTrail 請求
- 使用識別 Amazon S3 簽名版本 2 請求 CloudTrail
- 使用識別 S3 物件的存取權 CloudTrail

在日誌中識別對 Amazon S3 發出的 CloudTrail 請求

設定將事件傳遞 CloudTrail 到儲存貯體之後，您應該開始看到物件移至 Amazon S3 主控台上的目的地儲存貯體。格式如下所示：

```
s3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/Region/yyyy/mm/dd
```

記錄的事件 CloudTrail 會以壓縮的 gzipped JSON 物件形式存放在 S3 儲存貯體中。為了有效地找到請求，您應該使用 Amazon Athena 等服務來索引和查詢 CloudTrail 日誌。

如需 CloudTrail 和 Athena 的詳細資訊，請參閱 Amazon Athena 使用者指南中的使用分割投影在 Athena 中建立 AWS CloudTrail 日誌表格。

使用識別 Amazon S3 簽名版本 2 請求 CloudTrail

您可以使用 CloudTrail 事件日誌來識別在 Amazon S3 中用來簽署請求的 API 簽名版本。此功能相當重要，因為對 Signature 第 2 版的支援即將結束 (已淘汰)。之後，Amazon S3 將不再接受使用簽章第 2 版的請求，所有請求都必須使用簽章第 4 版來簽署。

強烈建議您使用 CloudTrail 來協助判斷是否有任何工作流程正在使用「簽名版本 2」簽署。透過將程式庫和程式碼升級為改用 Signature 第 4 版來修補這些工作流程，避免對您的業務造成影響。

如需詳細資訊，請參閱公告： AWS CloudTrail 針對 Amazon S3，在中新增了用於增強安全稽核的新欄位 AWS re:Post。

> **Note**
>
> CloudTrail Amazon S3 的事件將簽名版本包含在請求詳細資料中的金鑰名稱下的 'additionalEventData. 若要在 Amazon S3 中針對物件 (例如GET、PUT和請求) 發出的DELETE請求尋找簽名版本，您必須啟用 CloudTrail 資料事件。(此功能預設為關閉。)

AWS CloudTrail 是識別簽名版本 2 請求的首選方法。如果您使用 Amazon S3 伺服器存取日誌，請參閱 使用 Amazon S3 存取日誌來識別簽章第 2 版請求。

主題

- 識別 Amazon S3 簽章第 2 版請求的 Athena 查詢範例
- 分割 Signature 第 2 版資料

識別 Amazon S3 簽章第 2 版請求的 Athena 查詢範例

Example — 選取所有簽章第 2 版事件，並僅列印 **EventTime**、**S3_Action**、**Request_Parameters**、**Region**、**SourceIP** 和 **UserAgent**

在下列 Athena 查詢中，會將 *s3_cloudtrail_events_db.cloudtrail_table* 取代為 Athena 詳細資訊，並視需要提高或移除限制。

```
SELECT EventTime, EventName as S3_Action, requestParameters as Request_Parameters,
 awsregion as AWS_Region, sourceipaddress as Source_IP, useragent as User_Agent
FROM s3_cloudtrail_events_db.cloudtrail_table
WHERE eventsource='s3.amazonaws.com'
AND json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
LIMIT 10;
```

Example — 選取傳送簽章第 2 版流量的所有請求者

```
SELECT useridentity.arn, Count(requestid) as RequestCount
```

```
FROM s3_cloudtrail_events_db.cloudtrail_table
WHERE eventsource='s3.amazonaws.com'
    and json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
Group by useridentity.arn
```

分割 Signature 第 2 版資料

如果您要查詢的資料量很多，您可以建立分割資料表來降低 Athena 的成本與執行時間。

做法為建立含分割區的新資料表，如下所示。

```
CREATE EXTERNAL TABLE s3_cloudtrail_events_db.cloudtrail_table_partitioned(
     eventversion STRING,
     userIdentity STRUCT<
         type:STRING,
         principalid:STRING,
         arn:STRING,
         accountid:STRING,
         invokedby:STRING,
         accesskeyid:STRING,
         userName:STRING,
       sessioncontext:STRUCT<
                  attributes:STRUCT<
                  mfaauthenticated:STRING,
                  creationdate:STRING>,
                  sessionIssuer:STRUCT<
                  type:STRING,
                  principalId:STRING,
                  arn:STRING,
                  accountId:STRING,
                  userName:STRING>
             >
          >,
     eventTime STRING,
     eventSource STRING,
     eventName STRING,
     awsRegion STRING,
     sourceIpAddress STRING,
     userAgent STRING,
     errorCode STRING,
     errorMessage STRING,
     requestParameters STRING,
```

```
        responseElements STRING,
        additionalEventData STRING,
        requestId STRING,
        eventId STRING,
        resources ARRAY<STRUCT<ARN:STRING,accountId: STRING,type:STRING>>,
        eventType STRING,
        apiVersion STRING,
        readOnly STRING,
        recipientAccountId STRING,
        serviceEventDetails STRING,
        sharedEventID STRING,
        vpcEndpointId STRING
    )
    PARTITIONED BY (region string, year string, month string, day string)
    ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
    STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
    LOCATION 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/';
```

接著，請個別建立分割區。您無法從未建立的日期取得結果。

```
ALTER TABLE s3_cloudtrail_events_db.cloudtrail_table_partitioned ADD
    PARTITION (region= 'us-east-1', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/us-east-1/2019/02/19/'
    PARTITION (region= 'us-west-1', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/us-west-1/2019/02/19/'
    PARTITION (region= 'us-west-2', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/us-west-2/2019/02/19/'
    PARTITION (region= 'ap-southeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/ap-southeast-1/2019/02/19/'
    PARTITION (region= 'ap-southeast-2', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/ap-southeast-2/2019/02/19/'
    PARTITION (region= 'ap-northeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/ap-northeast-1/2019/02/19/'
    PARTITION (region= 'eu-west-1', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/eu-west-1/2019/02/19/'
    PARTITION (region= 'sa-east-1', year= '2019', month= '02', day= '19') LOCATION
 's3://DOC-EXAMPLE-BUCKET1/AWSLogs/111122223333/CloudTrail/sa-east-1/2019/02/19/';
```

您就可以根據這些分割區來提出請求，而不須載入整個儲存貯體。

```
SELECT useridentity.arn,
Count(requestid) AS RequestCount
FROM s3_cloudtrail_events_db.cloudtrail_table_partitioned
WHERE eventsource='s3.amazonaws.com'
AND json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
AND region='us-east-1'
AND year='2019'
AND month='02'
AND day='19'
Group by useridentity.arn
```

使用識別 S3 物件的存取權 CloudTrail

您可以使用 AWS CloudTrail 事件日誌來識別資料事件 (例如`GetObject`、和) 的 Amazon S3 物件存取請求 `DeleteObject``PutObject`，並探索有關這些請求的其他資訊。

下列範例顯示如何從 AWS CloudTrail 事PUT件日誌取得 Amazon S3 的所有物件請求。

主題

- 識別 Amazon S3 物件存取請求的 Athena 查詢範例

識別 Amazon S3 物件存取請求的 Athena 查詢範例

在下列 Athena 查詢範例中，會將 *s3_cloudtrail_events_db.cloudtrail_table* 取代為 Athena 詳細資訊，並視需要修改日期範圍。

Example — 選取具有 **PUT** 物件存取請求的所有事件，並僅列印 **EventTime**、**EventSource**、**SourceIP**、**UserAgent**、**BucketName**、**object** 和 **UserARN**

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  userIdentity.arn as userArn
FROM
```

```
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  eventName = 'PutObject'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example — 選取具有 **GET** 物件存取請求的所有事件，並僅列印 **EventTime**、**EventSource**、**SourceIP**、**UserAgent**、**BucketName**、**object** 和 **UserARN**

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  userIdentity.arn as userArn
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  eventName = 'GetObject'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example — 選取特定期間儲存貯體的所有匿名申請事件，並僅列印 **EventTime**、**EventName**、**EventSource**、**SourceIP**、**UserAgent**、**BucketName**、**UserARN** 和 **AccountID**

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  userIdentity.arn as userArn,
  userIdentity.accountId
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  userIdentity.accountId = 'anonymous'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example — 識別需要 ACL 進行授權的所有請求

下列 Amazon Athena 查詢範例示範如何識別對 S3 儲存貯體提出且需要存取控制清單 (ACL) 進行授權的所有請求。如果請求需要 ACL 進行授權，則 `additionalEventData` 中的 `aclRequired` 值為 `Yes`。如果不需要 ACL，則 `aclRequired` 不存在。您可以使用此資訊，將這些 ACL 許可遷移至適當的儲存貯體政策。在建立了這些儲存貯體政策之後，您可以針對這些儲存貯體停用 ACL。如需停用 ACL 的詳細資訊，請參閱 停用 ACL 的先決條件。

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  userIdentity.arn as userArn,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  json_extract_scalar(additionalEventData, '$.aclRequired') as aclRequired
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  json_extract_scalar(additionalEventData, '$.aclRequired') = 'Yes'
  AND eventTime BETWEEN '2022-05-10T00:00:00Z' and '2022-08-10T00:00:00Z'
```

Note

- 對安全監控時而言，這些查詢範例也可能相當實用。您可以檢閱來自意外或未授權 IP 地址或申請者的 `PutObject` 或 `GetObject` 呼叫的結果，以及識別對您儲存貯體的任何匿名請求。
- 此查詢只會擷取啟用日誌之後的資訊。

如果您使用 Amazon S3 伺服器存取日誌，請參閱 使用 Amazon S3 存取日誌來識別物件存取請求。

使用伺服器存取記錄記錄要求

伺服器存取記錄日誌，應儲存貯體要求，提出的詳細記錄。伺服器存取日誌對許多應用程式來說，都是個很有用的資料。舉例來說，存取記錄資訊在安全與存取稽核中相當實用。這些資訊也可幫助您了解自己的客戶群，並掌握 Amazon S3 帳單相關資料。

> **Note**
>
> 若區域是在 2019 年 3 月 20 日後才推出，則伺服器存取日誌不會記錄與其錯誤區域重新導向錯誤相關的資訊。當對物件或儲存貯體的請求在該儲存貯體存在的區域以外發出時，就會發生錯誤的區域重新導向錯誤。

如何啟用日誌交付？

若要啟用日誌傳遞，請執行下列基本步驟。如需詳細資訊，請參閱 啟用 Amazon S3 伺服器存取記錄日誌。

1. 提供目的地儲存貯體的名稱 (也稱為目標儲存貯體)。此儲存貯體是您希望 Amazon S3 將存取日誌儲存為物件的位置。來源和目的地儲存貯體必須位於同一 AWS 區域 ，並且由相同帳戶擁有。目的地儲存貯體不得具有 S3 物件鎖定預設保留期組態。目的地儲存貯體也必須未啟用「請求者付款」。

   您可將日誌交付給所有您擁有的儲存貯體，在同一的區域當做來源儲存貯體，包括來源儲存貯體本身。但為了更簡易進行記錄管理，建議您將存取記錄儲存在不同的儲存貯體中。

   當來源儲存貯體和目的地儲存貯體位於同一儲存貯體時，會為寫入儲存貯體的日誌建立額外的日誌，以建立日誌無限迴圈。不建議這麼做，因為它可能會導致您的儲存體帳單稍微增加。此外，額外的日誌可能會增加您的搜尋困難。

   若您選擇將存取日誌儲存在來源儲存貯體中，建議您為所有的日誌物件索引鍵指定目的地字首 (也稱為目標字首)。當您指定字首時，所有日誌物件名稱都會以通用字串開頭，如此就能更容易識別日誌物件。

2. (選用) 為所有 Amazon S3 日誌物件索引鍵指派目的地字首。目的地字首 (也稱為目標字首) 讓您更容易尋找日誌物件。例如，若指定的字首值為 `logs/`，則 Amazon S3 建立的每個日誌物件都會以 `logs/` 字首作為其索引鍵的開頭。

   ```
   logs/2013-11-01-21-32-16-E568B2907131C0C0
   ```

   如果您指定字首值 `logs`，則日誌物件會顯示如下：

   ```
   logs2013-11-01-21-32-16-E568B2907131C0C0
   ```

   當多個儲存貯體記錄到相同目的地儲存貯體時，字首也可用來區分來源儲存貯體。

此字首也可協助您刪除日誌。例如，您可為 Amazon S3 設定生命週期組態規則，刪除具有特定字首的物件。如需詳細資訊，請參閱 刪除 Amazon S3 日誌檔案。

3. (選用) 設定可讓其他人存取所產生日誌的許可。依預設，儲存貯體擁有者一律僅擁用日誌物件的完整存取權。如果目的地儲存貯體使用 S3 物件擁有權的儲存貯體擁有者強制執行設定來停用存取控制清單 (ACL)，則您無法在使用 ACL 的目的地授權 (也稱為目標授權) 中授予許可。但是，您可以更新目的地儲存貯體的儲存貯體政策，以將存取權授予其他使用者。如需詳細資訊，請參閱 適用於 Amazon S3 的 Identity and Access Management 及 日誌交付許可。
4. (選用) 設定日誌檔的日誌物件索引鍵格式。有兩種日誌物件索引鍵格式可供您選擇 (也稱為目標物件索引鍵格式)：

   - N on-date-based 分割 — 這是原始記錄物件索引鍵格式。如果您選擇此格式，日誌檔索引鍵格式會顯示如下：

     ```
     [DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
     ```

     例如，如果您指定 `logs/` 作為字首，則日誌物件的命名如下：

     ```
     logs/2013-11-01-21-32-16-E568B2907131C0C0
     ```

   - 日期行分割：如果您選擇日期型分割，則可以選擇日誌檔的事件時間或傳遞時間作為日誌格式中使用的日期來源。此格式較容易查詢日誌。

     如果您選擇日期型分割，日誌檔索引鍵格式會顯示如下：

     ```
     [DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/
     [DD]/[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
     ```

     例如，如果您指定 `logs/` 作為目標字首，則日誌物件的命名如下：

     ```
     logs/123456789012/us-west-2/DOC-EXAMPLE-SOURCE-
     BUCKET/2023/03/01/2023-03-01-21-32-16-E568B2907131C0C0
     ```

     對於傳遞時間傳遞，日誌檔名稱中的時間會與日誌檔的傳遞時間對應。

     對於事件時間傳遞，年、月和日會對應事件發生的日期，且索引鍵中的時、分和秒會設定為 00。這些日誌檔中傳遞的日誌僅適用特定的一天。

如果您是透過 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 Amazon S3 REST API 設定日誌，請使用`TargetObjectKeyFormat`來指定日誌物件金鑰格式。若要指定 non-date-based 分割區，請使用`SimplePrefix`。若要指定日期型分割，請使用 `PartitionedPrefix`。如果您使用 `PartitionedPrefix`，請使用 `PartitionDateSource` 指定 `EventTime` 或 `DeliveryTime`。

若是 `SimplePrefix`，日誌檔索引鍵格式顯示如下：

```
[TargetPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
```

若是具有事件時間或傳遞時間的 `PartitionedPrefix`，日誌檔索引鍵格式顯示如下：

```
[TargetPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/
[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
```

日誌物件金鑰格式

Amazon S3 會針對其上傳到目的地儲存貯體的日誌物件，使用下列物件索引鍵格式：

- N on-date-based 分割 — 這是原始記錄物件索引鍵格式。如果您選擇此格式，日誌檔索引鍵格式會顯示如下：

  ```
  [DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
  ```

- 日期行分割：如果您選擇日期型分割，則可以選擇日誌檔的事件時間或傳遞時間作為日誌格式中使用的日期來源。此格式較容易查詢日誌。

  如果您選擇日期型分割，日誌檔索引鍵格式會顯示如下：

  ```
  [DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/
  [YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
  ```

在日誌物件索引鍵中，YYYY、MM、DD、hh、mm 與 ss 分別是年、月、日、時、分與秒的數字。這些日期和時間都使用國際標準時間 (UTC)。

在特定時間交付的日誌檔，會包含該時間之前的任何時間點所寫入之記錄。但無法得知某特定時間間隔的所有日誌記錄是否皆已交付。

金鑰的 UniqueString 元件的存在原因，就是為了要避免覆寫檔案。它沒有任何意義，所以日誌處理軟體應會忽略它。

交付日誌的方式？

Amazon S3 會定期收集存取日誌記錄，將這些記錄整合為日誌檔，然後將日誌檔上傳至目的地儲存貯體作為日誌物件。若您對多個識別同一目的地儲存貯體的來源儲存貯體啟用了記錄功能，則此目的地儲存貯體就會有這些來源儲存貯體的存取日誌。但每個日誌物件都會回報特定來源儲存貯體的存取日誌記錄。

Amazon S3 使用特殊日誌交付帳戶來寫入伺服器存取日誌。這些寫入受一般的存取控制限制之約束。建議您更新目的地儲存貯體的儲存貯體政策，以授予日誌記錄服務主體的存取權 (logging.s3.amazonaws.com) 以進行存取日誌交付。您也可以透過儲存貯體存取控制清單 (ACL)，將存取日誌交付的存取權授予 S3 日誌交付群組。不過，不建議使用儲存貯體 ACL 授予 S3 日誌交付群組的存取權。

當啟用伺服器存取記錄並透過目的地儲存貯體政策授予存取日誌交付的存取權時，您必須更新政策以允許 s3:PutObject 存取記錄服務主體。如果您使用 Amazon S3 主控台來啟用伺服器存取記錄，則主控台會自動更新目的地儲存貯體政策，將這些許可授予記錄服務主體。如需有關授予伺服器存取日誌交付許可的詳細資訊，請參閱 日誌交付許可。

> **Note**
>
> 當 VPC 端點政策拒絕 VPC 端點請求或在評估 VPC 政策之前失敗的請求時，S3 不支援將日誌或伺服器存取日誌交付給請求者或儲存貯體擁有者。 CloudTrail

S3 物件擁有權的儲存貯體擁有者強制執行設定

如果目的地儲存貯體使用「物件擁有權」的儲存貯體擁有者強制設定，則 ACL 會停用且不再影響許可。您必須更新目的地儲存貯體上的儲存貯體政策，以授予記錄服務主體的存取權。如需「物件擁有權」的詳細資訊，請參閱 授予 S3 日誌交付群組的存取權以進行伺服器存取日誌記錄。

伺服器日誌交付最佳作法

伺服器存取日誌記錄會依最佳作法交付。大多數儲存貯體的要求，為日誌記錄結果適合組態，交付日誌記錄。大多數的日誌記錄會於記錄後的數小時內交付，但也可以常交付。

並不保證伺服器記錄的完成程度與時間先後順序。特定要求的日誌記錄，可能會在實際處理要求之後很久才交付，或者有可能完全不會交付。您甚至可能會看到日誌記錄的複寫。伺服器日誌的目的在於讓您

能了解儲存貯體流量的真實狀態。雖然日誌記錄極少遺失或重複，但請注意，伺服器記錄並不代表所有請求的完整記錄。

由於伺服器記錄的最佳作法特性，您的用量報告可能會包含一或多個未出現在已交付伺服器日誌中的存取請求。您可以在 AWS Billing and Cost Management 主控台的成本和用量報告下，找到這些用量報告。

儲存貯體記錄狀態變更會在一段時間後生效

記錄儲存貯體狀態的變更，要一段時間後才會實際影響到日誌檔交付。例如，若已啟用儲存貯體記錄，則在接下來的一小時內提出之要求，可能有些會記錄下來，有些則不會。假熱您將記錄的目的地儲存貯體從儲存貯體 A 變更為儲存貯體 B，則接下來的一小時內，有些日誌可能會繼續交付到儲存貯體 A，而有些則可能會交付到新的目的地儲存貯體 B。在任何情況之下，新的設定最後都會生效，您無須採取任何動作。

如需記錄和記錄檔案的詳細資訊，請參閱下列章節：

主題

- 啟用 Amazon S3 伺服器存取記錄日誌
- Amazon S3 伺服器存取日誌格式
- 刪除 Amazon S3 日誌檔案
- 使用 Amazon S3 伺服器存取日誌來識別請求

啟用 Amazon S3 伺服器存取記錄日誌

伺服器存取記錄，針對向 Amazon S3 儲存貯體提出的請求，提供詳細的記錄。伺服器存取日誌對許多應用程式來說，都是個很有用的資料。舉例來說，存取記錄資訊在安全與存取稽核中相當實用。這些資訊也可幫助您了解自己的客戶群，並掌握 Amazon S3 帳單相關資料。

Amazon S3 預設不會收集伺服器存取日誌。啟用記錄後，Amazon S3 會將來源儲存貯體的存取日誌交付給您所選擇的目的地儲存貯體 (也稱為目標儲存貯體)。目的地儲存貯體必須與來源儲存貯體位於相同 AWS 區域 和 AWS 帳戶 中。

存取日誌記錄包含對儲存貯體提出要求，內有詳細資訊。這資訊可能包含要求類型、要求中指定的資源，以及要求的處理時間和日期。如需記錄基本概念的詳細資訊，請參閱 使用伺服器存取記錄記錄要求。

Important

- 啟用 Amazon S3 儲存貯體的伺服器存取記錄日誌無須額外付費。不過，系統提供給您的任何日誌檔都會產生一般儲存費用 (您隨時都可刪除日誌檔)。我們不會評估日誌檔傳遞的資料傳輸費，但會收取存取日誌檔的一般資料傳輸費率。
- 您的目的地儲存貯體不應啟用伺服器存取記錄。您可將日誌交付給所有您擁有的儲存貯體，在同一的區域當做來源儲存貯體，包括來源儲存貯體本身。不過，將日誌交付至來源儲存貯體會導致日誌的無限迴圈，因此不建議這樣做。為了更簡易進行記錄管理，建議您將存取記錄儲存在不同的儲存貯體中。如需詳細資訊，請參閱 如何啟用日誌交付？。
- 已啟用 S3 物件鎖定的 S3 儲存貯體不能用作伺服器存取日誌的目的地儲存貯體。目的地儲存貯體不得具有預設保留期組態。
- 目的地儲存貯體不得啟用「請求者付款」。
- 只有在您使用伺服器端加密，搭配使用 256 位元進階加密標準 (AES-256) 的 Amazon S3 受管金鑰 (SSE-S3) 時，才能在目的地儲存貯體上使用預設儲存貯體加密。不支援使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 進行預設伺服器端加密。

您可以使用 Amazon S3 主控台、Amazon S3 API、 AWS Command Line Interface (AWS CLI) 或 AWS SDK，來啟用或停用伺服器存取記錄。

日誌交付許可

Amazon S3 使用特殊日誌交付帳戶來寫入伺服器存取日誌。這些寫入受一般的存取控制限制之約束。對於存取日誌交付，您必須將目的地儲存貯體存取權授予記錄服務主體 (`logging.s3.amazonaws.com`)。

若要授予 Amazon S3 進行日誌傳遞的許可，您可以使用儲存貯體政策或儲存貯體存取控制清單 (ACL)，取決於目的地儲存貯體的 S3 物件擁有權設定。不過，建議您使用儲存貯體政策，而不是 ACL。

S3 物件擁有權的儲存貯體擁有者強制執行設定

如果目的地儲存貯體使用「物件擁有權」的儲存貯體擁有者強制設定，則 ACL 會停用且不再影響許可。在此情況下，您必須更新目的地儲存貯體的儲存貯體政策，以授予記錄服務主體的存取權。您無法更新儲存貯體 ACL 以授予 S3 日誌交付群組的存取權。您也無法在 PutBucketLogging 組態中包含目的地授權 (也稱為目標授權)。

如需將存取日誌交付的現有儲存貯體 ACL 遷移至儲存貯體政策的相關資訊，請參閱 授予 S3 日誌交付群組的存取權以進行伺服器存取日誌記錄。如需「物件擁有權」的詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。建立新的儲存貯體時，預設會停用 ACL。

使用儲存貯體政策授予存取權

若要使用目的地儲存貯體上的儲存貯體政策授予存取權，請更新儲存貯體政策以將 `s3:PutObject` 許可授予記錄服務主體。如果您使用 Amazon S3 主控台來啟用伺服器存取記錄，則主控台會自動更新目的地儲存貯體上的儲存貯體政策，以將此許可授予記錄服務主體。如果您以程式設計方式啟用伺服器存取記錄，則必須手動更新目的地儲存貯體的儲存貯體政策，以將存取權授予記錄服務主體。

如需將存取權授予記錄服務主體的範例儲存貯體政策，請參閱 the section called “使用儲存貯體政策授予記錄服務主體的許可”。

使用儲存貯體 ACL 授予存取權

您可以交替使用儲存貯體 ACL 來授予存取日誌交付的存取權。您可以為將 `WRITE` 和 `READ_ACP` 許可授予 S3 日誌交付群組的儲存貯體 ACL 新增授予項目。不過，不建議使用儲存貯體 ACL 授予 S3 日誌交付群組的存取權。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。如需將存取日誌交付的現有儲存貯體 ACL 遷移至儲存貯體政策的相關資訊，請參閱 授予 S3 日誌交付群組的存取權以進行伺服器存取日誌記錄。如需將存取權授予記錄服務主體的範例 ACL，請參閱 the section called “使用儲存貯體 ACL 將許可授予日誌交付群組”。

使用儲存貯體政策授予記錄服務主體的許可

此範例儲存貯體政策會將 `s3:PutObject` 許可授予記錄服務主體 (`logging.s3.amazonaws.com`)。若要使用此儲存貯體政策，請以您自己的資訊取代 *`user input placeholders`*。在下列原則中，*`example-s3-destination-bucket`*是將傳送伺服器存取記錄的目的地儲存貯體，並且*`example-s3-source-bucket`*是來源儲存貯體。 *`EXAMPLE-LOGGING-PREFIX`*是您要用於記錄物件的選擇性目的地前置詞 (也稱為目標前置詞)。 *`SOURCE-ACCOUNT-ID`*是擁 AWS 帳戶 有源存儲桶的。

> **Note**
>
> 如果您的儲存貯體政策中有 `Deny` 陳述式，請確保它們不會阻止 Amazon S3 交付存取日誌。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::example-s3-destination-bucket/EXAMPLE-LOGGING-
PREFIX*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::example-s3-source-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

使用儲存貯體 ACL 將許可授予日誌交付群組

> **Note**
>
> 做為安全性最佳實務，根據預設，Amazon S3 會在所有新儲存貯體中停用存取控制清單 (ACL)。如需 Amazon S3 主控台中 ACL 許可的詳細資訊，請參閱 設定 ACL。

雖然我們不建議使用此方法，但您可以使用儲存貯體 ACL 將許可授予日誌交付群組。不過，如果目的地儲存貯體使用「物件擁有權」儲存貯體擁有者強制執行的設定，則您無法設定儲存貯體或物件 ACL。您也無法在 PutBucketLogging 組態中包含目的地授權 (也稱為目標授權)。反之，您必須使用儲存貯體政策，授予記錄服務主體 (logging.s3.amazonaws.com) 的存取權。如需詳細資訊，請參閱 日誌交付許可。

在儲存貯體 ACL 中，日誌交付群組會以下列 URL 表示：

```
http://acs.amazonaws.com/groups/s3/LogDelivery
```

若要授予 WRITE 和 READ_ACP (ACL 讀取) 許可，請將下列授權新增至目的地儲存貯體 ACL：

```
<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
    </Grantee>
    <Permission>WRITE</Permission>
</Grant>
<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
    </Grantee>
    <Permission>READ_ACP</Permission>
</Grant>
```

如需以程式設計方式新增 ACL 授權的範例，請參閱 設定 ACL。

> **Important**
>
> 當您在儲存貯體 AWS CloudFormation 上使用啟用 Amazon S3 伺服器存取日誌記錄，並使用 ACL 授與 S3 日誌交付群組的存取權時，還必須將 "新增AccessControl": "LogDeliveryWrite"至 CloudFormation 範本。這樣做很重要，因為您只能透過為值區建立 ACL 來授與這些權限，但無法為中 CloudFormation的值區建立自訂 ACL。您只能搭 CloudFormation配使用固定 ACL。

啟用伺服器存取日誌記錄

若要使用 Amazon S3 主控台、Amazon S3 REST API、 AWS 開發套件啟用伺服器存取記錄 AWS CLI，請使用下列程序。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇要啟用伺服器存取記錄日誌的儲存貯體名稱。
3. 選擇 Properties (屬性)。
4. 在 Server access logging (伺服器存取記錄日誌) 區段中，選擇 Edit (編輯)。
5. 在伺服器存取記錄下，選擇啟用。

6. 在目的地儲存貯體下，指定儲存貯體和選用的字首。如果您指定字首，建議您在字首後面加上正斜線 (/)，以便更容易找到日誌。

   > **Note**
   >
   > 指定包含斜線 (/) 的字首，可讓您更容易找到日誌物件。例如，若指定的字首值為 `logs/`，則 Amazon S3 建立的每個日誌物件都會以 `logs/` 字首作為其索引鍵的開頭，如下所示：
   >
   > ```
   > logs/2013-11-01-21-32-16-E568B2907131C0C0
   > ```
   >
   > 如果您指定字首值 `logs`，則日誌物件會顯示如下：
   >
   > ```
   > logs2013-11-01-21-32-16-E568B2907131C0C0
   > ```

7. 在日誌物件索引鍵格式下，執行下列其中一項操作：

   - 要選擇 non-date-based 分割區，請選擇 [DestinationPrefix] [YYYY]-[MM]-[DD]-[hh]-[毫米]-[ss]-[UniqueString]。
   - 若要選擇以日期為基礎的分割，請選擇 [DestinationPrefixSourceAccountIdSourceRegion] [SourceBucket]/[YYYY]/[MM]/[DD]/[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]，然後選擇 S3 事件時間或記錄檔傳送時間。

8. 選擇儲存變更。

   在儲存貯體上啟用伺服器存取記錄時，主控台會啟用來源儲存貯體上的記錄，同時更新目的地儲存貯體的儲存貯體政策，以將 `s3:PutObject` 許可授予記錄服務主體 (`logging.s3.amazonaws.com`)。如需此儲存貯體政策的詳細資訊，請參閱 使用儲存貯體政策授予記錄服務主體的許可。

   您可以在目的地儲存貯體中檢視日誌。啟用伺服器存取記錄後，可能需要好幾個小時才能將記錄傳遞到目標儲存貯體中。如需如何以及何時交付日誌的詳細資訊，請參閱 交付日誌的方式？。

如需詳細資訊，請參閱 檢視 S3 儲存貯體的屬性。

使用 REST API

若要啟用記錄，請提交 PutBucketLogging 請求，以在來源儲存貯體上新增記錄組態。請求會指定目的地儲存貯體 (也稱為目標儲存貯體)，以及選擇性地指定要搭配所有日誌物件索引鍵使用的字首。

下列範例將 *example-s3-destination-bucket* 識別為目的地儲存貯體，並將 *logs/* 識別為字首。

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>example-s3-destination-bucket</TargetBucket>
    <TargetPrefix>logs/</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
```

下列範例將 *example-s3-destination-bucket* 識別為目的地儲存貯體、將 *logs/* 識別為字首，並將 EventTime 識別為日誌物件索引鍵格式。

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>example-s3-destination-bucket</TargetBucket>
    <TargetPrefix>logs/</TargetPrefix>
    <TargetObjectKeyFormat>
      <PartitionedPrefix>
         <PartitionDateSource>EventTime</PartitionDateSource>
      </PartitionedPrefix>
  </TargetObjectKeyFormat>
  </LoggingEnabled>
</BucketLoggingStatus>
```

日誌物件由 S3 日誌交付帳戶寫入並擁有，且已為儲存貯體擁有者授予日誌物件的完整許可。您可以選擇使用目的地授權 (也稱為目標授權) 將許可授予其他使用者，讓他們能夠存取日誌。如需詳細資訊，請參閱 PutBucketLogging。

> **Note**
>
> 如果目的地儲存貯體使用「物件擁有權」的儲存貯體擁有者強制執行設定，則您無法使用目的地授權將許可授予其他使用者。若要將許可授予其他人，您可以在目的地儲存貯體上更新儲存貯體政策。如需詳細資訊，請參閱 日誌交付許可。

若要擷取儲存貯體上的記錄組態，請使用 GetBucketLogging API 操作。

若要刪除記錄組態，請傳送包含空白 BucketLoggingStatus 的 PutBucketLogging 請求：

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
```

```
</BucketLoggingStatus>
```

若要在儲存貯體上啟用記錄功能，您可以使用 Amazon S3 API 或 AWS SDK 包裝函式庫。

使用 AWS 軟體開發套件

下列範例會在儲存貯體上啟用記錄。您必須建立兩個儲存貯體，一個來源儲存貯體與一個目的地 (目標) 儲存貯體。這些範例會先更新目的地儲存貯體上的儲存貯體 ACL。然後將必要的許可授予日誌交付群組，以將日誌寫入目的地儲存貯體，接著再啟用來源儲存貯體上的記錄。

這些範例不適用於使用「物件擁有權」之儲存貯體擁有者強制執行設定的目的地儲存貯體。

如果目的地 (目標) 儲存貯體使用「物件擁有權」儲存貯體擁有者強制執行的設定，則您無法設定儲存貯體或物件 ACL。您也無法在PutBucketLogging設定中包含目的地 (目標) 授與。您必須使用儲存貯體政策授予日誌記錄服務主體 (logging.s3.amazonaws.com) 的存取權。如需詳細資訊，請參閱 日誌交付許可。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
/// logs.
/// </summary>
public class ServerAccessLogging
```

```
    {
        private static IConfiguration _configuration = null!;

        public static async Task Main()
        {
            LoadConfig();

            string bucketName = _configuration["BucketName"];
            string logBucketName = _configuration["LogBucketName"];
            string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
            string accountId = _configuration["AccountId"];

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Update bucket policy for target bucket to allow delivery of
logs to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);

                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }

        /// <summary>
        /// This method grants appropriate permissions for logging to the
```

```
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
        /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" +
logPrefix + @"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
                                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
                                ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3:::" + sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
accountId + @""" }
                                        }
                                    }]
                                }";
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
            Console.WriteLine(newPolicy);

            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
```

```
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
        /// This method enables logging for an Amazon S3 bucket. Logs will be
 stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
 used
        /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
 you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
 logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
                BucketName = bucketName,
                LoggingConfig = loggingConfig,
```

```
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
                .AddJsonFile("settings.local.json", true) // Optionally, load
  local settings.
                .Build();
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考[PutBucketLogging](#)中的。

Java

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLoggingStatus;
import software.amazon.awssdk.services.s3.model.LoggingEnabled;
import software.amazon.awssdk.services.s3.model.PartitionedPrefix;
import software.amazon.awssdk.services.s3.model.PutBucketLoggingRequest;
import software.amazon.awssdk.services.s3.model.TargetObjectKeyFormat;

// Class to set a bucket policy on a target S3 bucket and enable server access
 logging on a source S3 bucket.
public class ServerAccessLogging {
    private static S3Client s3Client;

    public static void main(String[] args) {
        String sourceBucketName = "SOURCE-BUCKET";
        String targetBucketName = "TARGET-BUCKET";
        String sourceAccountId = "123456789012";
        String targetPrefix = "logs/";
```

```
        // Create S3 Client.
        s3Client = S3Client.builder().
                region(Region.US_EAST_2)
                .build();

        // Set a bucket policy on the target S3 bucket to enable server access
 logging by granting the
        // logging.s3.amazonaws.com principal permission to use the PutObject
 operation.
        ServerAccessLogging serverAccessLogging = new ServerAccessLogging();
        serverAccessLogging.setTargetBucketPolicy(sourceAccountId, sourceBucketName,
 targetBucketName);

        // Enable server access logging on the source S3 bucket.
        serverAccessLogging.enableServerAccessLogging(sourceBucketName,
 targetBucketName,
                targetPrefix);

    }

    // Function to set a bucket policy on the target S3 bucket to enable server
 access logging by granting the
    // logging.s3.amazonaws.com principal permission to use the PutObject operation.
    public void setTargetBucketPolicy(String sourceAccountId, String
 sourceBucketName, String targetBucketName) {
        String policy = "{\n" +
                "    \"Version\": \"2012-10-17\",\n" +
                "    \"Statement\": [\n" +
                "        {\n" +
                "            \"Sid\": \"S3ServerAccessLogsPolicy\",\n" +
                "            \"Effect\": \"Allow\",\n" +
                "            \"Principal\": {\"Service\": \"logging.s3.amazonaws.com
\"},\n" +
                "            \"Action\": [\n" +
                "                \"s3:PutObject\"\n" +
                "            ],\n" +
                "            \"Resource\": \"arn:aws:s3:::" + targetBucketName + "/*
\",\n" +
                "            \"Condition\": {\n" +
                "                \"ArnLike\": {\n" +
                "                    \"aws:SourceArn\": \"arn:aws:s3:::" +
 sourceBucketName + "\"\n" +
                "                },\n" +
```

```
                "                \"StringEquals\": {\n" +
                "                    \"aws:SourceAccount\": \"" + sourceAccountId +
"\"\n" +
                "                }\n" +
                "            }\n" +
                "        }\n" +
                "    ]\n" +
                "}";
        s3Client.putBucketPolicy(b -> b.bucket(targetBucketName).policy(policy));
    }

    // Function to enable server access logging on the source S3 bucket.
    public void enableServerAccessLogging(String sourceBucketName, String
targetBucketName,
            String targetPrefix) {
        TargetObjectKeyFormat targetObjectKeyFormat =
TargetObjectKeyFormat.builder()

.partitionedPrefix(PartitionedPrefix.builder().partitionDateSource("EventTime").build())
                .build();
        LoggingEnabled loggingEnabled = LoggingEnabled.builder()
                .targetBucket(targetBucketName)
                .targetPrefix(targetPrefix)
                .targetObjectKeyFormat(targetObjectKeyFormat)
                .build();
        BucketLoggingStatus bucketLoggingStatus = BucketLoggingStatus.builder()
                .loggingEnabled(loggingEnabled)
                .build();
        s3Client.putBucketLogging(PutBucketLoggingRequest.builder()
                .bucket(sourceBucketName)
                .bucketLoggingStatus(bucketLoggingStatus)
                .build());
    }

}
```

使用 AWS CLI

我們建議您在每個擁有 S3 儲存貯體的每個儲存貯體中 AWS 區域 建立專用的記錄儲存貯體。然後將 Amazon S3 存取日誌交付至該 S3 儲存貯體。如需詳細資訊，請參閱《AWS CLI 參考》中的 put-bucket-logging。

如果目的地 (目標) 儲存貯體使用「物件擁有權」儲存貯體擁有者強制執行的設定，則您無法設定儲存貯體或物件 ACL。您也無法在PutBucketLogging設定中包含目的地 (目標) 授與。您必須使用儲存貯體政策授予日誌記錄服務主體 (logging.s3.amazonaws.com) 的存取權。如需詳細資訊，請參閱 日誌交付許可。

Example — 對跨兩個區域的五個儲存貯體啟用存取日誌

在此範例中，您擁有下列五個儲存貯體：

- 1-DOC-EXAMPLE-BUCKET1-us-east-1
- 2-DOC-EXAMPLE-BUCKET1-us-east-1
- 3-DOC-EXAMPLE-BUCKET1-us-east-1
- 1-DOC-EXAMPLE-BUCKET1-us-west-2
- 2-DOC-EXAMPLE-BUCKET1-us-west-2

> **Note**
>
> 下列程序的最後一個步驟提供範例 bash 指令碼，您可以使用這些指令碼建立記錄儲存貯體，並在這些儲存貯體上啟用伺服器存取記錄。若要使用這些指令碼，您必須建立 policy.json 和 logging.json 檔案，如下列程序所述。

1. 在美國西部 (奧勒岡) 和美國東部 (維吉尼亞北部) 區域建立兩個記錄目的地儲存貯體，並為其命名如下：
   - DOC-EXAMPLE-BUCKET1-logs-us-east-1
   - DOC-EXAMPLE-BUCKET1-logs-us-west-2
2. 稍後在這些步驟中，您將啟用伺服器存取記錄，如下所示：
   - 1-DOC-EXAMPLE-BUCKET1-us-east-1 記錄到 S3 儲存貯體 DOC-EXAMPLE-BUCKET1-logs-us-east-1，帶有字首 1-DOC-EXAMPLE-BUCKET1-us-east-1
   - 2-DOC-EXAMPLE-BUCKET1-us-east-1 記錄到 S3 儲存貯體 DOC-EXAMPLE-BUCKET1-logs-us-east-1，帶有字首 2-DOC-EXAMPLE-BUCKET1-us-east-1
   - 3-DOC-EXAMPLE-BUCKET1-us-east-1 記錄到 S3 儲存貯體 DOC-EXAMPLE-BUCKET1-logs-us-east-1，帶有字首 3-DOC-EXAMPLE-BUCKET1-us-east-1

- 1-DOC-EXAMPLE-BUCKET1-us-west-2 記錄到 S3 儲存貯體 DOC-EXAMPLE-BUCKET1-logs-us-west-2，帶有字首 1-DOC-EXAMPLE-BUCKET1-us-west-2
- 2-DOC-EXAMPLE-BUCKET1-us-west-2 記錄到 S3 儲存貯體 DOC-EXAMPLE-BUCKET1-logs-us-west-2，帶有字首 2-DOC-EXAMPLE-BUCKET1-us-west-2

3. 針對每個目的地記錄儲存貯體，使用儲存貯體 ACL 或儲存貯體政策，授予伺服器存取日誌交付的許可：

- 更新儲存貯體政策 (建議) - 若要將許可授予記錄服務主體，請使用下列 put-bucket-policy 命令：用您的目的地儲存貯體名稱取代 *example-s3-destination-bucket-logs*。

```
aws s3api put-bucket-policy --bucket example-s3-destination-bucket-logs --policy
 file://policy.json
```

Policy.json 為 JSON 文件，其位於包含下列儲存貯體政策的目前資料夾中。若要使用此儲存貯體政策，請以您自己的資訊取代 *user input placeholders*。在下列政策中，*example-s3-destination-bucket-logs* 是將交付伺服器存取日誌的目的地儲存貯體，而 *example-s3-source-bucket* 是來源儲存貯體。*SOURCE-ACCOUNT-ID* 是擁有來源儲存貯體的 AWS 帳戶 。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::example-s3-destination-bucket-logs/*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::example-s3-source-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-ACCOUNT-ID"
                }
```

```
            }
        }
    ]
}
```

- 更新儲存貯體 ACL - 若要將許可授予 S3 日誌交付群組，請使用下列 put-bucket-acl 命令。將 *example-s3-destination-bucket-logs* 取代為您的目的地 (目標) 儲存貯體名稱。

```
aws s3api put-bucket-acl --bucket example-s3-destination-bucket-logs  --grant-
write URI=http://acs.amazonaws.com/groups/s3/LogDelivery --grant-read-acp
 URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

4. 然後建立包含記錄組態的 logging.json 檔案 (根據以下三個範例之一)。建立 logging.json 檔案之後，您可以使用下列 put-bucket-logging 命令套用記錄組態。將 *example-s3-destination-bucket-logs* 取代為您的目的地 (目標) 儲存貯體名稱。

```
aws s3api put-bucket-logging --bucket example-s3-destination-bucket-logs --bucket-
logging-status file://logging.json
```

> **Note**
>
> 您可以使用下一個步驟中提供的其中一個 bash 指令碼，在每一個目的地儲存貯體上套用記錄組態，而不要使用此 put-bucket-logging 命令。若要使用這些指令碼，您必須建立 policy.json 和 logging.json 檔案，如此程序中所述。

logging.json 檔案為 JSON 文件，其位於包含記錄組態的目前資料夾中。如果目的地儲存貯體使用「物件擁有權」的儲存貯體擁有者強制執行設定，則您的記錄組態無法包含目的地 (目標) 授權。如需詳細資訊，請參閱 日誌交付許可。

Example - **logging.json**，沒有目的地 (目標) 授權

以下範例 logging.json 檔案不包含目的地 (目標) 授權。因此，您可以將此組態套用至使用「物件擁有權」的儲存貯體擁有者強制執行設定的目的地 (目標) 儲存貯體。

```
{
    "LoggingEnabled": {
        "TargetBucket": "example-s3-destination-bucket-logs",
        "TargetPrefix": "example-s3-destination-bucket/"
     }
 }
```

Example - **logging.json**，有目的地 (目標) 授權

以下範例 logging.json 檔案包含目的地 (目標) 授權。

如果目的地儲存貯體使用「物件擁有權」的儲存貯體擁有者強制執行設定，則無法將目的地 (目標) 授權納入您的 PutBucketLogging 組態中。如需詳細資訊，請參閱 日誌交付許可。

```
{
    "LoggingEnabled": {
        "TargetBucket": "example-s3-destination-bucket-logs",
        "TargetPrefix": "example-s3-destination-bucket/",
        "TargetGrants": [
             {
                "Grantee": {
                    "Type": "AmazonCustomerByEmail",
                    "EmailAddress": "user@example.com"
                 },
                "Permission": "FULL_CONTROL"
             }
         ]
     }
 }
```

Example - **logging.json**，其日誌物件索引鍵格式設定為 S3 事件時間

下列 logging.json 檔案會將日誌物件索引鍵格式變更為 S3 事件時間。如需設定日誌物件索引鍵格式的詳細資訊，請參閱 the section called “如何啟用日誌交付？”。

```
{
```

```
    "LoggingEnabled": {
        "TargetBucket": "example-s3-destination-bucket-logs",
        "TargetPrefix": "example-s3-destination-bucket/",
        "TargetObjectKeyFormat": {
            "PartitionedPrefix": {
                "PartitionDateSource": "EventTime"
            }
         }
    }
}
```

5. 使用下列其中一個 bash 指令碼，在您的帳戶中為所有儲存貯體新增存取記錄。將 *example-s3-destination-bucket-logs* 取代為目的地 (目標) 儲存貯體的名稱，並將 *us-west-2* 取代為您的儲存貯體所在區域的名稱。

> **Note**
>
> 只在您的所有儲存貯體都位在相同區域時，此指令碼才能運作。若您的儲存貯體位於多個區域，您必須調整指令碼。

Example – 授予儲存貯體的存取權，並為您的帳戶中的儲存貯體新增日誌記錄

```
loggingBucket='example-s3-destination-bucket-logs'
region='us-west-2'


# Create the logging bucket.
aws s3 mb s3://$loggingBucket --region $region

aws s3api put-bucket-policy --bucket $loggingBucket --policy file://policy.json

# List the buckets in this account.
buckets="$(aws s3 ls | awk '{print $3}')"

# Put a bucket logging configuration on each bucket.
for bucket in $buckets
    do
      # This if statement excludes the logging bucket.
      if [ "$bucket" != "$loggingBucket" ] ; then
```

```
            continue;
        fi
        printf '{
          "LoggingEnabled": {
            "TargetBucket": "%s",
            "TargetPrefix": "%s/"
        }
      }' "$loggingBucket" "$bucket"  > logging.json
      aws s3api put-bucket-logging --bucket $bucket --bucket-logging-status file://
logging.json
      echo "$bucket done"
  done

  rm logging.json

  echo "Complete"
```

Example – 授予儲存貯體 ACL 的存取權，並為您的帳戶中的儲存貯體新增日誌記錄

```
  loggingBucket='example-s3-destination-bucket-logs'
  region='us-west-2'


  # Create the logging bucket.
  aws s3 mb s3://$loggingBucket --region $region

  aws s3api put-bucket-acl --bucket $loggingBucket --grant-write URI=http://
acs.amazonaws.com/groups/s3/LogDelivery --grant-read-acp URI=http://
acs.amazonaws.com/groups/s3/LogDelivery

  # List the buckets in this account.
  buckets="$(aws s3 ls | awk '{print $3}')"

  # Put a bucket logging configuration on each bucket.
  for bucket in $buckets
      do
        # This if statement excludes the logging bucket.
        if [ "$bucket" != "$loggingBucket" ] ; then
            continue;
        fi
        printf '{
          "LoggingEnabled": {
```

```
            "TargetBucket": "%s",
            "TargetPrefix": "%s/"
        }
      }' "$loggingBucket" "$bucket"  > logging.json
      aws s3api put-bucket-logging --bucket $bucket --bucket-logging-status file://
logging.json
      echo "$bucket done"
  done

  rm logging.json

  echo "Complete"
```

驗證您的伺服器存取日誌設定

啟用伺服器存取記錄之後，請完成以下步驟：

- 存取目的地儲存貯體，並驗證是否正在交付日誌檔。在設定存取日誌之後，所有請求可能需要一個小時以上的時間才能適當地記錄和交付。您也可以使用 Amazon S3 請求指標，並為這些指標設定 Amazon CloudWatch 警示，自動驗證日誌交付。如需詳細資訊，請參閱 使用 Amazon 監控指標 CloudWatch。
- 驗證您是否能夠開啟和讀取日誌檔的內容。

如需伺服器存取記錄疑難排解資訊，請參閱 針對伺服器存取記錄進行疑難排解。

Amazon S3 伺服器存取日誌格式

伺服器存取記錄，針對向 Amazon S3 儲存貯體提出的請求，提供詳細的記錄。您可以將伺服器存取日誌用於下列目的：

- 執行安全與存取稽核
- 了解您的客戶群
- 了解您的 Amazon S3 計費

本節說明 Amazon S3 伺服器存取日誌檔案的格式和其他詳細資訊。

伺服器存取記錄檔，是由一連串換行分隔日誌記錄所組成。每筆日誌記錄都代表一項要求，且由多個空格分隔欄位所組成。

以下為五筆日誌記錄所組成的日誌範例。

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DOC-EXAMPLE-BUCKET1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 3E57427F3EXAMPLE
 REST.GET.VERSIONING - "GET /DOC-EXAMPLE-BUCKET1?versioning HTTP/1.1" 200 - 113 - 7 -
 "-" "S3Console/0.4" - s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/
XV/VLi31234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader DOC-EXAMPLE-BUCKET1.s3.us-
west-1.amazonaws.com TLSV1.2 arn:aws:s3:us-west-1:123456789012:accesspoint/example-AP
 Yes
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DOC-EXAMPLE-BUCKET1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 891CE47D2EXAMPLE
 REST.GET.LOGGING_STATUS - "GET /DOC-EXAMPLE-BUCKET1?logging HTTP/1.1" 200 -
 242 - 11 - "-" "S3Console/0.4" - 9vKBE6vMhrNiWHZmb2L0mXOcqPGzQOI5XLnCtZNPxev+Hf
+7tpT6sxDwDty4LHBUOZJG96N1234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader DOC-
EXAMPLE-BUCKET1.s3.us-west-1.amazonaws.com TLSV1.2 - -
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DOC-EXAMPLE-BUCKET1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be A1206F460EXAMPLE
 REST.GET.BUCKETPOLICY - "GET /DOC-EXAMPLE-BUCKET1?policy HTTP/1.1" 404
 NoSuchBucketPolicy 297 - 38 - "-" "S3Console/0.4" - BNaBsXZQQDbssi6xMBdBU2sLt
+Yf5kZDmeBUP35sFoKa3sLLeMC78iwEIWxs99CRUrbS4n11234= SigV4 ECDHE-RSA-AES128-GCM-SHA256
 AuthHeader DOC-EXAMPLE-BUCKET1.s3.us-west-1.amazonaws.com TLSV1.2 - Yes
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DOC-EXAMPLE-BUCKET1 [06/Feb/2019:00:01:00 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 7B4A0FABBEXAMPLE
 REST.GET.VERSIONING - "GET /DOC-EXAMPLE-BUCKET1?versioning HTTP/1.1" 200 -
 113 - 33 - "-" "S3Console/0.4" - Ke1bUcazaN1jWuUlPJaxF64cQVpUEhoZKEG/hmy/gijN/
I1DeWqDfFvnpybfEseEME/u7ME1234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader DOC-
EXAMPLE-BUCKET1.s3.us-west-1.amazonaws.com TLSV1.2 - -
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DOC-EXAMPLE-BUCKET1 [06/Feb/2019:00:01:57 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DD6CC733AEXAMPLE REST.PUT.OBJECT s3-dg.pdf "PUT /DOC-EXAMPLE-BUCKET1/
s3-dg.pdf HTTP/1.1" 200 - - 4406583 41754 28 "-" "S3Console/0.4" -
 10S62Zv81kBW7BB6SX4XJ48o6kpcl6LPwEoizZQQxJd5qDSCTLX0TgS37kYUBKQW3+bPdrg1234= SigV4
 ECDHE-RSA-AES128-SHA AuthHeader DOC-EXAMPLE-BUCKET1.s3.us-west-1.amazonaws.com TLSV1.2
 - Yes
```

> **Note**
>
> 任何欄位都可以設成 -，指出資料為未知或不可用，或欄位不適用於此要求。

主題

日誌記錄欄位

下列清單說明日誌記錄欄位。

儲存貯體擁有者

來源儲存貯體擁有者的正式使用者 ID。規範使用者 ID 是 ID 的另一種形式。 AWS 帳戶 如需正式使用者 ID 的詳細資訊，請參閱《AWS 一般參考》中的 AWS 帳戶 識別符。如需如何尋找帳戶正式使用者 ID 的資訊，請參閱尋找您的 AWS 帳戶的正式使用者 ID。

項目範例

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

儲存貯體

要求處理對象的儲存貯體名稱。如果系統收到格式錯誤的要求且無法判斷儲存貯體，則儲存貯體要求就不會出現在任何伺服器存取記錄中。

項目範例

```
DOC-EXAMPLE-BUCKET1
```

時間

收到請求的時間；這些日期和時間都使用國際標準時間 (UTC)。使用 `strftime()` 術語的格式如下：`[%d/%b/%Y:%H:%M:%S %z]`

項目範例

```
[06/Feb/2019:00:00:38 +0000]
```

遠端 IP

申請者清楚的 IP 地址。中間代理伺服器與防火牆可能會模糊提出請求之電腦的實際 IP 地址。

項目範例

```
192.0.2.3
```

要求者

要求者的正式使用者 ID，或未經驗證要求者的 -。如果請求者是 IAM 使用者，此欄位會傳回請求者的 IAM 使用者名稱以及 IAM 使用者所屬的名稱。 AWS 帳戶根使用者 此識別符與用於存取控制目的的識別符相同。

項目範例

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

如果請求者使用假定角色，此欄位會傳回假設的 IAM 角色。

項目範例

```
arn:aws:sts::123456789012:assumed-role/roleName/test-role
```

要求 ID

是由 Amazon S3 產生的字串，是可唯一識別每項請求。

項目範例

```
3E57427F33A59F07
```

操作

這裡列出的操作會宣告為 生命週期與記錄 的 SOAP.*operation*、REST.*HTTP_method*.*resource_type*、WEBSITE.*HTTP_method*.*resource_t* 或 BATCH.DELETE.OBJECT 或 S3.action.resource_type。

項目範例

```
REST.PUT.OBJECT
```

金鑰

請求的索引鍵 (物件名稱) 部分。

項目範例

```
/photos/2019/08/puppy.jpg
```

Request-URI

HTTP 請求訊息的 Request-URI 部分。

項目範例

```
"GET /DOC-EXAMPLE-BUCKET1/photos/2019/08/puppy.jpg?x-foo=bar HTTP/1.1"
```

HTTP 狀態

回應的數字 HTTP 狀態碼。

項目範例

```
200
```

錯誤代碼

Amazon S3 錯誤代碼 或 - (如果沒有發生錯誤)。

項目範例

```
NoSuchBucket
```

已傳送的位元組

已傳送的回應位元組數目 (排除 HTTP 通訊協定額外負荷) 或 - (若為零)。

項目範例

```
2662992
```

物件大小

所提及之物件的總大小。

項目範例

```
3462992
```

總時間

從伺服器角度計算的請求所經過的毫秒數。此值是從收到您要求的時間開始，計算到回應傳送出最後一組位元組的時間。由於網路延遲，從用戶端角度進行的測量可能較長。

項目範例

```
70
```

周轉時間

Amazon S3 處理您請求所花費的毫秒數。此值是從收到您要求的最後位元組的時間開始，計算到回應傳送出第一組位元組的時間。

項目範例

```
10
```

Referer

HTTP `Referer` 標頭的值，如果存在的話。提出要求時，HTTP 使用者代理程式 (例如：瀏覽器) 一般會將此標頭設為連結或內嵌頁面的 URL。

項目範例

```
"http://www.example.com/webservices"
```

User-Agent

HTTP `User-Agent` 標頭的值。

項目範例

```
"curl/7.15.1"
```

版本 Id

請求的版本 ID，或 - （如果操作未採用 versionId 參數）。

項目範例

```
3HL4kqtJvjVBH40Nrjfkd
```

主機 Id

x-amz-id-2 或 Amazon S3 延伸請求 ID。

項目範例

```
s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234=
```

簽章版本

簽章版本 (SigV2 或 SigV4)，用來驗證請求或未驗證請求的 -。

項目範例

```
SigV2
```

密碼套件

針對 HTTPS 請求或 HTTP 的 - 交涉的 Secure Sockets Layer (SSL) 密碼。

項目範例

```
ECDHE-RSA-AES128-GCM-SHA256
```

身分驗證類型

使用的請求身分驗證類型：AuthHeader 代表身分驗證標頭，QueryString 代表查詢字串 (預先簽章的 URL)，或 - 代表未身分驗證的請求。

項目範例

```
AuthHeader
```

主機標頭

用來連線到 Amazon S3 的端點。

項目範例

```
s3.us-west-2.amazonaws.com
```

某些早期區域支援舊版端點。您可能會在伺服器存取記錄或 AWS CloudTrail 記錄檔中看到這些端點。如需詳細資訊，請參閱 舊版端點。如需 Amazon S3 區域和端點的完整清單，請參閱《Amazon Web Services 一般參考》中的 Amazon S3 端點和配額。

TLS 版本控制

用戶端交涉的 Transport Layer Security (TLS) 版本。值為下列其中一個：TLSv1.1、TLSv1.2、TLSv1.3 或 - (如果未使用 TLS)。

項目範例

```
TLSv1.2
```

存取點 ARN

請求存取點的 Amazon Resource Name (ARN)。如果存取點 ARN 格式錯誤或未使用，該欄位將包含 -。如需存取點的詳細資訊，請參閱 使用存取點。如需 ARN 的詳細資訊，請參閱《AWS 參考指南》中的 Amazon Resource Name (ARN)。

項目範例

```
arn:aws:s3:us-east-1:123456789012:accesspoint/example-AP
```

aclRequired

字串，指出請求是否需要存取控制清單 (ACL) 進行授權。如果請求需要 ACL 進行授權，則字串為 Yes。如果不需要 ACL，則字串為 -。如需 ACL 的詳細資訊，請參閱「存取控制清單 (ACL) 概

觀」。如需使用 aclRequired 欄位停用 ACL 的詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

項目範例

```
Yes
```

複製操作的其他記錄

複製操作包括 GET 與 PUT。因此，執行複製操作時，我們會記錄兩筆記錄。上節說明與操作的 PUT 部分有關的欄位。下列清單說明記錄中與複製操作的 GET 部分有關的欄位。

儲存貯體擁有者

存放要複製物件之儲存貯體的正式使用者 ID。規範使用者 ID 是 ID 的另一種形式。 AWS 帳戶 如需正式使用者 ID 的詳細資訊，請參閱《AWS 一般參考》中的 AWS 帳戶 識別符。如需如何尋找帳戶正式使用者 ID 的資訊，請參閱尋找您的 AWS 帳戶的正式使用者 ID。

項目範例

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

儲存貯體

存放所複製物件的儲存貯體名稱。

項目範例

```
DOC-EXAMPLE-BUCKET1
```

時間

收到請求的時間；這些日期和時間都使用國際標準時間 (UTC)。使用 strftime() 術語的格式如下：[%d/%B/%Y:%H:%M:%S %z]

項目範例

```
[06/Feb/2019:00:00:38 +0000]
```

遠端 IP

申請者清楚的 IP 地址。中間代理伺服器與防火牆可能會模糊提出請求之電腦的實際 IP 地址。

項目範例

```
192.0.2.3
```

要求者

要求者的正式使用者 ID，或未經驗證要求者的 -。如果請求者是 IAM 使用者，此欄位將傳回請求者的 IAM 使用者名稱以及 IAM 使用者所屬的名稱。 AWS 帳戶根使用者 此識別符與用於存取控制目的的識別符相同。

項目範例

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

如果請求者使用假定角色，此欄位會傳回假設的 IAM 角色。

項目範例

```
arn:aws:sts::123456789012:assumed-role/roleName/test-role
```

要求 ID

是由 Amazon S3 產生的字串，是可唯一識別每項請求。

項目範例

```
3E57427F33A59F07
```

操作

這裡列出的操作會宣告為 SOAP.*operation*、REST.*HTTP_method*.*resource_type*、WEBSITE.*HTTP_method*.*resource_t* 或 BATCH.DELETE.OBJECT。

項目範例

```
REST.COPY.OBJECT_GET
```

金鑰

要複製之物件的金鑰 (物件名稱)；或 - (如果操作未採用金鑰參數)。

項目範例

```
/photos/2019/08/puppy.jpg
```

Request-URI

HTTP 請求訊息的 Request-URI 部分。

項目範例

```
"GET /DOC-EXAMPLE-BUCKET1/photos/2019/08/puppy.jpg?x-foo=bar"
```

HTTP 狀態

複製操作之 GET 部分的數字 HTTP 狀態碼。

項目範例

```
200
```

錯誤代碼

複製操作之 GET 部分的 Amazon S3 Amazon S3 錯誤代碼，或 - (如果未發生任何錯誤)。

項目範例

```
NoSuchBucket
```

已傳送的位元組

已傳送的回應位元組數目 (排除 HTTP 通訊協定額外負荷) 或 - (若為零)。

項目範例

```
2662992
```

物件大小

所提及之物件的總大小。

項目範例

```
3462992
```

總時間

從伺服器角度計算的請求所經過的毫秒數。此值是從收到您要求的時間開始，計算到回應傳送出最後一組位元組的時間。由於網路延遲，從用戶端角度進行的測量可能較長。

項目範例

```
70
```

周轉時間

Amazon S3 處理您請求所花費的毫秒數。此值是從收到您要求的最後位元組的時間開始，計算到回應傳送出第一組位元組的時間。

項目範例

```
10
```

Referer

HTTP `Referer` 標頭的值，如果存在的話。提出要求時，HTTP 使用者代理程式 (例如：瀏覽器) 一般會將此標頭設為連結或內嵌頁面的 URL。

項目範例

```
"http://www.example.com/webservices"
```

User-Agent

HTTP `User-Agent` 標頭的值。

項目範例

```
"curl/7.15.1"
```

版本 Id

要複製之物件的版本 ID，或 - (如果 x-amz-copy-source 標頭並未將 versionId 參數指定為複製來源的一部分)。

項目範例

```
3HL4kqtJvjVBH40Nrjfkd
```

主機 Id

x-amz-id-2 或 Amazon S3 延伸請求 ID。

項目範例

```
s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234=
```

簽章版本

簽章版本 (SigV2 或 SigV4)，用來驗證請求或未驗證請求的 -。

項目範例

```
SigV4
```

密碼套件

針對 HTTPS 請求或 HTTP 的 - 交涉的 Secure Sockets Layer (SSL) 密碼。

項目範例

```
ECDHE-RSA-AES128-GCM-SHA256
```

身分驗證類型

使用的請求身分驗證類型：AuthHeader 代表身分驗證標頭、QueryString 代表查詢字串 (預先簽章的 URL)，或 - 代表未驗證的請求。

項目範例

```
AuthHeader
```

主機標頭

用來連線到 Amazon S3 的端點。

項目範例

```
s3.us-west-2.amazonaws.com
```

某些早期區域支援舊版端點。您可能會在伺服器存取記錄或 AWS CloudTrail 記錄檔中看到這些端點。如需詳細資訊，請參閱 舊版端點。如需 Amazon S3 區域和端點的完整清單，請參閱《Amazon Web Services 一般參考》中的 Amazon S3 端點和配額。

TLS 版本控制

用戶端交涉的 Transport Layer Security (TLS) 版本。值為下列其中一個：TLSv1.1、TLSv1.2、TLSv1.3 或 - (如果未使用 TLS)。

項目範例

```
TLSv1.2
```

存取點 ARN

請求存取點的 Amazon Resource Name (ARN)。如果存取點 ARN 格式錯誤或未使用，該欄位將包含 -。如需存取點的詳細資訊，請參閱 使用存取點。如需 ARN 的詳細資訊，請參閱《AWS 參考指南》中的 Amazon Resource Name (ARN)。

項目範例

```
arn:aws:s3:us-east-1:123456789012:accesspoint/example-AP
```

aclRequired

字串，指出請求是否需要存取控制清單 (ACL) 進行授權。如果請求需要 ACL 進行授權，則字串為 Yes。如果不需要 ACL，則字串為 -。如需 ACL 的詳細資訊，請參閱「存取控制清單 (ACL) 概觀」。如需使用 aclRequired 欄位停用 ACL 的詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

項目範例

```
Yes
```

客戶存取日誌資訊

您可以包含要存放在請求存取日誌記錄中的自訂資訊。若要執行這項操作，請將自訂的查詢字串參數新增至請求的 URL。Amazon S3 忽略開頭為 x- 的查詢字串參數，但會將這些參數包含在請求的存取日誌日誌中，當成日誌記錄 Request-URI 欄位的一部分。

例如，GET 請求的 "s3.amazonaws.com/DOC-EXAMPLE-BUCKET1/photos/2019/08/puppy.jpg?x-user=johndoe" 運作方式與 "s3.amazonaws.com/DOC-EXAMPLE-BUCKET1/photos/2019/08/puppy.jpg" 請求相同，不同之處在於該 "x-user=johndoe" 字串包含在關聯日誌記錄的 Request-URI 欄位中。此功能僅有 REST 介面提供。

可擴展伺服器存取日誌格式的程式設計考量

有時候，我們可能要在每行結尾新增欄位，擴展存取日誌記錄的格式。因此，確定任何解析伺服器存取日誌的程式碼，可以處理其可能不理解的結尾欄位。

刪除 Amazon S3 日誌檔案

啟用了伺服器存取日誌記錄的 Amazon S3 儲存貯體，可以隨時累積許多伺服器日誌物件。建立這些存取日誌後經過一段特定的時間，您的應用程式可能會需要這些日誌，但在此期間結束後，可能希望刪除它們。您可以使用 Amazon S3 生命週期組態設定規則，讓 Amazon S3 自動將這些物件在期限到期時，將其排入刪除佇列。

您可以使用共用字首，為 S3 儲存貯體中的物件子集定義生命週期組態。若已在伺服器存取日誌組態中指定了字首，您可以設定生命週期組態規則，利用刪除含有該字首的日誌物件。

例如，假設日誌物件的字首為 logs/。在指定的期間之後，您可以設定生命週期組態規則，刪除儲存貯體中字首為 logs/ 的所有物件。

如需生命週期組態的詳細資訊，請參閱「管理儲存生命週期」。

如需伺服器存取記錄的一般資訊，請參閱 使用伺服器存取記錄記錄要求。

使用 Amazon S3 伺服器存取日誌來識別請求

您可以使用 Amazon S3 伺服器存取日誌來識別 Amazon S3 請求。

> **Note**
>
> - 若要識別 Amazon S3 請求，我們建議您使用 AWS CloudTrail 資料事件，而不是使用 Amazon S3 伺服器存取日誌。 CloudTrail 數據事件更容易設置並包含更多信息。如需詳細資訊，請參閱 使用識別 Amazon S3 請求 CloudTrail。
> - 根據您收到的存取要求數量，分析記錄可能需要比使用資 CloudTrail 料事件更多的資源或時間。

主題

使用 Amazon Athena 查詢請求的存取日誌

您可以使用 Amazon Athena 搭配 Amazon S3 存取日誌來識別 Amazon S3 請求。

Amazon S3 會將伺服器存取日誌當作物件存放於 S3 儲存貯體中。使用可以在 Amazon S3 中分析日誌的工具通常比較容易。Athena 支援分析 S3 物件，還可用來查詢 Amazon S3 存取日誌。

Example

以下範例示範如何在 Amazon Athena 中查詢 Amazon S3 伺服器存取日誌。請將下列範例中使用的 *user input placeholders* 取代為您自己的資訊。

> **Note**
>
> 若要在 Athena 查詢中指定 Amazon S3 位置，您必須提供交付日誌所在的儲存貯體的 S3 URI。此 URI 必須包含下列格式的儲存貯體名稱和字首：`s3://example-s3-bucket1-logs/prefix/`

1. 前往 https://console.aws.amazon.com/athena/ 開啟 Athena 主控台。
2. 在查詢編輯器中，執行類似如下的命令。將 *s3_access_logs_db* 取代為您要為資料庫指定的名稱。

```
CREATE DATABASE s3_access_logs_db
```

> **Note**
>
> 最佳做法是在與 S3 儲存貯體相同 AWS 區域 的位置建立資料庫。

3. 在查詢編輯器中，執行類似如下的命令，在您於步驟 2 建立的資料庫中建立資料表結構描述。將 *s3_access_logs_db.mybucket_logs* 取代為您要為資料表指定的名稱。STRING 及 BIGINT 資料類型值為存取日誌屬性。您可以在 Athena 中查詢這些屬性。在 LOCATION 的部分，輸入稍早記下的 S3 儲存貯體和字首路徑。

```
CREATE EXTERNAL TABLE `s3_access_logs_db.mybucket_logs`(
  `bucketowner` STRING,
  `bucket_name` STRING,
  `requestdatetime` STRING,
  `remoteip` STRING,
  `requester` STRING,
  `requestid` STRING,
  `operation` STRING,
  `key` STRING,
  `request_uri` STRING,
  `httpstatus` STRING,
  `errorcode` STRING,
  `bytessent` BIGINT,
  `objectsize` BIGINT,
  `totaltime` STRING,
  `turnaroundtime` STRING,
  `referrer` STRING,
  `useragent` STRING,
  `versionid` STRING,
  `hostid` STRING,
  `sigv` STRING,
  `ciphersuite` STRING,
  `authtype` STRING,
  `endpoint` STRING,
  `tlsversion` STRING,
  `accesspointarn` STRING,
  `aclrequired` STRING)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
```

```
  'input.regex'='([^ ]*) ([^ ]*) \\[(.*?)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\"|-) (-|[0-9]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 (\"[^\"]*\"|-) ([^ ]*)(?: ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*))?.*$')
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://DOC-EXAMPLE-BUCKET1-logs/prefix/'
```

4. 在導覽窗格的 Database (資料庫) 下，選擇您的資料庫。
5. 在 Tables (表格) 底下，選擇資料表名稱旁的 Preview table (預覽資料表)。

   在 Results (結果) 窗格中，應出現伺服器存取日誌的資料，例如 `bucketowner`、`bucket`、`requestdatetime` 等。這表示您成功建立 Athena 資料表。您現在可以查詢 Amazon S3 伺服器存取日誌。

Example — 顯示刪除物件的人與時間 (時間戳記、IP 地址和 IAM 使用者)

```
SELECT requestdatetime, remoteip, requester, key
FROM s3_access_logs_db.mybucket_logs
WHERE key = 'images/picture.jpg' AND operation like '%DELETE%';
```

Example — 顯示 IAM 使用者執行的所有操作

```
SELECT *
FROM s3_access_logs_db.mybucket_logs
WHERE requester='arn:aws:iam::123456789123:user/user_name';
```

Example — 顯示特定期間內針對某物件執行的所有操作

```
SELECT *
FROM s3_access_logs_db.mybucket_logs
WHERE Key='prefix/images/picture.jpg'
    AND parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
```

```
    BETWEEN parse_datetime('2017-02-18:07:00:00','yyyy-MM-dd:HH:mm:ss')
    AND parse_datetime('2017-02-18:08:00:00','yyyy-MM-dd:HH:mm:ss');
```

Example — 顯示特定時段有多少資料已傳輸至特定 IP 地址

```
SELECT coalesce(SUM(bytessent), 0) AS bytessenttotal
FROM s3_access_logs_db.mybucket_logs
WHERE remoteip='192.0.2.1'
AND parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2022-06-01','yyyy-MM-dd')
AND parse_datetime('2022-07-01','yyyy-MM-dd');
```

> **Note**
>
> 若要減少您保留日誌的時間，您可針對伺服器存取日誌儲存貯體建立 S3 生命週期組態。建立生命週期組態規則，以定期移除日誌檔。這麼做可降低 Athena 分析每個查詢時的資料量。如需詳細資訊，請參閱 在值區上設定生命週期組態。

使用 Amazon S3 存取日誌來識別簽章第 2 版請求

Amazon S3 將停止支援簽章第 2 版 (已淘汰)。之後，Amazon S3 將不再接受使用簽章第 2 版的請求，所有請求都必須使用簽章第 4 版來簽署。您可以使用 Amazon S3 存取日誌來識別簽章第 2 版請求。

> **Note**
>
> 若要識別簽章版本 2 請求，建議您使用 AWS CloudTrail 資料事件而非 Amazon S3 伺服器存取日誌。 CloudTrail 資料事件比伺服器存取記錄更容易設定，而且包含更多資訊。如需詳細資訊，請參閱 使用識別 Amazon S3 簽名版本 2 請求 CloudTrail。

Example — 顯示傳送簽章第 2 版流量的所有請求者

```
                SELECT requester, sigv, Count(sigv) as sigcount
                FROM s3_access_logs_db.mybucket_logs
```

```
                    GROUP BY requester, sigv;
```

使用 Amazon S3 存取日誌來識別物件存取請求

您可以在 Amazon S3 伺服器存取日誌上使用查詢，以識別 Amazon S3 物件存取請求，包括 GET、PUT 及 DELETE 等操作，並探索有關這些請求的詳細資訊。

以下 Amazon Athena 查詢範例示範如何從伺服器存取日誌取得 Amazon S3 的所有 PUT 物件請求。

Example — 顯示在特定期間內傳送 **PUT** 物件請求的所有申請者。

```
SELECT bucket_name, requester, remoteip, key, httpstatus, errorcode, requestdatetime
FROM s3_access_logs_db
WHERE operation='REST.PUT.OBJECT' AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

以下 Amazon Athena 查詢範例示範如何從伺服器存取日誌取得 Amazon S3 的所有 GET 物件請求。

Example — 顯示在特定期間內傳送 **GET** 物件請求的所有申請者。

```
SELECT bucket_name, requester, remoteip, key, httpstatus, errorcode, requestdatetime
FROM s3_access_logs_db
WHERE operation='REST.GET.OBJECT' AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

以下 Amazon Athena 查詢範例示範如何從伺服器存取日誌取得 S3 儲存貯體的所有匿名請求。

Example — 顯示在特定時段期間向儲存貯體提出請求的所有匿名申請者。

```
SELECT bucket_name, requester, remoteip, key, httpstatus, errorcode, requestdatetime
FROM s3_access_logs_db.mybucket_logs
WHERE requester IS NULL AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
```

```
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

下列 Amazon Athena 查詢示範如何識別對 S3 儲存貯體提出且需要存取控制清單 (ACL) 進行授權的所有請求。您可以使用此資訊，將這些 ACL 許可遷移至適當的儲存貯體政策，並停用 ACL。在建立了這些儲存貯體政策之後，您可以針對這些儲存貯體停用 ACL。如需停用 ACL 的詳細資訊，請參閱 停用 ACL 的先決條件。

Example — 識別需要 ACL 進行授權的所有請求

```
SELECT bucket_name, requester, key, operation, aclrequired, requestdatetime
FROM s3_access_logs_db
WHERE aclrequired = 'Yes' AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2022-05-10:00:00:00','yyyy-MM-dd:HH:mm:ss')
AND parse_datetime('2022-08-10:00:00:00','yyyy-MM-dd:HH:mm:ss')
```

> **Note**
>
> - 您可以視需要修改日期範圍以符合您的需求。
> - 對安全監控時而言，這些查詢範例也可能相當實用。您可以檢閱來自意外或未授權 IP 地址或申請者的 `PutObject` 或 `GetObject` 呼叫的結果，以及識別對您儲存貯體的任何匿名請求。
> - 此查詢只會擷取啟用日誌之後的資訊。
> - 如果您正在使用 AWS CloudTrail 記錄檔，請參閱使用識別 S3 物件的存取權 CloudTrail。

使用 Amazon 監控指標 CloudWatch

適用於 Amazon S3 的 Amazon CloudWatch 指標可協助您了解並改善使用 Amazon S3 之應用程式的效能。有幾種方法可以 CloudWatch 與 Amazon S3 搭配使用。

儲存貯體的每日儲存指標

使用來監控儲存貯體儲存 CloudWatch，該儲存體資料從 Amazon S3 收集並處理為可讀的每日指標。這些 Amazon S3 儲存指標每天會回報一次，並免費提供給所有客戶使用。

請求指標

監控 Amazon S3 請求，以快速找出並處理操作問題。這些指標會在一些處理延遲之後以 1 分鐘的間隔提供。這些 CloudWatch 指標的費率與 Amazon CloudWatch 自訂指標相同。如需有關 CloudWatch 定價的資訊，請參閱 Amazon CloudWatch 定價。若要了解如何選擇取得這些指標，請參閱 CloudWatch 度量組態。

啟用時，會回報所有物件操作的要求指標。這些 1 分鐘指標預設會在 Amazon S3 儲存貯體層級提供。您也可以為使用共用字首、物件標籤或存取點定義指標的篩選條件。

- 存取點 – 存取點為連接到儲存貯體的指定網路端點並可簡化 S3 中共用資料集的大規模資料存取管理。藉助存取點篩選條件，您可以深入了解存取點使用情況。如需存取點的詳細資訊，請參閱監控與記錄存取點。
- 字首 – 雖然 Amazon S3 資料模型是單層式結構，但您可以使用字首來推斷階層。字首類似於目錄名稱，可讓您在儲存貯體中對類似物件進行分組。S3 主控台採用資料夾的概念來支援字首。如果您依字首進行篩選，可知道指標組態含有具有相同字首的物件。如需字首的詳細資訊，請參閱使用字首整理物件。
- 標籤 – 標籤是您可新增至物件的鍵值名稱對。標籤可讓您輕鬆尋找與整理物件。您也可以將這些標籤當做指標組態的篩選條件，因此只有具有這些標籤的物件才會包含在指標組態中。如需物件標籤的詳細資訊，請參閱 使用標籤分類儲存空間。

若要將這些指標與特定商業應用程式、工作流程或內部組織調整一致，您可以針對共用字首、物件標籤或存取點進行篩選。

複寫指標

複寫指標 – 監控待複寫的 S3 API 操作總數、待複寫的物件總大小、目的地 AWS 區域的複寫時間上限，以及複寫失敗的操作總數。啟用 S3 複寫時間控制 (S3 RTC) 或 S3 複寫指標的複寫規則會發佈複寫指標。

如需詳細資訊，請參閱 使用複寫指標和 S3 事件通知監控進度 或 使用 S3 複寫時間控制 (S3 RTC) 來達到合規要求 。

Amazon S3 Storage Lens 指標

您可以將 S3 儲存鏡頭使用情況和活動指標發佈 CloudWatch 到 Amazon，以便在 CloudWatch 儀表板中建立操作狀態的統一檢視。S3 Storage Lens 指標可在 `AWS/S3/Storage-Lens` 命名空間使用。升級為進階指標和建議的 S3 儲存鏡頭儀表板可使用 CloudWatch 發佈選項。您可以在 S3 儲存鏡頭中為新的或現有的儀表板組態啟用 CloudWatch 發佈選項。

如需詳細資訊，請參閱 監控 CloudWatch 中的 S3 Storage Lens 指標。

所有 CloudWatch 統計資料都會保留 15 個月，以便您可以存取歷史資訊，並更好地瞭解 Web 應用程式或服務的執行情況。如需詳細資訊 CloudWatch，請參閱什麼是 Amazon CloudWatch？ 在 Amazon 用 CloudWatch 戶指南。您可能需要一些額外的設定來設定 CloudWatch 鬧鐘，具體取決於您的使用案例。例如，您可以使用指標數學運算式來建立警示。如需詳細資訊，請參閱 Amazon 使用者指南中的使用指 CloudWatch 標、使用指標數學運算、使 CloudWatch 用 Amazon CloudWatch 警示和根據指標數學運算式建立警示。 CloudWatch

最大努力 CloudWatch 指標交付

CloudWatch 指標是以最大的努力為基礎提供。大多數對具有請求指標的 Amazon S3 物件的請求都會導致資料點傳送到 CloudWatch。

不保證指標的完成程度與時間先後順序。特定要求的資料點回傳，回傳時附有的時間戳記可能會晚於實際處理要求時間。資料點可能會延遲一分鐘，然後才能透過使用 CloudWatch，否則可能根本無法傳遞。 CloudWatch 請求指標可讓您以近乎即時的方式瞭解儲存貯體流量的性質。並不表示完整考量所有要求。

由於遵循此功能的盡力本質，在帳單與成本管理儀表板提供的報告中，可能包含一或多個未出現在儲存貯體指標中的存取請求。

如需詳細資訊，請參閱下列主題。

主題

- 指標與維度
- 存取 CloudWatch 量度
- CloudWatch 度量組態

指標與維度

下表列出 Amazon S3 傳送給 Amazon CloudWatch 的儲存指標和維度。

最大努力 CloudWatch 指標交付

CloudWatch 指標是以最大的努力為基礎提供。大多數對具有請求指標的 Amazon S3 物件的請求都會導致資料點傳送到 CloudWatch。

不保證指標的完成程度與時間先後順序。特定要求的資料點回傳，回傳時附有的時間戳記可能會晚於實際處理要求時間。資料點可能會延遲一分鐘，然後才能透過使用 CloudWatch，否則可能根本無法傳

遞。 CloudWatch 請求指標可讓您以近乎即時的方式瞭解儲存貯體流量的性質。並不表示完整考量所有要求。

由於遵循此功能的盡力本質，在帳單與成本管理儀表板提供的報告中，可能包含一或多個未出現在儲存貯體指標中的存取請求。

主題

- 適用於儲存貯體的 Amazon S3 每日儲存指標 CloudWatch
- Amazon S3 請求指標 CloudWatch
- S3 複寫指標 CloudWatch
- S3 儲存鏡頭指標 CloudWatch
- S3 物件 Lambda 請求指標 CloudWatch
- Outposts 上的 Amazon S3 指標 CloudWatch
- Amazon S3 維度 CloudWatch
- S3 複寫維度 CloudWatch
- S3 儲存鏡頭尺寸 CloudWatch
- S3 物件 Lambda 請求維度 CloudWatch

適用於儲存貯體的 Amazon S3 每日儲存指標 CloudWatch

AWS/S3 命名空間包含下列每日儲存貯體儲存體指標。

指標	描述
BucketSizeBytes	存放在下列儲存類別中儲存貯體的資料量 (以位元組為單位)： • S3 標準 (STANDARD) • S3 智慧型分層服務 (INTELLIGENT_TIERING) • S3 S3 標準 - 不常存取 (STANDARD_IA) • S3 單區域-不常存取 () ONEZONE_IA • 低冗餘儲存 (RRS) (REDUCED_REDUNDANCY) • S3 Glacier Instant Retrieval (GLACIER_IR) • S3 Glacier Deep Archive (DEEP_ARCHIVE) • S3 Glacier Flexible Retrieval (GLACIER)

指標	描述
	• S3 Express One Zone (EXPRESS_ONEZONE) 此值的計算方式是將值區 (包括目前和非目前物件) 中的所有物件和中繼資料 (例如儲存貯體名稱) 的大小加總，包括所有不完整分段上傳至值區的所有零件大小。 有效的儲存類型篩選條件：StandardStorage 、IntelligentTieringFAStorage 、IntelligentTieringIAStorage 、IntelligentTieringAAStorage 、IntelligentTieringAIAStorage 、IntelligentTieringDAAStorage 、StandardIAStorage 、StandardIASizeOverhead 、StandardIAObjectOverhead 、OneZoneIAStorage 、OneZoneIASizeOverhead 、ReducedRedundancyStorage 、GlacierInstantRetrievalSizeOverhead 、GlacierInstantRetrievalStorage 、GlacierStorage 、GlacierStagingStorage 、GlacierObjectOverhead 、GlacierS3ObjectOverhead 、DeepArchiveStorage 、DeepArchiveObjectOverhead 、DeepArchiveS3ObjectOverhead 、DeepArchiveStagingStorage 和 ExpressOneZone (請參閱 StorageType 維度) 單位：位元組 有效的統計資訊：平均

指標	描述
NumberOfObjects	針對所有儲存類別在一般用途儲存貯體中存放的物件總數 此值是計算儲存貯體中所有物件的數量 (包含目前及非目前物件)、刪除標記以及所有分段上傳到儲存貯體的所有不完整部分的總數而計算得出。對於具有 S3 Express One Zone 儲存類別中物件的目錄儲存貯體，此值的計算方式是計算值區中的所有物件，但不包括不完整的多次上傳到值區。 有效的儲存體類型篩選條件：AllStorageTypes （請參閱 StorageType 維度) 單位：計數 有效的統計資訊：平均

Amazon S3 請求指標 CloudWatch

AWS/S3 命名空間包含下列要求指標。這些量度包括不可計費的請求 (在來自GET要求CopyObject和複寫的情況下)。

> **Note**
>
> 目錄儲存貯體 CloudWatch 不支援中的 Amazon S3 請求指標。

指標	描述
AllRequests	對 Amazon S3 儲存貯體提出的 HTTP 請求總數，不論類型為何。如果您要搭配使用指標組態與篩選條件，則此指標只會傳回符合篩選條件需求的 HTTP 請求。 單位：計數 有效的統計資訊：總和
GetRequests	對 Amazon S3 儲存貯體中的物件提出的 HTTPGET 請求數目。這不包含列出操作。此測量結果會CopyObject 根據每個要求的來源遞增。 單位：計數

指標	描述
	有效的統計資訊：總和 Note 此量度不包含分頁清單導向要求 (例如ListMultipartUploadsListPartsListObjectVersions、、及其他)。
PutRequests	對 Amazon S3 儲存貯體中的物件提出的 HTTPPUT 請求數目。此測量結果會CopyObject 根據每個要求的目的地遞增。 單位：計數 有效的統計資訊：總和
DeleteRequests	對 Amazon S3 儲存貯體中的物件提出的 HTTPDELETE 請求數目。此量度也包含DeleteObjects要求。此指標顯示所提出的請求數目，而非已刪除的物件數目。 單位：計數 有效的統計資訊：總和
HeadRequests	對 Amazon S3 儲存貯體提出的 HTTP HEAD 請求數目。 單位：計數 有效的統計資訊：總和
PostRequests	對 Amazon S3 儲存貯體提出的 HTTP POST 請求數目。 單位：計數 有效的統計資訊：總和 Note DeleteObjects和SelectObjectContent要求不包含在此量度中。

指標	描述
SelectRequests	針對 Amazon S3 儲存貯體中的物件發出的 Amazon S3 SelectObjectContent請求數目。 單位：計數 有效的統計資訊：總和
SelectBytesScanned	Amazon S3 儲存貯體中使用 Amazon S3 SelectObjectContent請求掃描的資料位元組數。 單位：位元組 有效的統計資訊：平均 (每個要求的位元組數)、總和 (每個期間的位元組數)、範例計數、最小值、最大值 (相同於 p100)，任何介於 p0.0 與 p99.9 的百分位數
SelectBytesReturned	Amazon S3 儲存貯體中使用 Amazon S3 SelectObjectContent請求傳回的資料位元組數。 單位：位元組 有效的統計資訊：平均 (每個要求的位元組數)、總和 (每個期間的位元組數)、範例計數、最小值、最大值 (相同於 p100)，任何介於 p0.0 與 p99.9 的百分位數
ListRequests	列出儲存貯體內容的 HTTP 請求的數目。 單位：計數 有效的統計資訊：總和
BytesDownloaded	因為對 Amazon S3 儲存貯體提出的請求而下載的位元組數，其中回應包含內文。 單位：位元組 有效的統計資訊：平均 (每個要求的位元組數)、總和 (每個期間的位元組數)、範例計數、最小值、最大值 (相同於 p100)，任何介於 p0.0 與 p99.9 的百分位數

指標	描述
BytesUploaded	針對 Amazon S3 儲存貯體提出之請求所上傳的位元組數，其中請求包含內文。 單位：位元組 有效的統計資訊：平均 (每個要求的位元組數)、總和 (每個期間的位元組數)、範例計數、最小值、最大值 (相同於 p100)，任何介於 p0.0 與 p99.9 的百分位數
4xxErrors	對具有 0 或 1 的 Amazon S3 儲存貯體發出 HTTP 4 xx 用戶端錯誤狀態碼請求的數目。平均統計資訊會顯示錯誤率，而總和統計資訊會顯示該類型錯誤在每個期間的數目。 單位：計數 有效的統計資訊：平均 (每個要求的報告)、總和 (每個期間的報告)、下限、上限、範例計數
5xxErrors	對具有 0 或 1 的 Amazon S3 儲存貯體發出 HTTP 5 xx 伺服器錯誤狀態碼請求的數目。平均統計資訊會顯示錯誤率，而總和統計資訊會顯示該類型錯誤在每個期間的數目。 單位：計數 有效的統計資訊：平均 (每個要求的報告)、總和 (每個期間的報告)、下限、上限、範例計數
FirstByteLatency	從 Amazon S3 儲存貯體收到完整請求起，到回應開始傳回為止，每個請求所花的時間。 單位：毫秒 有效的統計資訊：平均、總和、下限、上限 (與 p100 相同)、範例計數、任何介於 p0.0 與 p100 的百分位數

指標	描述
TotalRequestLatency	從收到第一個位元組起，到最後一個位元組傳送至 Amazon S3 儲存貯體為止，每個請求所經過的時間。此指標包含接收要求內文與傳送回應內文 (未包含在 FirstByteLatency 中) 所需的時間。 單位：毫秒 有效的統計資訊：平均、總和、下限、上限 (與 p100 相同)、範例計數、任何介於 p0.0 與 p100 的百分位數

S3 複寫指標 CloudWatch

您可以透過追蹤擱置的位元組、擱置中的作業和複寫延遲，以監控 S3 複寫指標的複寫進度。如需詳細資訊，請參閱監控複寫指標的進度。

> **Note**
>
> 您可以在 Amazon 中為複寫指標啟用警示 CloudWatch。當您為複寫指標設定警示時，請將 Missing data treatment (遺失資料處理) 欄位設定為 Treat missing data as ignore (maintain the alarm state) (將遺失的資料視為忽略 (維護警示狀態))。

指標	描述
ReplicationLatency	指定複製規則中，複製目的地 AWS 區域 位 AWS 區域 於來源之後的最大秒數。 單位：秒 有效統計資訊：Max
BytesPendingReplication	指定複寫規則擱置中複寫的物件位元組總數。 單位：位元組 有效統計資訊：Max

指標	描述
`OperationsPendingReplication`	指定複寫規則的擱置中複寫操作數目。 單位：計數 有效統計資訊：Max
`OperationsFailedReplication`	指定複寫規則的複寫失敗操作數目。 單位：計數 有效統計資料：總和 (失敗操作總數)、平均 (失敗率)、範例計數 (複寫操作總數)

S3 儲存鏡頭指標 CloudWatch

您可以將 S3 儲存鏡頭用量和活動指標發佈 CloudWatch 到 Amazon，以便在CloudWatch儀表板中建立操作狀態的統一檢視。S3 儲存鏡頭指標會發佈至中的`AWS/S3/Storage-Lens`命名空間 CloudWatch。已升級為進階指標和建議的 S3 Storage Lens 儀表板可使用 CloudWatch 發佈選項。

如需發佈到的 S3 儲存鏡頭指標清單 CloudWatch，請參閱Amazon S3 Storage Lens 指標詞彙表。如需維度的完整清單，請參閱 維度。

S3 物件 Lambda 請求指標 CloudWatch

S3 Object Lambda 包含下列請求指標。

指標	描述
`AllRequests`	使用 Object Lambda 存取點對 Amazon S3 儲存貯體提出的 HTTP 請求總數。 單位：計數 有效的統計資訊：總和
`GetRequests`	使用 Object Lambda 存取點為物件提出的 HTTP GET 請求數目。此指標不包含列出操作。

指標	描述
	單位：計數 有效的統計資訊：總和
BytesUploaded	使用 Object Lambda 存取點 (其中請求包含內文) 上傳至 Amazon S3 儲存貯體的位元組數。 單位：位元組 有效的統計資訊：平均 (每個要求的位元組數)、總和 (每個期間的位元組數)、範例計數、最小值、最大值 (相同於 p100)，任何介於 p0.0 與 p99.9 的百分位數
PostRequests	使用 Object Lambda 存取點對 Amazon S3 儲存貯體提出的 HTTP POST 請求數目。 單位：計數 有效的統計資訊：總和
PutRequests	使用 Object Lambda 存取點為 Amazon S3 儲存貯體中物件提出的 HTTP PUT 請求數目。 單位：計數 有效的統計資訊：總和
DeleteRequests	使用 Object Lambda 存取點為 Amazon S3 儲存貯體中物件提出的 HTTP DELETE 請求數目。此量度包括DeleteObjects要求。此指標顯示所提出的請求數目，而非已刪除的物件數目。 單位：計數 有效的統計資訊：總和

指標	描述
BytesDownloaded	因為使用 Object Lambda 存取點 (其中回應包含內文) 對 Amazon S3 儲存貯體提出的請求而下載的位元組數。 單位：位元組 有效的統計資訊：平均 (每個要求的位元組數)、總和 (每個期間的位元組數)、範例計數、最小值、最大值 (相同於 p100)，任何介於 p0.0 與 p99.9 的百分位數
FirstByteLatency	從 Amazon S3 儲存貯體透過 Object Lambda 存取點收到完整請求起，到回應開始傳回為止，每個請求所花的時間。此指標取決於 AWS Lambda 函數在該函數將位元組傳回至 Object Lambda 存取點之前轉換物件的執行時間。 單位：毫秒 有效的統計資訊：平均、總和、下限、上限 (與 p100 相同)、範例計數、任何介於 p0.0 與 p100 的百分位數
TotalRequestLatency	從收到第一個位元組起，到最後一個位元組傳送至 Object Lambda 存取點為止，每個請求所經過的時間。此指標包含接收要求內文與傳送回應內文 (未包含在 FirstByteLatency 中) 所需的時間。 單位：毫秒 有效的統計資訊：平均、總和、下限、上限 (與 p100 相同)、範例計數、任何介於 p0.0 與 p100 的百分位數
HeadRequests	使用 Object Lambda 存取點對 Amazon S3 儲存貯體提出的 HTTP HEAD 請求數目。 單位：計數 有效的統計資訊：總和

指標	描述
ListRequests	列出 Amazon S3 儲存貯體內容的 HTTP GET 請求數目。此指標同時包含 ListObjects 與 ListObjectsV2 操作。 單位：計數 有效的統計資訊：總和
4xxErrors	透過使用值為 0 或 1 的物件 Lambda 存取點，向 Amazon S3 儲存貯體發出 HTTP 4 xx 用戶端錯誤狀態碼請求的數目。平均統計資訊會顯示錯誤率，而總和統計資訊會顯示該類型錯誤在每個期間的數目。 單位：計數 有效的統計資訊：平均 (每個要求的報告)、總和 (每個期間的報告)、下限、上限、範例計數
5xxErrors	透過使用值為 0 或 1 的物件 Lambda 存取點向 Amazon S3 儲存貯體發出的 HTTP 5 xx 伺服器錯誤狀態碼請求數目。平均統計資訊會顯示錯誤率，而總和統計資訊會顯示該類型錯誤在每個期間的數目。 單位：計數 有效的統計資訊：平均 (每個要求的報告)、總和 (每個期間的報告)、下限、上限、範例計數
ProxiedRequests	傳回標準 Amazon S3 API 回應之 Object Lambda 存取點的 HTTP 要求數目。(這類請求未設定 Lambda 函數。) 單位：計數 有效的統計資訊：總和
InvokedLambda	已在其中叫用 Lambda 函數之 S3 物件的 HTTP 請求數目。 單位：計數 有效的統計資訊：總和

指標	描述
LambdaResponseRequests	Lambda 函數提出的 WriteGetObjectResponse 請求數目。此指標僅適用於 GetObject 請求。
LambdaResponse4xx	WriteGetObjectResponse 從 Lambda 函數呼叫時，發生的 HTTP 4xx 用戶端錯誤數目。此指標會提供與 4xxErrors 相同的資訊，但僅適用於 WriteGetObjectResponse 呼叫。
LambdaResponse5xx	WriteGetObjectResponse 從 Lambda 函數呼叫時，發生的 HTTP 5xx 伺服器錯誤數目。此指標會提供與 5xxErrors 相同的資訊，但僅適用於 WriteGetObjectResponse 呼叫。

Outposts 上的 Amazon S3 指標 CloudWatch

如需 Outposts 儲存貯體中 CloudWatch 用於 S3 的指標清單，請參閱CloudWatch 度量。

Amazon S3 維度 CloudWatch

下列維度用來篩選 Amazon S3 指標。

維度	描述
BucketName	此維度會篩選僅針對已識別儲存貯體所要求的資料。
StorageType	此維度會依下列儲存體類型篩選儲存貯體中已存放的資料： • StandardStorage – 用於 STANDARD 儲存類別中物件的位元組數。 • IntelligentTieringAAStorage – 用於 INTELLIGENT_TIERING 儲存類別之 Archive 存取層中物件的位元組數。 • IntelligentTieringAIAStorage – 用於 INTELLIGENT_TIERING 儲存類別之 Archive 即時存取層中物件的位元組數。

維度	描述
	• IntelligentTieringDAAStorage – 用於 INTELLIGENT_TIERING 儲存類別之 Deep Archive 存取層中物件的位元組數。 • IntelligentTieringFAStorage – 用於 INTELLIGENT_TIERING 儲存類別之經常存取層中物件的位元組數。 • IntelligentTieringIAStorage – 用於 INTELLIGENT_TIERING 儲存類別之不常存取層中物件的位元組數。 • StandardIAStorage — S3 標準-不常存取 (STANDARD_IA) 儲存類別中用於物件的位元組數。 • StandardIASizeOverhead – 用於 STANDARD_IA 儲存類別中小於 128 KB 之物件的位元組數。 • IntAAObjectOverhead – 針對 INTELLIGENT_TIERING 儲存類別中 Archive 存取層的每個物件，S3 Glacier 新增 32 KB，存放索引和相關的中繼資料。為了能識別及還原您的物件，將需要這項額外的資料。系統會以 S3 Glacier Flexible Retrieval 費率向您收取此額外儲存的費用。 • IntAAS3ObjectOverhead – 針對 INTELLIGENT_TIERING 儲存類別中 Archive 存取層的每個物件，Amazon S3 新增 8 KB，存放物件的名稱和其他中繼資料。將就這項額外的儲存體向您收取 S3 Standard 費率。 • IntDAAObjectOverhead – 針對 INTELLIGENT_TIERING 儲存類別中 Deep Archive 存取層的每個物件，S3 Glacier 新增 32 KB，存放索引和相關的中繼資料。為了能識別及還原您的物件，將需要這項額外的資料。將向您收取此額外儲存的 S3 Glacier Deep Archive 儲存費用。 • IntDAAS3ObjectOverhead – 針對 INTELLIGENT_TIERING 儲存類別中 Deep Archive 存取層的每個物件，Amazon S3 新增 8 KB，存放索引和相關的中繼資料。為了能識別及還原您的物件，將需要這項額外的資料。將就這項額外的儲存體向您收取 S3 Standard 費率。 • OneZoneIAStorage – 用於 S3 單區域 – 不常存取 (ONEZONE_IA) 儲存類別中物件的位元組數。

維度	描述
	• OneZoneIASizeOverhead – 用於 ONEZONE_IA 儲存類別中小於 128 KB 之物件的位元組數。 • ReducedRedundancyStorage - 低冗餘儲存體 (RRS) 類別中的物件所使用的位元組數。 • GlacierInstantRetrievalSizeOverhead – 用於 S3 Glacier Instant Retrieval 儲存類別中小於 128 KB 之物件的位元組數。 • GlacierInstantRetrievalStorage – S3 Glacier Instant Retrieval 儲存類別中的物件所使用的位元組數。 • GlacierStorage – S3 Glacier Flexible Retrieval 儲存類別中的物件所使用的位元組數。 • GlacierStagingStorage – 在 S3 Glacier Flexible Retrieval 儲存類別中的物件上完成 CompleteMultipartUpload 請求之前，用於分段物件的數個片段的位元組數。 • GlacierObjectOverhead - S3 Glacier 會為每個封存物件新增 32 KB 的儲存體，供索引及相關中繼資料使用。為了能識別及還原您的物件，將需要這項額外的資料。系統會以 S3 Glacier Flexible Retrieval 費率向您收取此額外儲存的費用。 • GlacierS3ObjectOverhead – Amazon S3 會為每個封存至 S3 Glacier Flexible Retrieval 的物件，使用 8 KB 的儲存空間，供物件的名稱及其他中繼資料使用。將就這項額外的儲存體向您收取 S3 Standard 費率。 • DeepArchiveStorage - S3 Glacier Deep Archive 儲存類別中的物件所使用的位元組數。 • DeepArchiveObjectOverhead - S3 Glacier 會為每個封存物件新增 32 KB 的儲存體，供索引及相關中繼資料使用。為了能識別及還原您的物件，將需要這項額外的資料。將向您收取此額外儲存體的 S3 Glacier Deep Archive 費用。 • DeepArchiveS3ObjectOverhead - Amazon S3 會為每個封存至 S3 Glacier Deep Archive 的物件使用 8 KB 的儲存體，供物件的名稱及其他中繼資料使用。將就這項額外的儲存體向您收取 S3 Standard 費率。

維度	描述
	• DeepArchiveStagingStorage – 在 S3 Glacier Deep Archive 儲存類別中的物件上完成 CompleteMultipartUpload 請求之前，用於分段物件的數個片段的位元組數。 • ExpressOneZone ：用於 S3 Express One Zone 儲存類別中物件的位元組數。
FilterId	此維度會篩選針對儲存貯體上請求指標所指定的指標組態。建立指標組態時，您可以指定篩選 ID (例如，字首、標籤或存取點)。如需詳細資訊，請參閱建立指標組態。

S3 複寫維度 CloudWatch

下列維度用於篩選 S3 複寫指標。

維度	描述
SourceBucket	值區物件的名稱會從中複製。
DestinationBucket	值區物件的名稱會複製到。
RuleId	觸發此複製測量結果更新之規則的唯一識別碼。

S3 儲存鏡頭尺寸 CloudWatch

如需用來篩選 S3 儲存鏡頭指標的維度清單 CloudWatch，請參閱維度。

S3 物件 Lambda 請求維度 CloudWatch

下列維度用來篩選 Object Lambda 存取點中的資料。

維度	描述
AccessPointName	將要提出請求的存取點名稱。

維度	描述
DataSourceARN	Object Lambda 存取點將要在其中擷取資料的來源。如果請求調用 Lambda 函數，則該函數將引用 Lambda Amazon 資源名稱（ARN）。否則，稱為存取點 ARN。

存取 CloudWatch 量度

您可以使用下列程序來檢視 Amazon S3 的儲存指標。若要取得涉及的 Amazon S3 指標，您必須設定開始與結束時間戳記。針對任何給定 24 小時期間的指標，將期間設定為 86400 秒 (一天的秒數)。也請記得設定 BucketName 與 StorageType 維度。

使用 AWS CLI

例如，如果您想要使用 AWS CLI 來取得特定值區大小的平均值 (以位元組為單位)，您可以使用下列命令：

```
aws cloudwatch get-metric-statistics --metric-name BucketSizeBytes --namespace AWS/S3
 --start-time 2016-10-19T00:00:00Z --end-time 2016-10-20T00:00:00Z --statistics Average
 --unit Bytes --region us-west-2 --dimensions Name=BucketName,Value=DOC-EXAMPLE-BUCKET
 Name=StorageType,Value=StandardStorage --period 86400 --output json
```

此範例會產生下列輸出：

```
{
    "Datapoints": [
        {
            "Timestamp": "2016-10-19T00:00:00Z",
            "Average": 1025328.0,
            "Unit": "Bytes"
        }
    ],
    "Label": "BucketSizeBytes"
}
```

使用 S3 主控台

使用 Amazon CloudWatch 主控台檢視指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

2. 在左側導覽窗格中，選擇 Metrics (指標)。
3. 選擇 S3 (S3) 命名空間。
4. (選用) 若要檢視指標，請在搜尋方塊中輸入指標名稱。
5. (選擇性) 若要依StorageType維度篩選，請在搜尋方塊中輸入儲存類別的名稱。

若要檢視為您 AWS 帳戶 儲存的有效量度清單，請使用 AWS CLI

- 在命令提示中，使用下列命令。

```
aws cloudwatch list-metrics --namespace "AWS/S3"
```

有關存取 CloudWatch 儀表板所需許可的詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的 Amazon CloudWatch 儀表板許可。

CloudWatch 度量組態

使用 Amazon S3 的 Amazon CloudWatch 請求指標，您可以接收 1 分鐘 CloudWatch 指標、設定 CloudWatch 警示和存取 CloudWatch 儀表板以檢視 Amazon S3 儲存的 near-real-time 操作和效能。針對與雲端儲存體相依的應用程式，這些指標可讓您快速找出並處理操作問題。啟用時，這些 1 分鐘指標預設會在 Amazon S3 儲存貯體層級提供。

如果您想要取得值區中物件的要 CloudWatch 求指標，您必須建立值區的指標組態。如需詳細資訊，請參閱 為值區中的所有物件建立 CloudWatch 指標設定。

您也可以使用共用字首、物件標籤或存取點來為收集的指標定義篩選條件。該定義篩選條件的方法可讓您將指標篩選條件與特定商業應用程式、工作流程或內部組織調整一致。如需詳細資訊，請參閱 建立依字首、物件標籤或存取點篩選的指標組態。如需有關可用 CloudWatch度量以及儲存體和要求度量之間差異的詳細資訊，請參閱使用 Amazon 監控指標 CloudWatch。

使用指標組態時，請注意下列各項：

- 每個儲存貯體最多可以有 1,000 個指標組態。
- 您可以使用篩選條件來選擇儲存貯體中要包含在指標組態中的物件。您可以針對共用字首、物件標籤或存取點進行篩選，從而將指標篩選條件與特定商業應用程式、工作流程或內部組織調整一致。若要要求整個儲存貯體的指標，請建立沒有篩選條件的指標組態。
- 只需要指標組態，就能啟用要求指標。儲存體層級每日儲存體指標一律會予以開啟，並免費提供。目前無法取得已篩選物件子集的每日儲存體指標。

- 每個指標組態都會啟用一組完整可用的請求指標。如果您的儲存貯體或篩選條件有該類型的請求，則只會回報操作特定指標 (例如 `PostRequests`)。
- 系統會回報物件層級操作的要求指標。對於列出儲存貯體內容的操作，例如GET 儲存貯體 (列出物件)、GET 儲存貯體物件版本及列出分段上傳，也會回報請求指標；但對於儲存貯體的其他操作，則不會回報。
- 請求指標可支援透過字首、物件標籤或存取點的篩選，但儲存指標不支援此功能。

最大努力 CloudWatch 指標交付

CloudWatch 指標是以最大的努力為基礎提供。大多數對具有請求指標的 Amazon S3 物件的請求都會導致資料點傳送到 CloudWatch。

不保證指標的完成程度與時間先後順序。特定要求的資料點回傳，回傳時附有的時間戳記可能會晚於實際處理要求時間。資料點可能會延遲一分鐘，然後才能透過使用 CloudWatch，否則可能根本無法傳遞。 CloudWatch 請求指標可讓您以近乎即時的方式瞭解儲存貯體流量的性質。並不表示完整考量所有要求。

由於遵循此功能的盡力本質，在帳單與成本管理儀表板提供的報告中，可能包含一或多個未出現在儲存貯體指標中的存取請求。

如需在 Amazon S3 中使用 CloudWatch 指標的詳細資訊，請參閱下列主題。

主題

- 為值區中的所有物件建立 CloudWatch 指標設定
- 建立依字首、物件標籤或存取點篩選的指標組態
- 刪除指標篩選條件

為值區中的所有物件建立 CloudWatch 指標設定

設定要求指標時，您可以為值區中的所有物件建立 CloudWatch 指標設定，也可以依前置詞、物件標籤或存取點進行篩選。本主題中的程序說明如何為儲存貯體中的所有物件建立組態。若要建立依物件標籤、字首或存取點篩選的組態，請參閱 建立依字首、物件標籤或存取點篩選的指標組態。

Amazon Amazon S3 有三種類型的 Amazon CloudWatch 指標：儲存指標、請求指標和複寫指標。儲存體指標會每天回報一次，並免費提供給所有客戶。請求指標會在一些處理延遲之後，以 1 分鐘的間隔提供。請求指標會以標準費 CloudWatch 率計費。您必須在主控台中設定或透過 Amazon S3 API 來選擇使用請求指標。S3 複寫指標提供了複寫組態中複寫規則的詳細指標。使用複寫指標，您可以追蹤擱置的位元組、擱置中的作業、複寫失敗的作業以及複寫延遲來監視 minute-by-minute 進度。

如需 Amazon S3 CloudWatch 指標的詳細資訊，請參閱使用 Amazon 監控指標 CloudWatch。

您可以使用 Amazon S3 主控台、 AWS Command Line Interface (AWS CLI) 或 Amazon S3 REST API，將指標組態新增至儲存貯體。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在儲存貯體清單中，選擇包含您要請求指標之物件的儲存貯體名稱。
3. 選擇 Metrics (指標) 標籤。
4. 在 Bucket metrics (儲存貯體指標) 下，選擇 View additional charts (檢視其他圖表)。
5. 選擇 Request metrics (請求指標) 標籤。
6. 選擇 Create filter (建立篩選條件)。
7. 在 Filter name (篩選條件名稱) 方塊中，輸入篩選條件的名稱。

   名稱僅可包含字母、數字、句點、破折號和底線。建議您使用 `EntireBucket` 來為適用於所有物件的篩選條件命名。
8. 在 Filter scope (篩選條件範圍) 底下，選擇 This filter applies to all objects in the bucket (此篩選條件適用於儲存貯體中的所有物件)。

   您也可以定義篩選條件，僅收集與報告儲存貯體中物件子集的指標。如需詳細資訊，請參閱 建立依字首、物件標籤或存取點篩選的指標組態。
9. 選擇 Save changes (儲存變更)。
10. 在 Request metrics (請求指標) 標籤的 Filters (篩選條件) 底下，選擇您剛才建立的篩選條件。

    大約 15 分鐘後， CloudWatch 開始追蹤這些要求量度。Request metrics (請求指標) 標籤中會顯示這些指標，您可以在 Amazon S3 或 CloudWatch 主控台上查看指標的圖形。請求指標會以標準費 CloudWatch 率計費。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

使用 REST API

您也可以使用 Amazon S3 REST API，以程式設計方式新增指標組態。如需新增和使用指標組態的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列主題：

- PUT 儲存貯體指標組態
- GET 儲存貯體指標組態

- List 儲存貯體指標組態
- DELETE 儲存貯體指標組態

使用 AWS CLI

1. 安裝並設定 AWS CLI. 如需說明，請參閱《AWS Command Line Interface 使用者指南》中的安裝、更新和解除安裝 AWS CLI。
2. 開啟終端機。
3. 執行下列命令來新增指標組態：

```
aws s3api put-bucket-metrics-configuration --endpoint https://s3.us-
west-2.amazonaws.com --bucket bucket-name --id metrics-config-id --metrics-
configuration '{"Id":"metrics-config-id"}'
```

建立依字首、物件標籤或存取點篩選的指標組態

Amazon Amazon S3 有三種類型的 Amazon CloudWatch 指標：儲存指標、請求指標和複寫指標。儲存體指標會每天回報一次，並免費提供給所有客戶。請求指標會在一些處理延遲之後，以 1 分鐘的間隔提供。請求指標會以標準費 CloudWatch 率計費。您必須在主控台中設定或透過 Amazon S3 API 來選擇使用請求指標。S3 複寫指標提供了複寫組態中複寫規則的詳細指標。使用複寫指標，您可以追蹤擱置的位元組、擱置中的作業、複寫失敗的作業以及複寫延遲來監視 minute-by-minute 進度。

如需 Amazon S3 CloudWatch 指標的詳細資訊，請參閱使用 Amazon 監控指標 CloudWatch。

設定 CloudWatch 指標時，您可以為值區中的所有物件建立篩選器，也可以將組態篩選為單一值區內的相關物件群組。您也可以根據下列一或多個篩選條件類型，篩選儲存貯體中要包含在指標組態中的物件：

- 物件金鑰名稱字首 – 雖然 Amazon S3 資料模型是單層式結構，但您可以使用字首來推斷階層。Amazon S3 主控台採用資料夾的概念來支援這些前綴。如果您依字首進行篩選，可知道指標組態含有具有相同字首的物件。如需字首的詳細資訊，請參閱 使用字首整理物件。
- 標籤 – 您可以將標籤 (鍵/值名稱對) 新增至物件。標籤可讓您輕鬆尋找與整理物件。您也可以將這些標籤當做指標組態的篩選條件。如需物件標籤的詳細資訊，請參閱 使用標籤分類儲存空間。
- 存取點 – S3 存取點為連接到儲存貯體的指定網路端點並可簡化 S3 中共用資料集的大規模資料存取管理。當您建立存取點篩選條件時，Amazon S3 包含對您在指標組態中指定的存取點的請求。如需詳細資訊，請參閱 監控與記錄存取點。

> **Note**
>
> 當您建立依存取點篩選的指標組態時，必須使用存取點 Amazon Resource Name (ARN)，而不是存取點別名。確定您使用 ARN 作為存取點本身，而不是特定物件的 ARN。如需存取點 ARN 的更多資訊，請參閱 使用存取點。

如果您指定篩選條件，則只有對單一物件進行操作的要求才能符合篩選條件，並包含在所要求的指標中。類似要求DeleteObjects和`ListObjects`要求不會傳回任何含有篩選器之組態的指標。

若要要求更複雜的篩選，請選擇兩個或多個元素。只有具有所有這些元素的物件才會包含在指標組態中。如果您未設定篩選條件，則儲存貯體中的所有物件都會包含在指標組態中。

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇包含您要請求指標之物件的儲存貯體名稱。
3. 選擇 Metrics (指標) 標籤。
4. 在 Bucket metrics (儲存貯體指標) 下，選擇 View additional charts (檢視其他圖表)。
5. 選擇 Request metrics (請求指標) 標籤。
6. 選擇 Create filter (建立篩選條件)。
7. 在 Filter name (篩選條件名稱) 方塊中，輸入篩選條件的名稱。

   名稱可包含字母、數字、句點、破折號和底線。
8. 在 Filter scope (篩選條件範圍) 底下，選擇 Limit the scope of this filter using a prefix, object tags, and an S3 Access Point, or a combination of all three (使用字首、物件標籤和 S3 存取點或組合全部三者來限制此篩選條件的範圍)。
9. 在 Filter type (篩選條件類型) 底下，至少選擇一種篩選條件類型：字首、物件標籤或存取點。
10. 若要定義字首篩選條件並將篩選條件的範圍限制為單一路徑，請在 Prefix (字首) 方塊中，輸入字首。
11. 若要定義物件標籤篩選條件，在 Object tags (物件標籤) 下，選擇 Add tag (新增標籤)，然後輸入標籤 Key (金鑰) 和 Value (數值)。
12. 若要定義存取點篩選條件，請在 S3 Access Point (S3 存取點) 欄位中，輸入存取點 ARN，或選擇 Browse S3 (瀏覽 S3) 以導覽至存取點。

> **Important**
>
> 您無法輸入存取點別名。您必須輸入 ARN 作為存取點本身，而不是特定物件的 ARN。

13. 選擇儲存變更。

    Amazon S3 隨即會建立使用所指定字首、標籤或存取點的篩選條件。

14. 在 Request metrics (請求指標) 標籤的 Filters (篩選條件) 底下，選擇您剛才建立的篩選條件。

    您現在已建立篩選條件，可依字首、物件標籤或存取點限制請求指標範圍。 CloudWatch 開始追蹤這些請求指標大約 15 分鐘後，您可以查看 Amazon S3 和 CloudWatch 主控台上的指標圖表。請求指標會以標準費 CloudWatch 率計費。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

    您也可以在儲存貯體層級配置請求指標。如需相關資訊，請參閱為值區中的所有物件建立 CloudWatch 指標設定。

使用 AWS CLI

1. 安裝並設定 AWS CLI. 如需說明，請參閱《AWS Command Line Interface 使用者指南》中的安裝、更新和解除安裝 AWS CLI。
2. 開啟終端機。
3. 若要新增指標組態，請執行下列命令：

   Example ：若要依字首進行篩選

   ```
   aws s3api put-bucket-metrics-configuration --bucket DOC-EXAMPLE-BUCKET1 --
   id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
   {"Prefix":"prefix1"}} '
   ```

   Example ：若要依標籤進行篩選

   ```
   aws s3api put-bucket-metrics-configuration --bucket DOC-EXAMPLE-BUCKET1 --
   id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
   {"Tag": {"Key": "string", "Value": "string"}} '
   ```

Example ：若要依存取點進行篩選

```
aws s3api put-bucket-metrics-configuration --bucket DOC-EXAMPLE-BUCKET1 --
id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
{"AccessPointArn":"arn:aws:s3:Region:account-id:accesspoint/access-point-name"}} '
```

Example ：若要依字首、標籤和存取點進行篩選

```
aws s3api put-bucket-metrics-configuration --endpoint https://
s3.Region.amazonaws.com --bucket DOC-EXAMPLE-BUCKET1 --id metrics-config-id --
metrics-configuration '
{
    "Id": "metrics-config-id",
    "Filter": {
        "And": {
            "Prefix": "string",
            "Tags": [
                {
                    "Key": "string",
                    "Value": "string"
                }
            ],
            "AccessPointArn": "arn:aws:s3:Region:account-id:accesspoint/access-
point-name"
        }
    }
}'
```

使用 REST API

您也可以使用 Amazon S3 REST API，以程式設計方式新增指標組態。如需新增和使用指標組態的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列主題：

- PUT 儲存貯體指標組態
- GET 儲存貯體指標組態
- List 儲存貯體指標組態
- DELETE 儲存貯體指標組態

刪除指標篩選條件

如果不再需要 Amazon CloudWatch 請求指標篩選器，可以刪除該篩選器。刪除篩選條件時，您不再需要對使用該特定篩選條件的請求指標付費。不過，您仍需繼續對存在的任何其他篩選組態付費。

當您刪除篩選條件時，您無法再將該篩選條件用於請求指標。刪除篩選條件無法復原。

如需有關建立指標篩選條件的相關資訊，請參閱下列主題：

- 為值區中的所有物件建立 CloudWatch 指標設定
- 建立依字首、物件標籤或存取點篩選的指標組態

使用 S3 主控台

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇您的儲存貯體名稱。
3. 選擇 Metrics (指標) 標籤。
4. 在 Bucket metrics (儲存貯體指標) 下，選擇 View additional charts (檢視其他圖表)。
5. 選擇 Request metrics (請求指標) 標籤。
6. 選擇 Manage filters (管理篩選條件)。
7. 選擇您的篩選條件。

   > **Important**
   >
   > 刪除篩選條件無法復原。

8. 選擇 Delete (刪除)。

   Amazon S3 會刪除您的篩選條件。

使用 REST API

您也可以使用 Amazon S3 REST API，以程式設計方式新增指標組態。如需新增和使用指標組態的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列主題：

- PUT 儲存貯體指標組態

- GET 儲存貯體指標組態
- List 儲存貯體指標組態
- DELETE 儲存貯體指標組態

Amazon S3 事件通知

您可以使用 Amazon S3 事件通知功能，在 S3 儲存貯體中發生特定事件時接收通知。若要啟用通知，請新增通知組態，以識別您想要 Amazon S3 發佈的事件。請確定通知組態也會識別您想要 Amazon S3 傳送通知的目的地。您會將此組態存放在儲存貯體相關聯的通知子資源中。如需詳細資訊，請參閱儲存貯體組態選項。Amazon S3 提供 API，讓您管理此子資源。

> **Important**
>
> Amazon S3 事件通知的設計是要至少傳送一次。事件通知一般能在幾秒內交付，但有時候會耗費一分鐘或更長的時間。

Amazon S3 事件通知概觀

目前 Amazon S3 可以發佈下列事件的通知：

- 新物件建立的事件
- 物件移除事件
- 還原物件事件
- 低冗餘儲存 (RRS) 物件遺失事件
- 複寫事件
- S3 生命週期過期事件
- S3 生命週期轉換事件
- S3 Intelligent-Tiering 自動封存事件
- 物件標記事件
- 物件 ACL PUT 事件

如需所有支援事件類型的描述，請參閱 SQS、SNS 和 Lambda 支援的事件類型。

Amazon S3 可將事件通知訊息傳送至下列目標。您必須在通知組態中，指定這些目的地的 Amazon Resource Name (ARN) 值。

- Amazon Simple Notification Service (Amazon SNS) 主題
- Amazon Simple Queue Service (Amazon SQS) 佇列
- AWS Lambda 函數
- Amazon EventBridge

如需詳細資訊，請參閱 支援的事件目的地。

> **Note**
>
> Amazon Simple Queue Service FIFO (先進先出) 佇列不支援做為 Amazon S3 事件通知目的地。要將 Amazon S3 事件的通知發送到 Amazon SQS FIFO 隊列，您可以使用 Amazon。EventBridge如需詳細資訊，請參閱 啟用 Amazon EventBridge。

> **Warning**
>
> 如果您的通知寫入觸發通知的同一個儲存貯體，則可能會導致執行迴圈。例如，如果儲存貯體在物件每次上傳時都觸發 Lambda 函式，且該函式會將物件上傳至儲存貯體，則函式會間接地觸發本身。若要避免此狀況，請使用兩個儲存貯體，或將觸發設定為僅套用至傳入物件所用的字首。
>
> 如需詳細資訊和搭配使用 Amazon S3 通知的範例 AWS Lambda，請參閱AWS Lambda 開發人員指南中的AWS Lambda 搭配 Amazon S3 使用。

有關每個儲存貯體可建立之事件通知組態數量的詳細資訊，請參閱《AWS 一般參考》中的 Amazon S3 服務配額。

如需有關事件通知的詳細資訊，請參閱下列章節。

主題

- 事件通知類型與目的地
- 使用 Amazon SQS、Amazon SNS 和 Lambda
- 使用 EventBridge

事件通知類型與目的地

Amazon S3 支援多種事件通知類型和可發佈通知的目的地。您可以在設定事件通知時，指定事件類型和目的地。每個事件通知只能指定一個目的地。Amazon S3 事件通知會針對每個通知訊息傳送一個事件項目。

主題

支援的事件目的地

Amazon S3 可將事件通知訊息傳送至下列目標。

- Amazon Simple Notification Service (Amazon SNS) 主題
- Amazon Simple Queue Service (Amazon SQS) 佇列
- AWS Lambda
- Amazon EventBridge

但是，每個事件通知只能指定一個目的地類型。

> **Note**
>
> 您必須授予 Amazon S3 許可才能將訊息張貼到 Amazon SNS 主題或 Amazon SQS 佇列。您還必須授予 Amazon S3 權限，才能代表您叫用 AWS Lambda 函數。如需如何授予這些許可的詳細資訊，請參閱 授予許可以將事件通知訊息發佈至目標。

Amazon SNS 主題

Amazon SNS 是管理完善且靈活的推送訊息服務。您可以使用此項服務將訊息推播至行動裝置或分散式服務。有了 SNS，只需發佈一次訊息，就可以進行一或多次的傳遞。目前，標準 SNS 僅允許作為 S3 事件通知目的地，而不允許作為 SNS FIFO。

Amazon SNS 會同時協調與管理訂閱端點或用戶端的訊息傳送或送達。您可以使用 Amazon SNS 主控台來建立可將通知傳送至其中的 Amazon SNS 主題。

該主題必須與您的 Amazon S3 儲存貯體相同 AWS 區域 。如需有關如何建立 Amazon SNS 主題的資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的 Amazon SNS 入門，以及 Amazon SNS 常見問答集。

您需要有下列資訊，才能使用可建立為事件通知目的地的 Amazon SNS 主題：

- Amazon SNS 主題的 Amazon Resource Name (ARN)
- 有效的 Amazon SNS 主題訂閱。有了它，當訊息發佈至 Amazon SNS 主題時，主題訂閱者會收到通知。

Amazon SQS 佇列

Amazon SQS 提供可靠並可擴展的代管佇列，供訊息往返電腦期間儲存訊息之用。您可以使用 Amazon SQS 傳輸任何資料量，但不需要其他服務一直都能使用。您可以使用 Amazon SQS 主控台來建立可將通知傳送至其中的 Amazon SQS 佇列。

Amazon SQS 佇列必須與您的 Amazon S3 儲存貯體位於 AWS 區域 相同的位置。如需有關如何建立 Amazon SQS 佇列的指示，請參閱《Amazon Simple Queue Service 開發人員指南》中的什麼是 Amazon Simple Queue Service 和 Amazon SQS 入門。

您需要有下列資訊，才能使用 Amazon SQS 佇列作為事件通知目的地：

- Amazon SQS 佇列的 Amazon Resource Name (ARN)

> **Note**
>
> Amazon Simple Queue Service FIFO (先進先出) 佇列不支援做為 Amazon S3 事件通知目的地。要將 Amazon S3 事件的通知發送到 Amazon SQS FIFO 隊列，您可以使用 Amazon。EventBridge如需詳細資訊，請參閱 啟用 Amazon EventBridge。

Lambda 功能

您可以使用自訂邏輯 AWS Lambda 來擴充其他 AWS 服務，或建立自己的後端，以大 AWS 規模、效能和安全性運作。使用 Lambda，您可以建立離散的事件驅動應用程式，只在需要時執行。您也可以使用它來自動將這些應用程式從每天幾個請求擴展到每秒數千個請求。

Lambda 可執行自訂程式碼來回應 Amazon S3 儲存貯體事件。您可以自訂程式碼上傳到 Lambda，以建立 Lambda 函數。當 Amazon S3 偵測到特定類型的事件時，它可以將事件發佈至您的函數， AWS Lambda 並在 Lambda 中叫用您的函數。作為回應，Lambda 會執行您的函數。例如，它可能會偵測到的其中一個事件類型是建立物件的事件。

您可以使用主 AWS Lambda 控台建立 Lambda 函數，以使用 AWS 基礎設施代表您執行程式碼。Lambda 函式必須位在與 S3 儲存貯體相同的區域中。您也必須具有 Lambda 函式的名稱或 ARN，以將 Lambda 函式設定為事件通知目的地。

> **Warning**
>
> 如果您的通知寫入觸發通知的同一個儲存貯體，則可能會導致執行迴圈。例如，如果儲存貯體在物件每次上傳時都觸發 Lambda 函式，且該函式會將物件上傳至儲存貯體，則函式會間接地觸發本身。若要避免此狀況，請使用兩個儲存貯體，或將觸發設定為僅套用至傳入物件所用的字首。
>
> 如需詳細資訊和搭配使用 Amazon S3 通知的範例 AWS Lambda，請參閱AWS Lambda 開發人員指南中的AWS Lambda 搭配 Amazon S3 使用。

Amazon EventBridge

Amazon EventBridge 是一個無服務器事件總線，它接收來自 AWS 服務的事件。您可以設定規則以比對事件並將其交付至目標，例如 AWS 服務或 HTTP 端點。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南 EventBridge中的內容。

與其他目的地不同，您可以 EventBridge 針對值區啟用或停用要傳遞的事件。如果您啟用傳遞，所有事件都會傳送到 EventBridge。此外，您可以使用 EventBridge 規則將事件路由到其他目標。

SQS、SNS 和 Lambda 支援的事件類型

Amazon S3 可以發佈下列類型的事件。您必須在通知組態中指定這些事件類型。

事件類型	描述
S3：TestEvent	啟用通知時，Amazon S3 會發佈測試通知。這是為了確保主題存在，且儲存貯體擁有者擁有發佈指定主題的許可。 如果啟用通知失敗，則不會收到測試通知。

事件類型	描述
S3:ObjectCreated: * S3:: ObjectCreated 放 S3:: 文章 ObjectCreated S3:: 複ObjectCreated製 S3:ObjectCreated: CompleteMultipartUpload	PUT、POST 和 COPY 等 Amazon S3 API 操作可以建立物件。使用這些事件類型，您可以在使用特定 API 操作建立物件時啟用通知。或者，您可以使用 `s3:ObjectCreated:*` 事件類型來請求通知，無論用於建立物件的 API 為何。 `s3:ObjectCreated:CompleteMultipartUpload` 包括使用「複製」作業UploadPartCopy建立的物件。
S3:ObjectRemoved: * S3:: 刪除 ObjectRemoved S3:ObjectRemoved: DeleteMarkerCreated	透過使用ObjectRemoved事件類型，您可以在從值區中移除物件或批次物件時啟用通知。 您可以使用 `s3:ObjectRemoved:Delete` 事件類型，在刪除物件或永久刪除版本控制的物件時要求通知。或者您可以使用 `s3:ObjectRemoved:DeleteMarkerCreated` ，在建立版本控制物件的刪除標記時，請求通知。如需如何刪除版本控制物件的指示，請參閱 刪除啟用版本控制功能之儲存貯體中的物件。您也可以使用萬用字元 `s3:ObjectRemoved:*` ，在刪除物件時隨時要求通知。 這些事件通知不會提醒您生命週期組態自動刪除或操作失敗。
S3:ObjectRestore: * S3:: 文章 ObjectRestore s3:: ObjectRestore 已完成 S3:: 刪除 ObjectRestore	透過使用ObjectRestore事件類型，您可以在從 S3 Glacier 彈性擷取儲存類別、S3 Glacier 深度存檔儲存類別、S3 智慧型分層封存存取層和 S3 智慧型分層深度存檔存取層還原物件時，收到事件啟動和完成的通知。您也可以在還原的物件複本過期時收到通知。 `s3:ObjectRestore:Post` 事件類型會通知您物件還原的起始。`s3:ObjectRestore:Completed` 事件類型會通知您有關還原完成的情況。`s3:ObjectRestore:Delete` 事件類型會在還原物件的臨時複本過期時通知您。

事件類型	描述
S3：ReducedRedundancyLostObject	您會在 Amazon S3 偵測到缺少 RRS 儲存類別物件時，收到此通知事件。
s3:Replication:* S3: 複製：OperationFailedReplication S3: 複製：OperationMissedThreshold S3: 複製：OperationReplicatedAfterThreshold S3: 複製：OperationNotTracked	透過使用「複寫」事件類型，您可以收到已啟用 S3 複寫指標或 S3 複寫時間控制 (S3 RTC) 之複寫組態的通知。您可以追蹤擱置的位元組、擱置中的作業和複寫延遲，來監視複寫事件的 minute-by-minute 進度。如需複寫指標的相關資訊，請參閱使用複寫指標和 S3 事件通知監控進度。 `s3:Replication:OperationFailedReplication` 事件類型會在有資格複寫的物件無法複寫時通知您。`s3:Replication:OperationMissedThreshold` 事件類型會在有資格複寫的物件超過複寫的 15 分鐘閾值時通知您。 當有資格使用 S3 複寫時間控制進行複寫的物件，在 15 分鐘閾值之後複寫時，`s3:Replication:OperationReplicatedAfterThreshold` 事件類型會通知您。當有資格使用 S3 複寫時間控制進行複寫的物件但不再透過複寫指標追蹤時，`s3:Replication:OperationNotTracked` 事件類型會通知您。
S3:LifecycleExpiration: * S3:: 刪除 LifecycleExpiration S3:LifecycleExpiration: DeleteMarkerCreated	透過使用LifecycleExpiration事件類型，您可以在 Amazon S3 根據 S3 生命週期組態刪除物件時收到通知。 `s3:LifecycleExpiration:Delete` 事件類型會在刪除未進行版本控制之儲存貯體中的物件時通知您。當 S3 生命週期組態永久刪除物件版本時，它也會通知您。S3 生命週期在建立用於刪除版本控制儲存貯體中物件之最新版本的刪除標記時，`s3:LifecycleExpiration:DeleteMarkerCreated` 事件類型會通知您。
S3：LifecycleTransition	當物件透過 S3 生命週期組態轉換至另一個 Amazon S3 儲存類別時，您會收到此通知事件。

事件類型	描述
S3：IntelligentTiering	當 S3 Intelligent-Tiering 儲存類別中的物件移至封存存取層或 Deep Archive 存取層時，您會收到此通知事件。
S3:ObjectTagging: * S3:: ObjectTagging 放 S3:: 刪除 ObjectTagging	透過使用ObjectTagging事件類型，您可以在物件中新增或刪除物件標籤時啟用通知。 `s3:ObjectTagging:Put` 事件類型會在物件上的標籤為 PUT 或現有標籤已更新時通知您。`s3:ObjectTagging:Delete` 事件類型會在標籤從物件移除時通知您。
S3:: ObjectAcl 放	當 ACL 在物件上標記為 PUT 或現有 ACL 變更時，您會收到此通知事件。當請求對物件的 ACL 沒有變更時，則不會產生事件。

Amazon 支援的事件類型 EventBridge

如需 Amazon S3 將傳送至 Amazon 的事件類型清單 EventBridge，請參閱使用 EventBridge。

事件排序和重複事件

Amazon S3 事件通知的設計目的是至少交付一次通知，但不能保證它們的到達順序與事件發生的順序相同。在極少數情況下，Amazon S3 的重試機制可能會導致同一物件事件重複的 S3 事件通知。如需有關處理重複或亂序事件的詳細資訊，請參閱 S AWS torage 部落格上的使用 Amazon S3 事件通知管理事件訂購和重複事件。

使用 Amazon SQS、Amazon SNS 和 Lambda

啟用通知是儲存貯體層級的操作。您可以將通知組態資訊存放在與儲存貯體關聯的通知子資源中。建立或變更儲存貯體通知組態後，通常要等待 5 分鐘變更才會生效。首次啟用通知時，則會發生 `s3:TestEvent`。您可以使用下列任一方法來管理通知組態：

- 使用 Amazon S3 主控台 – 您可以使用主控台 UI 在儲存貯體上設定通知組態，無需編寫任何程式碼。如需詳細資訊，請參閱 使用 Amazon S3 主控台啟用和設定事件通知。

- 以程式設計方式使用 AWS 開發套件 — 在內部，主控台和開發套件都會呼叫 Amazon S3 REST API 來管理與儲存貯體關聯的通知子資源。如需使用 AWS SDK 的通知組態範例，請參閱 演練：設定儲存貯體的通知 (SNS 主題或 SQS 佇列)。

> **Note**
>
> 您還可以直接透過程式碼進行 Amazon S3 REST API 呼叫。但這麼做會很麻煩，因為這需要編寫程式碼來對您的請求進行身分驗證。

無論您使用何種方法，Amazon S3 都會將通知組態以 XML 的形式存放在與儲存貯體相關聯的通知子資源中。如需儲存貯體子資源的相關資訊，請參閱 儲存貯體組態選項。

主題

- 授予許可以將事件通知訊息發佈至目標
- 使用 Amazon S3 主控台啟用和設定事件通知
- 以程式設計方式設定事件通知
- 演練：設定儲存貯體的通知 (SNS 主題或 SQS 佇列)
- 使用物件金鑰名稱篩選來設定事件通知
- 事件訊息結構

授予許可以將事件通知訊息發佈至目標

您必須將必要的許可授予 Amazon S3 主體，以呼叫相關的 API，將訊息發佈至 SNS 主題、SQS 佇列或 Lambda 函數。這樣 Amazon S3 便可將事件通知訊息發佈至目的地。

若要進行疑難排解，以將事件通知發佈到目的地，請參閱將 Amazon S3 事件通知發佈到 Amazon Simple Notification Service 主題的疑難排解。

主題

- 授與叫用 AWS Lambda 函數的權限
- 授予許可以將訊息發佈到 SNS 主題或 SQS 佇列

授與叫用 AWS Lambda 函數的權限

Amazon S3 AWS Lambda 透過叫用 Lambda 函數並提供事件訊息做為引數，將事件訊息發佈到。

當您使用 Amazon S3 主控台，在 Amazon S3 儲存貯體上設定 Lambda 函數的事件通知時，該主控台將設定 Lambda 函數的必要許可。這樣 Amazon S3 便可從儲存貯體叫用函數。如需詳細資訊，請參閱使用 Amazon S3 主控台啟用和設定事件通知。

您也可以從授與 Amazon S3 許可以叫 AWS Lambda 用您的 Lambda 函數。如需詳細資訊，請參閱AWS Lambda 開發人員指南中的教學課程： AWS Lambda 搭配 Amazon S3 使用。

授予許可以將訊息發佈到 SNS 主題或 SQS 佇列

若要授與 Amazon S3 許可以將訊息發佈到 SNS 主題或 SQS 佇列，請將 AWS Identity and Access Management (IAM) 政策附加到目的地 SNS 主題或 SQS 佇列。

如需如何將政策連接到 SNS 主題或 SQS 佇列的範例，請參閱 演練：設定儲存貯體的通知 (SNS 主題或 SQS 佇列)。如需許可的詳細資訊，請參閱下列主題：

- 《Amazon Simple Notification Service 開發人員指南》中的 Amazon SNS 存取控制的範例案例
- 《Amazon Simple Queue Service 開發人員指南》中的 Amazon SQS 中的 Identity and Access Management

目的地 SNS 主題的 IAM 政策

以下是附加至目標 SNS 主題的 AWS Identity and Access Management (IAM) 政策範例。如需有關如何使用此政策為事件通知設定目的地 Amazon SNS 主題的指示，請參閱 演練：設定儲存貯體的通知 (SNS 主題或 SQS 佇列)。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "Example SNS topic policy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "SNS-topic-ARN",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:bucket-name"
```

```
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

目的地 SQS 佇列的 IAM 政策

以下是您附加至目標 SQS 佇列之 IAM 政策的範例。如需有關如何使用此政策為事件通知設定目的地 Amazon SQS 佇列的指示，請參閱 演練：設定儲存貯體的通知 (SNS 主題或 SQS 佇列)。

若要使用此政策，您必須更新 Amazon SQS 佇列 ARN、儲存貯體名稱和儲存貯體擁有者的 AWS 帳戶識別碼。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SQS:SendMessage"
            ],
            "Resource": "arn:aws:sqs:Region:account-id:queue-name",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

針對 Amazon SNS 和 Amazon SQS IAM 政策，您可以在政策中指定 `StringLike` 條件，不是 `ArnLike` 條件。

使用 `ArnLike` 時，ARN 的分割區、服務、帳戶 ID、資源類型和部分資源 ID 部分必須與請求內容中的 ARN 完全相符。只有區域和資源路徑允許部分相符。

當使用 `StringLike` 而非 `ArnLike` 時，比對會忽略 ARN 結構並允許部分相符，而不管該萬用字元部分。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素。

```
"Condition": {
  "StringLike": { "aws:SourceArn": "arn:aws:s3:*:*:bucket-name" }
  }
```

AWS KMS 金鑰政策

如果 SQS 佇列或 SNS 主題使用 AWS Key Management Service (AWS KMS) 客戶受管金鑰加密，您必須授與 Amazon S3 服務主體權限才能處理加密的主題或佇列。若要將許可授予 Amazon S3 服務委託人，請將下列陳述式新增至客戶受管金鑰的金鑰政策。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

如需關 AWS KMS 鍵原則的詳細資訊，請參閱AWS Key Management Service 開發人員指南 AWS KMS中的使用金鑰原則。

如需將伺服器端加密與 Amazon SQS 和 Amazon SNS 搭配 AWS KMS 使用的詳細資訊，請參閱以下內容：

- 《Amazon Simple Notification Service 開發人員指南》中的金鑰管理。
- 《Amazon Simple Queue Service 開發人員指南》中的金鑰管理。
- AWS 運算部落格中的透過 AWS KMS加密發佈到 Amazon SNS 的訊息。

使用 Amazon S3 主控台啟用和設定事件通知

您可以啟用特定的 Amazon S3 儲存貯體事件，於發生事件時傳送通知訊息給目的地。本節說明如何使用 Amazon S3 主控台來啟用事件通知。如需如何搭配 AWS 開發套件和 Amazon S3 REST API 使用事件通知的相關資訊，請參閱以程式設計方式設定事件通知。

先決條件：您必須先設定其中一個目的地類型，然後設定許可，才能啟用儲存貯體的事件通知。如需更多詳細資訊，請參閱 支援的事件目的地 及 授予許可以將事件通知訊息發佈至目標。

> **Note**
>
> Amazon Simple Queue Service FIFO (先進先出) 佇列不支援做為 Amazon S3 事件通知目的地。要將 Amazon S3 事件的通知發送到 Amazon SQS FIFO 隊列，您可以使用 Amazon。EventBridge如需詳細資訊，請參閱 啟用 Amazon EventBridge。

主題

- 使用 Amazon S3 主控台啟用 Amazon SNS、Amazon SQS 或 Lambda 通知

使用 Amazon S3 主控台啟用 Amazon SNS、Amazon SQS 或 Lambda 通知

啟用及設定 S3 儲存貯體的事件通知

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在儲存貯體名稱清單中，選擇要啟用事件的儲存貯體名稱。
3. 選擇 Properties (屬性)。
4. 瀏覽至事件通知區段，然後選擇建立事件通知。
5. 在一般組態區段中，指定事件通知的描述性事件名稱。或者，您也可以指定首碼和字尾，將通知限制在具有以指定字元結尾的金鑰的物件。

a. 輸入事件名稱的說明。

如果您未輸入名稱，則會產生全域唯一識別碼 (GUID) 並用於名稱。

b. (選用) 若要依字首篩選事件通知，請輸入 Prefix (字首)。

例如，您可以設定字首篩選條件，只在檔案新增至特定資料夾 (例如，`images/`) 時才收到通知。

c. (選用) 若要依尾碼篩選事件通知，請輸入 Prefix (尾碼)。

如需詳細資訊，請參閱 使用物件金鑰名稱篩選來設定事件通知。

6. 在 Event types (事件類型) 區段中，選取您要接收通知的一或多個事件類型。

如需不同事件類型的清單，請參閱 SQS、SNS 和 Lambda 支援的事件類型。

7. 在目的地區段中，選擇事件通知目的地。

> **Note**
>
> 在發佈事件通知之前，您必須授予 Amazon S3 主體必要的許可來呼叫相關 API。這樣可以將通知發佈至 Lambda 函數、SNS 主題或 SQS 佇列。

a. 選取目的地類型： Lambda 函式、SNS 主題或 SQS 佇列。

b. 選擇目的地類型後，請從清單中選擇功能、主題或佇列。

c. 或者，如果您想要指定 Amazon Resource Name (ARN)，請選取輸入 ARN 並輸入 ARN。

如需詳細資訊，請參閱 支援的事件目的地。

8. 選擇 Save changes (儲存變更)，Amazon S3 會將測試訊息傳送到事件通知目的地。

以程式設計方式設定事件通知

根據預設，所有事件類型都不會啟用通知。因此，通知子資源一開始存放的是空白組態。

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</NotificationConfiguration>
```

若要啟用特定類型事件的通知，請以合適的組態取代 XML，識別 Amazon S3 所要發佈的事件類型，以及所需的事件發佈目標。為每個目標新增對應的 XML 組態。

將事件訊息發佈至 SQS 佇列

若要將 SQS 佇列設定為一或多種事件類型的通知目標，請新增 `QueueConfiguration`。

```
<NotificationConfiguration>
  <QueueConfiguration>
    <Id>optional-id-string</Id>
    <Queue>sqs-queue-arn</Queue>
    <Event>event-type</Event>
    <Event>event-type</Event>
     ...
  </QueueConfiguration>
   ...
</NotificationConfiguration>
```

將事件訊息發佈至 SNS 主題

若要將 SNS 主題設定為特定事件類型的通知目標，請新增 `TopicConfiguration`。

```
<NotificationConfiguration>
  <TopicConfiguration>
     <Id>optional-id-string</Id>
     <Topic>sns-topic-arn</Topic>
     <Event>event-type</Event>
     <Event>event-type</Event>
      ...
  </TopicConfiguration>
   ...
</NotificationConfiguration>
```

叫用 AWS Lambda 函數並提供事件訊息作為引數

若要將 Lambda 函數設定為特定事件類型的通知目的地，請新增 `CloudFunctionConfiguration`。

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>optional-id-string</Id>
     <CloudFunction>cloud-function-arn</CloudFunction>
```

```
      <Event>event-type</Event>
      <Event>event-type</Event>
       ...
   </CloudFunctionConfiguration>
    ...
</NotificationConfiguration>
```

移除儲存貯體上設定的所有通知

若要移除儲存貯體上設定的所有通知，請在通知子資源中儲存空白的 <NotificationConfiguration/> 元素。

當 Amazon S3 偵測到特定類型的事件，即會發佈附事件資訊的訊息。如需詳細資訊，請參閱 事件訊息結構。

如需有關設定事件通知的資訊，請參閱下列主題：

- 演練：設定儲存貯體的通知 (SNS 主題或 SQS 佇列).
- 使用物件金鑰名稱篩選來設定事件通知

演練：設定儲存貯體的通知 (SNS 主題或 SQS 佇列)

您可以使用 Amazon Simple Notification Service (Amazon SNS) 或 Amazon Simple Queue Service (Amazon SQS) 接收 Amazon S3 通知。在此演練中，您使用 Amazon SNS 主題和 Amazon SQS 佇列將通知設定新增至儲存貯體。

> **Note**
>
> Amazon Simple Queue Service FIFO (先進先出) 佇列不支援做為 Amazon S3 事件通知目的地。要將 Amazon S3 事件的通知發送到 Amazon SQS FIFO 隊列，您可以使用 Amazon。EventBridge如需詳細資訊，請參閱 啟用 Amazon EventBridge。

主題

- 演練摘要
- 步驟 1：建立 Amazon SQS 佇列
- 步驟 2：建立 Amazon SNS 主題
- 步驟 3：將通知組態新增至儲存貯體

- 步驟 4：測試設定

演練摘要

此演練可以協助您執行以下操作：

- 將 `s3:ObjectCreated:*` 類型的事件發佈至 Amazon SQS 佇列。
- 將 `s3:ReducedRedundancyLostObject` 類型的事件發佈至 Amazon SNS 主題。

如需通知組態的資訊，請參閱「使用 Amazon SQS、Amazon SNS 和 Lambda」。

您可以使用主控台執行所有這些步驟，不用撰寫任何程式碼。此外，還提供了使用適用於 Java 和 .NET 的 AWS SDK 的代碼示例，以幫助您以編程方式添加通知配置。

此程序包含以下步驟：

1. 建立 Amazon SQS 佇列。

   使用 Amazon SQS 主控台建立 SQS 佇列。您可以利用程式設計的方式，存取 Amazon S3 傳送到佇列的任何訊息。但在此演練中，您會在主控台中驗證通知訊息。

   您要將存取政策連接到佇列，以授予 Amazon S3 張貼訊息的許可。
2. 建立 Amazon SNS 主題。

   使用 Amazon SNS 主控台，建立 SNS 主題並訂閱主題。如此一來，張貼其上的任何事件都會傳遞給您。您會將電子郵件指定為通訊協定。建立主題後，Amazon SNS 會傳送電子郵件。您可以使用電子郵件中的連結，確認主題訂閱。

   您要將存取政策連接到主題，以授予 Amazon S3 張貼訊息的許可。
3. 將通知組態新增至儲存貯體。

步驟 1：建立 Amazon SQS 佇列

按照步驟建立和訂閱 Amazon Simple Queue Service (Amazon SQS) 佇列。

1. 使用 Amazon SQS 主控台建立佇列。如需指示，請參閱《Amazon Simple Queue Service 開發人員指南》中的 Amazon SQS 入門。
2. 將連接至佇列的存取政策以下列政策取代。

a. 在 Amazon SQS 主控台的 Queues (佇列) 清單中，選擇佇列名稱。

b. 在 Access policy (存取政策) 標籤中選擇 Edit (編輯)。

c. 取代連接至佇列的存取政策。在其中，提供 Amazon SQS ARN、來源儲存貯體名稱以及儲存貯體擁有者帳戶 ID。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SQS:SendMessage"
            ],
            "Resource": "SQS-queue-ARN",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

d. 選擇儲存。

3. (選擇性) 如果 Amazon SQS 佇列或 Amazon SNS 主題已啟用伺服器端加密 AWS Key Management Service (AWS KMS)，請將下列政策新增至關聯的對稱加密客戶受管金鑰。

   您必須將政策新增至客戶受管金鑰，因為您無法修改適用於 Amazon SQS 或 Amazon SNS 的 AWS 受管金鑰。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
```

```
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

如需將 SSE 與 Amazon SQS 和 Amazon SNS 搭配使用的相關資訊 AWS KMS，請參閱下列內容：

- 《Amazon Simple Notification Service 開發人員指南》中的金鑰管理。
- 《Amazon Simple Queue Service 開發人員指南》中的金鑰管理。

4. 記下佇列 ARN。

   您建立的 SQS 佇列是 AWS 帳戶中的另一個資源。它具有獨特的 Amazon Resource Name (ARN)。下一個步驟需要用到此 ARN。ARN 的格式如下：

   ```
   arn:aws:sqs:aws-region:account-id:queue-name
   ```

步驟 2：建立 Amazon SNS 主題

按照步驟建立並訂閱 Amazon SNS 主題。

1. 使用 Amazon SNS 主控台建立一個主題。如需說明，請參閱《Amazon Simple Notification Service 開發人員指南》中的建立 Amazon SNS 主題。
2. 訂閱此主題。在此練習中，請使用電子郵件作為通訊協定。如需詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的訂閱 Amazon SNS 主題。

   您會收到要求確認訂閱主題的電子郵件。確認訂閱。

3. 下列列政策取代連接至主題的存取政策。在其中，提供您的 SNS 主題 ARN、儲存貯體名稱以及儲存貯體擁有者帳戶 ID。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "Example SNS topic policy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "SNS-topic-ARN",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:bucket-name"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

4. 請記下主題 ARN。

   您建立的 SNS 主題是您的另一個資源 AWS 帳戶，它具有唯一的 ARN。下一個步驟需要用到此 ARN。ARN 的格式如下：

```
arn:aws:sns:aws-region:account-id:topic-name
```

步驟 3：將通知組態新增至儲存貯體

您可以使用 Amazon S3 主控台或使用 AWS 開發套件以程式設計方式啟用儲存貯體通知。選擇任一選項在儲存貯體上設定通知。本節提供使用適用於 Java 和 .NET 的 AWS SDK 程式碼範例。

(選項 A)：使用主控台啟用儲存貯體上的通知

使用 Amazon S3 主控台，新增請求 Amazon S3 的通知組態新增以執行下列操作：

- 將所有物件建立事件類型的事件發佈至您的 Amazon SQS 佇列。
- 將 RRS 中物件遺失 類型的事件發佈至您的 Amazon SNS 主題。

儲存通知組態後，Amazon S3 會張貼測試訊息，您會透過電子郵件收到此訊息。

如需說明，請參閱使用 Amazon S3 主控台啟用和設定事件通知。

選項 B：使用 AWS SDK 在值區上啟用通知

.NET

下列 C# 程式碼範例提供完整的程式碼清單，將通知組態新增至儲存貯體。您必須更新程式碼，並提供您的儲存貯體名稱和 SNS 主題 ARN。如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class EnableNotificationsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string snsTopic = "*** SNS topic ARN ***";
        private const string sqsQueue = "*** SQS topic ARN ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            EnableNotificationAsync().Wait();
        }
```

```
        static async Task EnableNotificationAsync()
        {
            try
            {
               PutBucketNotificationRequest request = new
 PutBucketNotificationRequest
                {
                    BucketName = bucketName
                };

                TopicConfiguration c = new TopicConfiguration
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic
                };
                request.TopicConfigurations = new List<TopicConfiguration>();
                request.TopicConfigurations.Add(c);
                request.QueueConfigurations = new List<QueueConfiguration>();
                request.QueueConfigurations.Add(new QueueConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedPut },
                    Queue = sqsQueue
                });

                PutBucketNotificationResponse response = await
 client.PutBucketNotificationAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' ",
 e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown error encountered on server.
 Message:'{0}' ", e.Message);
            }
        }
    }
}
```

Java

以下範例說明如何將通知設定新增至儲存貯體。如需如何建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.EnumSet;

public class EnableNotificationOnABucket {

    public static void main(String[] args) throws IOException {
        String bucketName = "*** Bucket name ***";
        Regions clientRegion = Regions.DEFAULT_REGION;
        String snsTopicARN = "*** SNS Topic ARN ***";
        String sqsQueueARN = "*** SQS Queue ARN ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();
            BucketNotificationConfiguration notificationConfiguration = new
 BucketNotificationConfiguration();

            // Add an SNS topic notification.
            notificationConfiguration.addConfiguration("snsTopicConfig",
                    new TopicConfiguration(snsTopicARN,
 EnumSet.of(S3Event.ObjectCreated)));

            // Add an SQS queue notification.
            notificationConfiguration.addConfiguration("sqsQueueConfig",
                    new QueueConfiguration(sqsQueueARN,
 EnumSet.of(S3Event.ObjectCreated)));
```

```
            // Create the notification configuration request and set the bucket
 notification
            // configuration.
            SetBucketNotificationConfigurationRequest request = new
 SetBucketNotificationConfigurationRequest(
                    bucketName, notificationConfiguration);
            s3Client.setBucketNotificationConfiguration(request);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

步驟 4：測試設定

現在，只要將物件上傳至儲存貯體，即可測試設定，並在 Amazon SQS 主控台中驗證事件通知。如需指示，請參閱 Amazon Simple Queue Service 開發人員指南「入門」一節中的接收訊息。

使用物件金鑰名稱篩選來設定事件通知

設定 Amazon S3 事件通知時，您必須指定哪些支援的 Amazon S3 事件類型會導致 Amazon S3 傳送通知。如果您未指定的事件類型在 S3 儲存貯體中發生，則 Amazon S3 不會傳送通知。

您可以設定按物件金鑰名稱的字首和尾碼篩選通知。例如，您可以設定組態，只在將副檔名為「.jpg」的映像檔案新增至儲存貯體時才傳送通知。或者，您也可以擁有一個組態，在將具有前置詞 "images/" 的物件新增至值區時，將具有 "logs/" 前綴的物件傳送至 AWS Lambda 函數時，將通知傳送至 Amazon SNS 主題。

> **Note**
>
> 萬用字元 ("*") 不能在篩選條件中用作前綴或後綴。如果您的字首或字尾包含空格，則必須將其取代為 "+" 字元。如果您在字首或字尾的值中使用任何其他特殊字元，則必須以 URL 編碼 (百

分比編碼) 格式將其輸入。如需在事件通知的字首或字尾中使用時，必須轉換為 URL 編碼格式之特殊字元的完整清單，請參閱 安全字元。

您可以設定通知組態，在 Amazon S3 主控台使用物件金鑰名稱篩選。您可以透過 AWS 開發套件或其餘 API 直接使用 Amazon S3 API 來執行此操作。如需使用主控台 UI 在儲存貯體上設定通知組態的相關資訊，請參閱「使用 Amazon S3 主控台啟用和設定事件通知」。

Amazon S3 會將通知組態儲存為 XML，放在與儲存貯體相關聯的通知子資源中，如 使用 Amazon SQS、Amazon SNS 和 Lambda 所述。您必須使用 `Filter` XML 結構定義規則，依物件金鑰名稱的字首或尾碼篩選通知。如需 `Filter` XML 結構的詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的 PUT 儲存貯體通知。

使用 `Filter` 的通知組態無法定義重疊字首、重疊尾碼或字首和尾碼重疊的篩選規則。下列小節提供使用物件金鑰名稱篩選的有效通知組態範例。它們也包含因為字首和尾碼重疊而無效的通知組態範例。

主題

- 以物件金鑰名稱篩選的有效通知組態範例
- 無效字首和尾碼重疊的通知組態範例

以物件金鑰名稱篩選的有效通知組態範例

下列通知組態包含的佇列組態，可識別 Amazon SQS 佇列，讓 Amazon S3 將事件發佈成 `s3:ObjectCreated:Put` 類型。只要字首為 `images/` 和尾碼為 `jpg` 的物件 PUT 到儲存貯體，就會發佈事件。

```
<NotificationConfiguration>
  <QueueConfiguration>
      <Id>1</Id>
      <Filter>
          <S3Key>
              <FilterRule>
                  <Name>prefix</Name>
                  <Value>images/</Value>
              </FilterRule>
              <FilterRule>
                  <Name>suffix</Name>
                  <Value>jpg</Value>
              </FilterRule>
          </S3Key>
```

```
      </Filter>
      <Queue>arn:aws:sqs:us-west-2:444455556666:s3notificationqueue</Queue>
      <Event>s3:ObjectCreated:Put</Event>
   </QueueConfiguration>
</NotificationConfiguration>
```

下列通知組態有多個非重疊字首。組態的定義是：images/ 資料夾中的 PUT 請求通知排入佇列 A，而 logs/ 資料夾中的 PUT 請求通知排入佇列 B。

```
<NotificationConfiguration>
  <QueueConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images/</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-A</Queue>
     <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
  <QueueConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>logs/</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-B</Queue>
     <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
</NotificationConfiguration>
```

下列通知組態有多個非重疊尾碼。組態的定義是：所有近期新增至儲存貯體的 .jpg 映像都由 Lambda cloud-function-A 處理，而所有近期新增的 .png 影像則由 cloud-function-B 處理。.png 和 .jpg 字尾即使最後一個字母相同也不重疊。如果指定的字串可以這兩個尾碼結束，則兩個尾碼視為重疊。字串的結尾不能是 .png 和 .jpg，所以範例組態中的字尾不是重疊的字尾。

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.jpg</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
  <CloudFunctionConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.png</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
</NotificationConfiguration>
```

使用 `Filter` 的通知組態無法定義相同事件類型的重疊字首篩選規則。如果重疊字首搭配的尾碼不重疊，則只能這麼做。下列範例組態示範如何將以常見字首但不重疊尾碼建立的物件，傳送至不同的目標。

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
```

```
                    <Value>images</Value>
                </FilterRule>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.jpg</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
  <CloudFunctionConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images</Value>
                </FilterRule>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.png</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
</NotificationConfiguration>
```

無效字首和尾碼重疊的通知組態範例

在大多數情況下，使用 Filter 的通知組態無法定義相同事件類型的重疊字首、重疊尾碼或字首和尾碼重疊組合的篩選規則。只要字首不重疊就可以有重疊的前綴。如需範例，請參閱使用物件金鑰名稱篩選來設定事件通知。

不同的事件類型可以使用重疊的物件金鑰名稱篩選。例如，您可以建立通知組態，在 image/ 事件類型使用字首 ObjectCreated:Put，在 image/ 事件類型使用字首 ObjectRemoved:*。

當您使用 Amazon S3 主控台或 API 時，如果嘗試儲存的通知組態其相同事件類型具無效的重疊名稱篩選條件，就會發生錯誤。本節示範因為重疊名稱篩而無效的通知組態範例。

現有的全部通知組態，都假設分別擁有與任何其他字首和尾碼符合的預設字首和尾碼。下列通知組態因有重疊的字首而無效。明確地說，根字首會與任何其他字首重疊。在本範例中，如果使用尾碼而不使用字首，也是同樣的情況。根尾碼會與任何其他尾碼重疊。

```
<NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-two</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>prefix</Name>
                     <Value>images</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration>
```

下列通知組態因有重疊的字尾而無效。如果指定的字串可以這兩個尾碼結束，則兩個尾碼視為重疊。字串的結尾可以使用 jpg 和 pg。所以，字尾會重疊。字首也是一樣。如果指定的字串可以這兩個字首開始，則兩個字首視為重疊。

```
 <NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-two</Topic>
```

```
         <Event>s3:ObjectCreated:Put</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>pg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration
```

下列通知組態因有重疊的字首和字尾而無效。

```
<NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>prefix</Name>
                     <Value>images</Value>
                 </FilterRule>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-two</Topic>
         <Event>s3:ObjectCreated:Put</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
```

```
</NotificationConfiguration>
```

事件訊息結構

Amazon S3 傳送以發佈事件的通知訊息為 JSON 格式。

如需設定事件通知的一般概觀和指示，請參閱 Amazon S3 事件通知。

此範例顯示事件通知 JSON 結構的 2.2 版。Amazon S3 使用該事件結構的版本 2.1、2.2 和 2.3。Amazon S3 會將版本 2.2 用於跨區域複寫事件通知。它將版本 2.3 用於 S3 生命週期、S3 Intelligent-Tiering、物件 ACL、物件標記和物件還原刪除事件。這些版本包含特定於這些操作的額外資訊。版本 2.2 及 2.3 在其他方面與版本 2.1 相容，而 Amazon S3 目前可用於全部其他事件通知類型。

```
{
   "Records":[
      {
         "eventVersion":"2.2",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"The time, in ISO-8601 format, for example,
 1970-01-01T00:00:00.000Z, when Amazon S3 finished processing the request",
         "eventName":"event-type",
         "userIdentity":{
            "principalId":"Amazon-customer-ID-of-the-user-who-caused-the-event"
         },
         "requestParameters":{
            "sourceIPAddress":"ip-address-where-request-came-from"
         },
         "responseElements":{
            "x-amz-request-id":"Amazon S3 generated request ID",
            "x-amz-id-2":"Amazon S3 host that processed the request"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"ID found in the bucket notification configuration",
            "bucket":{
               "name":"bucket-name",
               "ownerIdentity":{
                  "principalId":"Amazon-customer-ID-of-the-bucket-owner"
               },
               "arn":"bucket-ARN"
            },
            "object":{
```

```
                "key":"object-key",
                "size":"object-size in bytes",
                "eTag":"object eTag",
                "versionId":"object version if bucket is versioning-enabled, otherwise
 null",
                "sequencer": "a string representation of a hexadecimal value used to
 determine event sequence, only used with PUTs and DELETEs"
             }
          },
          "glacierEventData": {
             "restoreEventData": {
                "lifecycleRestorationExpiryTime": "The time, in ISO-8601 format, for
 example, 1970-01-01T00:00:00.000Z, of Restore Expiry",
                "lifecycleRestoreStorageClass": "Source storage class for restore"
             }
          }
       }
    ]
}
```

請注意，以下是事件訊息結構：

- eventVersion 金鑰值包含以 <major>.<minor> 形式的主要和次要版本。

  如果 Amazon S3 對不向下相容的事件結構進行變更，主要版本就會增加。這包含已經存在的 JSON 欄位，或變更欄位內容的呈現方式 (例如，日期格式)。

  如果 Amazon S3 將新欄位新增至事件結構，次要版本就會增加。如果針對部分或所有現有事件提供新資訊，則可能會發生這種情況。如果只有新引進的事件類型才提供新資訊，也可能會發生這種情況。應用程式應略過新欄位，以與新次要版本的事件結構維持正向相容。

  如果引進新的事件類型，但卻未修改事件結構，則事件版本不會變更。

  為確保您的應用程式可以正確剖析事件結構，我們建議您對主要版本編號進行「等於」比較。為了確保您的應用程序預期的字段存在，我們還建議對次要版本進行 greater-than-or-equal-to 比較。

- 所以 eventName 會參考事件通知類型的清單，但不會包含 s3: 字首。
- 如果您想通過跟進來跟踪請求，則responseElements鍵值非常有用 AWS Support。x-amz-request-id 和 x-amz-id-2 都能協助 Amazon S3 追蹤個別請求。這些值與 Amazon S3 為回應起始事件之請求而傳回的值相同。因此，它們可以用來比對事件與請求。

- `s3` 金鑰提供與事件有關之儲存貯體和物件的相關資訊。此物件金鑰名稱值是以 URL 編碼而成。例如「red flower.jpg」會變成「red+flower.jpg」(Amazon S3 傳回「`application/x-www-form-urlencoded`」當做此回應內容類型)。
- `sequencer` 金鑰提供判斷事件順序的方式。事件通知不保證按事件發生的相同順序抵達。不過，來自建立物件 (PUT) 和刪除物件之事件的通知包含 `sequencer`。它可以用來確定指定物件金鑰的事件順序。

  如果比較 `sequencer` 字串和相同物件金鑰的兩項事件通知，`sequencer` 十六進位值較大的事件通知是發生較晚的事件。如果使用事件通知維護個別的 Amazon S3 物件資料庫或索引，則建議您在處理每項事件通知時，比較並存放 `sequencer` 值。

  注意下列事項：

  - 您不能使用 `sequencer` 來判斷不同物件金鑰上的事件順序。
  - 排序器長度各不相同。所以，為了比較這些值，請先在較短值的右邊填充零，再執行字典排序比較。
- 只有 `s3:ObjectRestore:Completed` 事件才可見到 `glacierEventData`金鑰。
- `restoreEventData` 鍵包含與您的還原請求相關的屬性。
- `replicationEventData` 金鑰只會對複寫事件顯示。
- `intelligentTieringEventData` 鍵只會對 S3 Intelligent-Tiering 事件顯示。
- `lifecycleEventData` 鍵只會對 S3 生命週期轉換事件顯示。

範例訊息

以下是 Amazon S3 事件通知訊息的範例。

Amazon S3 測試訊息

您在儲存貯體上設定事件通知後，Amazon S3 會傳送下列測試訊息。

```
{
   "Service":"Amazon S3",
   "Event":"s3:TestEvent",
   "Time":"2014-10-13T15:57:02.089Z",
   "Bucket":"bucketname",
   "RequestId":"5582815E1AEA5ADF",
   "HostId":"8cLeGAmw098X5cv4Zkwcmo8vvZa3eH3eKxsPzbB9wrR+YstdA6Knx4Ip8EXAMPLE"
}
```

使用 PUT 請求建立物件時的範例訊息

以下訊息是 Amazon S3 傳送以發佈 `s3:ObjectCreated:Put` 事件的訊息範例。

```
{
   "Records":[
      {
         "eventVersion":"2.1",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"ObjectCreated:Put",
         "userIdentity":{
            "principalId":"AIDAJDPLRKLG7UEXAMPLE"
         },
         "requestParameters":{
            "sourceIPAddress":"127.0.0.1"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"mybucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::mybucket"
            },
            "object":{
               "key":"HappyFace.jpg",
               "size":1024,
               "eTag":"d41d8cd98f00b204e9800998ecf8427e",
               "versionId":"096fKKXTRTtl3on89fVO.nfljtsv6qko",
               "sequencer":"0055AED6DCD90281E5"
            }
         }
      }
   ]
```

```
}
```

對於每個 IAM 識別碼字首 (例如 AIDA、AROA、AGPA) 的定義，請參閱《IAM 使用者指南》中的 IAM 識別符。

使用 EventBridge

只要儲存貯體中發生某些事件，Amazon S3 EventBridge 就可以將事件傳送到 Amazon。與其他目的地不同，您不需要選取想要傳遞的事件類型。啟 EventBridge 用後，以下所有事件都會傳送至 EventBridge。您可以使用 EventBridge 規則將事件路由傳送至其他目標。以下列出 Amazon S3 傳送到的事件 EventBridge。

事件類型	描述
物件已建立	已建立物件。 事件訊息結構中的原因欄位會指出使用哪個 S3 API 建立物件：PutObjectCopyObject、POST 物件或CompleteMultipartUpload。
已刪除物件 (DeleteObject) 物件已刪除 (生命週期過期)	已刪除物件。 使用 S3 API 呼叫刪除物件時，原因欄位會設定為 DeleteObject。當 S3 生命週期過期規則刪除物件時，原因欄位會設定為生命週期過期。如需詳細資訊，請參閱 即將到期的物件。 刪除未進行版本控制的物件，或永久刪除版本控制的物件時，deletion-type 欄位會設定為 Permanently Deleted (永久刪除)。當為版本控制的物件建立刪除標記時，deletion-type 欄位會設定為 Delete Marker Created (刪除建立的標記)。如需詳細資訊，請參閱 刪除啟用版本控制功能之儲存貯體中的物件。
還原物件已起始	從 S3 Glacier 或 S3 Glacier Deep Archive 儲存類別或者 S3 Intelligent-Tiering 封存存取或 Deep Archive 存取層起始物件還原。如需詳細資訊，請參閱 使用封存的物件。
物件還原已完成	物件還原已完成。

事件類型	描述
物件還原已過期	從 S3 Glacier 或 S3 Glacier Deep Archive 還原的物件臨時複本已過期且已刪除。
物件儲存類別已變更	物件已轉換至不同的儲存類別。如需詳細資訊，請參閱 使用 Amazon S3 生命週期轉換物件。
物件存取層已變更	物件轉換至 S3 Intelligent-Tiering Archive 存取層或 Deep Archive 存取層。如需詳細資訊，請參閱 Amazon S3 Intelligent Tiering。
物件 ACL 已更新	物件的存取控制清單 (ACL) 是使用 PutObject ACL 設定的。當請求對物件的 ACL 沒有變更時，則不會產生事件。如需詳細資訊，請參閱 存取控制清單 (ACL) 概觀。
物件標籤已新增	使用將一組標籤新增至物件 PutObjectTagging。如需詳細資訊，請參閱 使用標籤分類儲存空間。
物件標籤已刪除	使用從物件中移除所有標籤 DeleteObjectTagging。如需詳細資訊，請參閱 使用標籤分類儲存空間。

> **Note**
>
> 如需 Amazon S3 事件類型如何對應至 EventBridge 事件類型的詳細資訊，請參閱Amazon EventBridge 映射和故障診。

您可以 EventBridge 將 Amazon S3 事件通知與編寫規則，以便在儲存貯體中發生事件時採取動作。例如，您可以讓其為您傳送通知。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南 EventBridge中的內容。

如需有關您可以使用 EventBridge API 進行互動的動作和資料類型的詳細資訊，請參閱 Amazon EventBridge API 參考中的 Amazon EventBridge API 參考資料。

如需有關定價的資訊，請參閱 Amazon EventBridge 定價。

主題

- Amazon EventBridge 許可
- 啟用 Amazon EventBridge
- EventBridge 事件訊息結構
- Amazon EventBridge 映射和故障診

Amazon EventBridge 許可

Amazon S3 不需要任何其他許可即可將事件傳送到 Amazon EventBridge。

啟用 Amazon EventBridge

您可以 EventBridge 使用 S3 主控台 AWS Command Line Interface (AWS CLI) 或 Amazon S3 REST API 啟用亞馬遜。

使用 S3 主控台

在 S3 主控台中啟用 EventBridge 事件傳遞。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在儲存貯體名稱清單中，選擇要啟用事件的儲存貯體名稱。
3. 選擇 Properties (屬性)。
4. 導航到事件通知部分，找到 Amazon 子 EventBridge節。選擇編輯。
5. 在「將此值區中所有事件 EventBridge 的通知傳送至 Amazon」下方，選擇「開啟」。

> **Note**
>
> 啟用之後 EventBridge，變更大約需要五分鐘才會生效。

使用 AWS CLI

下列範例會為 EventBridge 啟用 Amazon 的儲存貯體DOC-EXAMPLE-BUCKET1建立值區通知組態。

```
aws s3api put-bucket-notification-configuration --bucket example-s3-bucket1 --
notification-configuration='{ "EventBridgeConfiguration": {} }'
```

使用 REST API

您可以呼叫 Amazon S3 REST API，以程式設計方式 EventBridge 在儲存貯體上啟用亞馬遜。如需詳細資訊，請參閱 Amazon 簡單儲存服務 API 參考PutBucketNotificationConfiguration中的。

下列範例顯示在 EventBridge 啟用 Amazon 的情況下，用於建立值區通知組態的 XML。

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <EventBridgeConfiguration>
  </EventBridgeConfiguration>
</NotificationConfiguration>
```

建立 EventBridge 規則

啟用後，您可以為某些任務建立 Amazon EventBridge 規則。例如，您可以在建立物件時傳送電子郵件通知。如需完整教學課程，請參閱 Amazon EventBridge 使用者指南中的教學：在建立 Amazon S3 物件時傳送通知。

EventBridge 事件訊息結構

Amazon S3 傳送以發佈事件的通知訊息為 JSON 格式。當 Amazon S3 向 Amazon 發送事件時 EventBridge，會出現以下字段。

- version (版本) – 所有事件目前為 0 (零)。
- ID – 為每個事件產生的版本 4 UUID。
- detail-type – 要傳送的事件類型。如需事件類型的清單，請參閱 使用 EventBridge。
- source (來源) – 識別產生事件的服務。
- account (帳戶) – 儲存貯體擁有者的 12 位數 AWS 帳戶 ID。
- time (時間) — 事件發生的時間。
- region (區域) – 識別儲存貯體的 AWS 區域 。
- resource (資源) — 包含儲存貯體之 Amazon Resource Name (ARN) 的 JSON 陣列。
- detail (詳細資訊) – 包含事件相關資訊的 JSON 物件。如需此欄位可以包含哪些項目的詳細資訊，請參閱 事件訊息詳細資訊欄位。

事件訊息結構範例

以下是一些可以傳送至 Amazon 的 Amazon S3 事件通知訊息的範例 EventBridge。

物件已建立

```
{
  "version": "0",
  "id": "17793124-05d4-b198-2fde-7ededc63b103",
  "detail-type": "Object Created",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "DOC-EXAMPLE-BUCKET1"
    },
    "object": {
      "key": "example-key",
      "size": 5,
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "IYV3p45BT0ac8hjHg1houSdS1a.Mro8e",
      "sequencer": "617f08299329d189"
    },
    "request-id": "N4N7GDK58NMKJ12R",
    "requester": "123456789012",
    "source-ip-address": "1.2.3.4",
    "reason": "PutObject"
  }
}
```

已刪除物件 (使用 DeleteObject)

```
{
  "version": "0",
  "id": "2ee9cc15-d022-99ea-1fb8-1b1bac4850f9",
  "detail-type": "Object Deleted",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
```

```
  "resources": [
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "DOC-EXAMPLE-BUCKET1"
    },
    "object": {
      "key": "example-key",
      "etag": "d41d8cd98f00b204e9800998ecf8427e",
      "version-id": "1QW9g1Z99LUNbvaaYVpW9xDlOLU.qxgF",
      "sequencer": "617f0837b476e463"
    },
    "request-id": "0BH729840619AG5K",
    "requester": "123456789012",
    "source-ip-address": "1.2.3.4",
    "reason": "DeleteObject",
    "deletion-type": "Delete Marker Created"
  }
}
```

物件已刪除 (使用生命週期過期)

```
{
  "version": "0",
  "id": "ad1de317-e409-eba2-9552-30113f8d88e3",
  "detail-type": "Object Deleted",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "DOC-EXAMPLE-BUCKET1"
    },
    "object": {
      "key": "example-key",
```

```
      "etag": "d41d8cd98f00b204e9800998ecf8427e",
      "version-id": "mtB0cV.jejK63XkRNceanNMC.qXPWLeK",
      "sequencer": "617b398000000000"
    },
    "request-id": "20EB74C14654DC47",
    "requester": "s3.amazonaws.com",
    "reason": "Lifecycle Expiration",
    "deletion-type": "Delete Marker Created"
  }
}
```

物件還原已完成

```
{
  "version": "0",
  "id": "6924de0d-13e2-6bbf-c0c1-b903b753565e",
  "detail-type": "Object Restore Completed",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "DOC-EXAMPLE-BUCKET1"
    },
    "object": {
      "key": "example-key",
      "size": 5,
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "KKsjUC1.6gIjqtvhfg5AdMI0eCePIiT3"
    },
    "request-id": "189F19CB7FB1B6A4",
    "requester": "s3.amazonaws.com",
    "restore-expiry-time": "2021-11-13T00:00:00Z",
    "source-storage-class": "GLACIER"
  }
}
```

事件訊息詳細資訊欄位

詳細資訊欄位包含具有事件相關資訊的 JSON 物件。下列欄位可能會出現在詳細資訊欄位中。

- version (版本) – 所有事件目前為 0 (零)。
- bucket (儲存貯體) – 與事件有關之 Amazon S3 儲存貯體的相關資訊。
- object (物件) – 與事件有關之 Amazon S3 物件的相關資訊。
- request-id – S3 回應中的請求 ID。
- 請求者 — 請求者的 AWS 帳戶 ID 或 AWS 服務主體。
- source-ip-address— S3 請求的源 IP 地址。僅適用於由 S3 請求觸發的事件。
- reason — 針對物件建立事件，用來建立物件的 S3 API：PutObjectCopyObject、POST 物件或 CompleteMultipartUpload. 對於「已刪除物件」事件，此值會設定為由 S3 API 呼叫刪除物件時，或在 S3 生命週期到期規則刪除物件時生命週期到期。DeleteObject如需詳細資訊，請參閱 即將到期的物件。
- deletion-type – 對於 Object Deleted (物件已刪除) 事件，刪除未進行版本控制的物件時，或永久刪除版本控制的物件時，會設定為 Permanently Deleted (永久刪除)。當為版本控制物件建立刪除標記時，會設定為 Delete Marker Created (刪除建立的標記)。如需詳細資訊，請參閱 刪除啟用版本控制功能之儲存貯體中的物件。
- restore-expiry-time— 針對「物件還原已完成」事件，將從 S3 刪除物件暫時複本的時間。如需詳細資訊，請參閱 使用封存的物件。
- source-storage-class— 針對「已啟動物件還原」和「物件還原完成」事件，則為要還原之物件的儲存類別。如需詳細資訊，請參閱 使用封存的物件。
- destination-storage-class— 對於「物件儲存類別已變更」事件，則為物件的新儲存類別。如需詳細資訊，請參閱 使用 Amazon S3 生命週期轉換物件。
- destination-access-tier— 對於「物件存取層已變更」事件，即物件的新存取層。如需詳細資訊，請參閱 Amazon S3 Intelligent Tiering。

Amazon EventBridge 映射和故障診

下表說明 Amazon S3 事件類型如何對應至 Amazon EventBridge 事件類型。

S3 事件類型	Amazon EventBridge 詳細類型
ObjectCreated：放	物件已建立

S3 事件類型	Amazon EventBridge 詳細類型
ObjectCreated: 帖子 ObjectCreated: 複製 ObjectCreated:CompleteMultipartUpload	
ObjectRemoved: 刪除 ObjectRemoved:DeleteMarkerCreated LifecycleExpiration: 刪除 LifecycleExpiration:DeleteMarkerCreated	物件已刪除
ObjectRestore: 帖子	還原物件已起始
ObjectRestore: 已完成	物件還原已完成
ObjectRestore: 刪除	物件還原已過期
LifecycleTransition	物件儲存類別已變更
IntelligentTiering	物件存取層已變更
ObjectTagging：放	物件標籤已新增
ObjectTagging: 刪除	物件標籤已刪除
ObjectAcl：放	物件 ACL 已更新

Amazon EventBridge 排除

如需有關如何疑難排解的資訊 EventBridge，請參閱 Amazon EventBridge 使用者指南 EventBridge中的疑難排解 Amazon。

使用分析與洞見

您可以在 Amazon S3 中使用分析和洞見功能來了解、分析和最佳化儲存使用量。如需詳細資訊，請參閱下列主題。

主題

Amazon S3 分析 – 儲存類別分析

您可以使用 Amazon S3 分析「儲存類別分析」來分析儲存體存取模式，協助您決定何時將正確的資料轉移至正確的儲存類別。這個新的 Amazon S3 分析功能會觀察資料存取模式，協助您決定何時將不常存取的 STANDARD 儲存，轉移至 STANDARD_IA (IA 表示不常存取) 儲存類別。如需儲存體方案的詳細資訊，請參閱「使用 Amazon S3 儲存體方案」。

在儲存體方案分析觀察一段時間之已篩選資料集的不常存取模式後，您可以使用分析結果協助改善生命週期組態。您可以設定儲存體方案分析，分析儲存貯體中的所有物件。或者，您可以設定篩選條件，依共同的字首 (亦即，名稱使用共同字串開頭的物件)、物件標籤或這兩者來分組物件進行分析。您一定會發現，依物件群組篩選，是最能發揮儲存體方案分析優勢的方法。

> **Important**
>
> 儲存類別分析僅提供標準到標準 IA 類別的建議。

您的每一個儲存貯體最多可有 1,000 個儲存體方案分析，而每項篩選條件都會得到不同的分析。多篩選條件的組態可讓您分析特定的物件群組，從而改進將物件轉換為 STANDARD_IA 的生命週期組態。

儲存類別分析會在 Amazon S3 主控台中，提供每日更新的儲存用量視覺化。您也可以將這些每日用量資料匯出到 S3 儲存貯體，並在試算表應用程式中檢視這些資料，或使用 Amazon 等商業智慧工具進行檢視 QuickSight。

有與儲存類別分析相關的成本。如需定價資訊，請參閱「管理與複寫」Amazon S3 定價。

主題

如何設定儲存體方案分析？

您可以藉由設定您要分析的物件資料來設定儲存體方案分析。設定儲存體方案分析可以執行下列作業：

- 分析儲存貯體的全部內容。

  您會收到儲存貯體中所有物件的分析。
- 分析依前綴及標籤分組的物件。

  您可以設定篩選條件，依字首、物件標籤或這兩者來分組物件進行分析。您設定的每項篩選條件都會收到不同的分析。您的每一個儲存貯體最多可有 1,000 個篩選條件組態。
- 匯出分析資料。

  當您設定儲存貯體或篩選條件的儲存體方案分析時，可以選擇每天將分析資料匯出至檔案。當日的分析會新增到檔案中，成為所設定之篩選條件的歷史分析日誌。此檔案在所選的目標會每日更新。在選取要匯出的資料時，您必須指定檔案寫入的目標儲存貯體及選用的目標字首。

您可以使用 Amazon S3 主控台、REST API 或 AWS CLI 或 AWS 開發套件來設定儲存類別分析。

- 如需有關如何在 Amazon S3 主控台中設定儲存類別分析的資訊，請參閱「設定儲存類別分析」。
- 若要使用 Amazon S3 API，請使用或 AWS 開發套件中的PutBucketAnalyticsConfiguration其他 API AWS CLI 或同等的 API。

如何使用儲存體方案分析？

您可以使用儲存體方案分析，觀察一段時間的資料存取模式從中收集資訊，藉此改善 STANDARD_IA 儲存體的生命週期管理。在設定篩選條件之後的 24 到 48 小時內，您在 Amazon S3 主控台中會開始看到根據篩選條件的資料分析。但儲存體方案分析會觀察篩選所得之資料集的存取模式 30 天或更久，先收集分析資訊，然後再提供結果。分析在得到第一次的結果之後會繼續進行，並隨著存取模模式變更而更新結果。

第一次設定篩選條件時，Amazon S3 主控台可能需要一點時間來分析您的資料。

儲存體方案分析會觀察篩選所得之資料集的存取模式 30 天或久，以收集足夠的資訊進行分析。當儲存類別分析收集到足夠的資訊後，您在 Amazon S3 主控台中會看到分析完成的訊息。

在對不常存取的物件執行分析時，儲存類別分析會觀察篩選後的物件集，這些物件是根據上傳到 Amazon S3 後的存留期而組合在一起。儲存體方案分析會依據下列因素，觀察篩選後的資料集，判斷存留期群組是否不常存取：

- STANDARD 儲存體方案中的物件超過 128 KB。
- 您的每一個存留期群組之平均儲存體總量。
- 每個存留期群組傳出的平均位元組數 (非經常性)。
- Analytics 的匯出資料只包含對儲存體方案分析相關資料的要求。這可能會造成要求數、上傳總計及要求的位元組數，與儲存體指標或您內部系統追蹤所顯示數據有所差異。
- 失敗的 GET 及 PUT 要求不會計入分析。但您仍會在儲存指標中看到失敗的要求。

我擷取了多少儲存體？

Amazon S3 主控台會以圖表顯示在觀察期間，已從篩選後的資料集擷取多少儲存體。

我擷取了多少百分比的儲存體？

Amazon S3 主控台也會以圖表顯示在觀察期間，已從篩選後的資料集擷取多少百分比的儲存體。

如本主題前文所述，在對不常存取的物件執行分析時，儲存類別分析會觀察篩選後的物件集，這些物件是根據上傳到 Amazon S3 後的存留期而組合在一起。儲存體方案分析使用下列預先定義的物件存留期群組：

- 不及 15 天的 Amazon S3 物件
- 15-29 天的 Amazon S3 物件
- 30-44 天的 Amazon S3 物件
- 45-59 天的 Amazon S3 物件
- 60-74 天的 Amazon S3 物件
- 75-89 天的 Amazon S3 物件
- 90-119 天的 Amazon S3 物件
- 120-149 天的 Amazon S3 物件
- 150-179 天的 Amazon S3 物件
- 180-364 天的 Amazon S3 物件

- 365-729 天的 Amazon S3 物件
- 超過 730 天 (含) 的 Amazon S3 物件

觀察存取模式一般大約需要 30 天，才能收集到足夠的資訊取得分析結果。此期間也可能會超過 30 天，視資料的特有的存取模式而定。然而，在設定篩選條件之後的 24 到 48 小時內，您在 Amazon S3 主控台中會開始看到根據篩選條件的資料分析。在 Amazon S3 主控台中，您可以看到依物件存留期群組分組的每日物件存取分析。

我的儲存體中不常存取的部分有多少？

Amazon S3 主控台會顯示存取模式，並依預先定義的物件存留期群組分組。顯示的 Frequently accessed (經常存取) 或 Infrequently accessed (不常存取) 文字是作為協助您完成生命週期建立程序的視覺輔助。

如何匯出儲存體方案分析資料？

您可以選擇將儲存體方案分析的分析報告匯出為逗號分隔值 (CSV) 的一般檔案。報告會每日更新，並依據您設定的物件存留期群組加以篩選。使用 Amazon S3 主控台建立篩選條件時，您可以選擇匯出報告選項。選取資料匯出時，必須指定寫入檔案的目標儲存貯體，並選擇是否要指定目標字首。您可以將資料匯出到其他帳戶中的目標儲存貯體。目標儲存貯體與您設定所要分析的儲存貯體，必須位在相同的區域。

您必須在目的地儲存貯體上建立儲存貯體政策，以授予 Amazon S3 許可，以驗證 AWS 帳戶 擁有儲存貯體的物件，並將物件寫入定義位置的儲存貯體。如需政策範例，請參閱「授予 S3 清查與 S3 分析的許可」。

當您設定儲存體方案分析報告 24 小時後，就會開始每天收到匯出的報告。之後，Amazon S3 會持續監視並提供每日的匯出。

您可以在試算表應用程式中開啟 CSV 檔案，或將檔案匯入其他應用程式 (例如 Amazon) QuickSight。如需搭配 Amazon 使用 Amazon S3 檔案的相關資訊 QuickSight，請參閱 Amazon 使用 QuickSight 者指南中的使用 Amazon S3 檔案建立資料集。

檔案匯出中的資料在物件存留期群組中會依日期儲存，如下列範例所示。若儲存體方案是 STANDARD，資料列也會包含 `ObjectAgeForSIATransition` 及 `RecommendedObjectAgeForSIATransition` 資料行的資料。

Date	ConfigId	Filter	StorageClass	ObjectAge	ObjectCount	DataUploaded_MB	Storage_MB	DataRetrieved_MB	GetRequestCount	CumulativeAccessRatio	ObjectAgeForSIATransition	RecommendedObjectAgeForSIATransition
8/17/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/2/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/22/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
8/25/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/6/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/30/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/28/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/21/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/5/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		

報告結尾的物件存留期群組指定為 ALL。ALL 資料列包含該天所有存留期群組的累加總計，包括小於 128 KB 的物件。

8/24/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0	000-014
9/3/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0.02426125	015-029
8/28/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0.03545875	015-029
8/17/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0	000-014
8/25/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0	000-014
9/6/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0.0209529	015-029
9/4/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0.02304819	015-029
8/22/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0	000-014
8/21/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0	000-014
8/30/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0.03073092	015-029
8/20/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0	000-014

下節說明報告中使用的資料行。

匯出的檔案配置

下表說明匯出之檔案的配置。

設定儲存類別分析

使用 Amazon S3 Analytics 儲存類別分析工具，可以分析儲存體存取模式，協助您決定何時將正確的資料轉移到正確的儲存類別。儲存體方案分析會觀察資料存取模式，協助您判斷何時將不常存取的標準型儲存體轉移至標準型 (IA) (IA 表示不常存取) 儲存體方案。如需 STANDARD_IA 的詳細資訊，請參閱 Amazon S3 常見問題集 和 使用 Amazon S3 儲存體方案。

您可以藉由設定您要分析的物件資料來設定儲存體方案分析。設定儲存體方案分析可以執行下列作業：

- 分析儲存貯體的全部內容。

  您會收到儲存貯體中所有物件的分析。
- 分析依前綴及標籤分組的物件。

  您可以設定篩選條件，依字首、物件標籤或這兩者來分組物件進行分析。您設定的每項篩選條件都會收到不同的分析。您的每一個儲存貯體最多可有 1,000 個篩選條件組態。
- 匯出分析資料。

  當您設定儲存貯體或篩選條件的儲存體方案分析時，可以選擇每天將分析資料匯出至檔案。當日的分析會新增到檔案中，成為所設定之篩選條件的歷史分析日誌。此檔案在所選的目標會每日更新。在選取要匯出的資料時，您必須指定檔案寫入的目標儲存貯體及選用的目標字首。

您可以使用 Amazon S3 主控台、REST API 或 AWS CLI 或 AWS 開發套件來設定儲存類別分析。

> **Important**
>
> 儲存類別分析不建議轉移至 ONEZONE_IA 或 S3 Glacier Flexible Retrieval 儲存類別。如果您想要設定儲存類別分析以將發現項目匯出為 .csv 檔案，而目的地儲存貯體使用預設儲存貯體加密搭配 AWS KMS key，則必須更新 AWS KMS 金鑰政策以授與 Amazon S3 加密 .csv 檔案的權限。如需說明，請參閱「對 Amazon S3 授予許可使用您的客戶受管金鑰進行加密」。

如需分析的詳細資訊，請參閱「Amazon S3 分析 – 儲存類別分析」。

使用 S3 主控台

設定儲存體方案分析

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體) 清單中，選擇您要配置儲存方案分析的儲存貯體名稱。
3. 選擇 Metrics (指標) 標籤。
4. 在 Storage Class Analysis (儲存類別分析) 下，選擇 Create analytics configuration (建立分析組態)。
5. 輸入篩選條件的名稱。如果您希望分析整個儲存貯體，請將 Prefix (字首) 欄位留白。
6. 在 Prefix (字首) 欄位中，輸入您要分析之物件的前綴文字。
7. 若要新增標籤，請選擇 Add tag (新增標籤)。為標籤輸入金鑰和值。您可以輸入一個字首和多個標籤。
8. 或者，您可以也選擇匯出 CSV 下的啟用，將分析報告匯出為逗號分隔值 (.csv) 一般檔案。選擇可存放檔案的目的地儲存貯體。您可以輸入目的地儲存貯體的字首。目的地值區必須與您 AWS 區域要設定分析的值區相同。目的地儲存貯體可位於不同的 AWS 帳戶中。

   如果 .csv 檔案的目標儲存貯體使用預設儲存貯體加密搭配 KMS 金鑰，您必須更新金 AWS KMS 鑰政策以授與 Amazon S3 加密 .csv 檔案的權限。如需說明，請參閱「對 Amazon S3 授予許可使用您的客戶受管金鑰進行加密」。
9. 選擇 Create Configuration (建立組態)。

Amazon S3 會在目的地儲存貯體建立儲存貯體政策，將寫入許可授予 Amazon S3。這將允許它將導出數據寫入存儲桶。

如果在嘗試建立儲存貯體政策時發生錯誤，則會提供其修正說明。例如，如果您在另一個 AWS 帳戶中選擇目的地儲存貯體，而且沒有讀取與寫入儲存貯體政策的許可，則會看到下列訊息。您必須是目的地儲存貯體擁有者，才能將顯示的儲存貯體政策新增至目的地儲存貯體。如果原則未新增至目的地儲存貯體，則無法取得匯出資料，因為 Amazon S3 沒有寫入目的地儲存貯體的許可。如果是由與目前使用者不同的帳戶擁有來源儲存貯體，則必須替換政策中來源儲存貯體的正確帳戶 ID。

如需匯出資料及篩選條件如何運作的資訊，請參閱「Amazon S3 分析 – 儲存類別分析」。

使用 REST API

若要使用 REST API 設定儲存類別分析，請使用 PutBucketAnalyticsConfiguration. 您也可以將等效作業與 AWS CLI 或 AWS SDK 搭配使用。

您可以使用下列 REST API 來進行儲存類別分析：

- DELETE 儲存貯體分析組態
- GET 儲存貯體分析組態
- List 儲存貯體分析組態

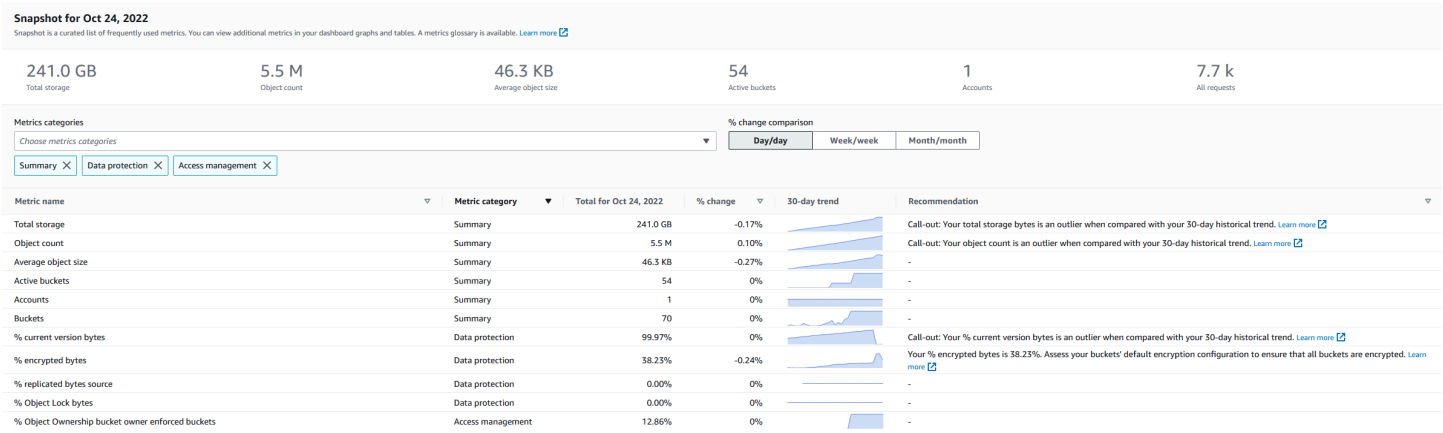
使用 Amazon S3 Storage Lens 評估儲存活動和使用量

Amazon S3 Storage Lens 是一項雲端儲存體分析功能，可用來讓整個組織了解物件儲存體和活動。S3 Storage Lens 也會分析指標，以提供內容相關建議，您可以用來最佳化儲存成本，並套用最佳實務保護您的資料。

您可以使用 S3 Storage Lens 指標產生摘要洞察。例如，您可以了解您在整個組織中擁有多少儲存體，或是成長速度最快的儲存貯體和字首為何。您也可以使用 S3 Storage Lens 指標，識別成本最佳化機會、實作資料保護和存取管理最佳實務，以及改善應用程式工作負載的效能。例如，您可以識別沒有 S3 生命週期規則的儲存貯體，這些規則可中止超過 7 天未完成的分段上傳。您也可以識別未遵循資料保護最佳實務的儲存貯體，例如使用 S3 複寫或 S3 版本控制。

S3 Storage Lens 會彙總您的指標，並在 Amazon S3 主控台 Buckets (儲存貯體) 頁面上的 Account snapshot (帳戶快照) 區段中顯示資訊。S3 Storage Lens 也會提供互動式儀表板，您可以用來以視覺化方式呈現洞見與趨勢、標記異常值，以及接收最佳化儲存成本和套用資料保護最佳實務的建議。您的儀表板具有向下切入選項可產生及視覺化組織、帳戶、AWS 區域、儲存類別、儲存貯體、字首或

Storage Lens 群組層級的洞察。您也可以將 CSV 或 Parquet 格式的每日指標匯出傳送到 S3 儲存貯體。



S3 Storage Lens 指標和功能

S3 Storage Lens 會提供每日更新的互動式預設儀表板。S3 Storage Lens 會預先設定此儀表板，以視覺化整個帳戶的摘要洞見和趨勢，並在 Amazon S3 主控台中每日更新它們。此儀表板中的指標也會 Buckets (儲存貯體) 頁面的帳戶快照中進行彙總。如需詳細資訊，請參閱「預設儀表板」。

若要建立其他儀表板，並依 AWS 區域、S3 儲存貯體或帳戶 (適用於 AWS Organizations) 限定其範圍，請建立 S3 Storage Lens 儀表板組態。您可以使用 Amazon S3 主控台、AWS Command Line Interface、(AWS CLI)、AWS SDK 或 Amazon S3 REST API，建立和管理 S3 Storage Lens 儀表板組態。建立或編輯 S3 Storage Lens 儀表板時，您可以定義儀表板範圍和指標選擇。

S3 Storage Lens 提供免費指標，以及您可以升級至其中，但需額外付費的進階指標和建議。使用進階指標和建議，您可以存取其他指標和功能，以洞察您的儲存體。這些功能包括進階字首類別、字首彙總、內容相關建議，以及 Amazon CloudWatch 發佈。字首彙總和內容相關建議只能在 Amazon S3 主控台中使用。如需有關 S3 Storage Lens 定價的詳細資訊，請參閱 Amazon S3 定價。

指標類別

在免費和進階方案中，指標會組織成符合重要使用案例的類別，例如成本最佳化和資料保護。免費指標包括摘要、成本最佳化、資料保護、存取管理、效能和事件指標。升級至進階指標和建議時，您可以啟用進階成本最佳化和資料保護指標。您可以使用這些進階指標，進一步降低 S3 儲存成本並改善資料保護立場。您也可以啟用活動指標和詳細狀態碼指標，以改善正在存取 S3 儲存貯體之應用程式工作負載的效能。如需免費和進階指標類別的詳細資訊，請參閱 指標選擇。

您可以根據 S3 最佳實務評估儲存體，例如分析已啟用加密或 S3 物件鎖定或 S3 版本控制的儲存貯體百分比。您也可以識別潛在的成本節省機會。例如，您可以使用 S3 生命週期規則計數指標，來識別哪

些儲存貯體缺少生命週期過期或轉移規則。您也可以分析每個儲存貯體的請求活動，尋找可以將其物件移動到成本較低之儲存體類別的儲存貯體。如需詳細資訊，請參閱「Amazon S3 Storage Lens 指標使用案例」。

指標匯出

除了檢視 S3 主控台上的儀表板外，您還可以透過 CSV 或 Parquet 格式，將指標匯出到 S3 儲存貯體，以使用您選擇的分析工具執行進一步分析。如需詳細資訊，請參閱「使用資料匯出檢視 Amazon S3 Storage Lens 指標」。

Amazon CloudWatch 發佈

您可以將 S3 Storage Lens 用量和活動指標發佈到 Amazon CloudWatch，以便在 CloudWatch 儀表板中建立統一的運作狀態檢視。您也可以使用 CloudWatch 功能 (例如警示和觸發動作、指標數學和異常偵測) 來監控 S3 Storage Lens 指標並對其採取動作。此外，CloudWatch API 操作可讓應用程式 (包括第三方供應商) 存取您的 S3 Storage Lens 指標。CloudWatch 發佈選項適用於升級至 S3 Storage Lens 進階指標和建議的儀表板。如需有關 CloudWatch 中 S3 Storage Lens 指標的詳細資訊，請參閱監控 CloudWatch 中的 S3 Storage Lens 指標。

如需使用 S3 Storage Lens 的詳細資訊，請參閱下列主題。

主題

了解 Amazon S3 Storage Lens

> **Important**
>
> Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。從 2023 年 1 月 5 日起，所有上傳到 Amazon S3 的新物件都會

自動加密，無需額外費用，也不會影響效能。S3 儲存貯體預設加密組態和新物件上傳的自動加密狀態會顯示於 AWS CloudTrail 日誌、S3 清查、S3 Storage Lens、Amazon S3 主控台，並做為 AWS Command Line Interface 和 AWS SDK 的其他 Amazon S3 API 回應標頭。如需詳細資訊，請參閱預設加密常見問答集。

Amazon S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。您可以使用 S3 Storage Lens 指標產生摘要洞見，例如了解您在整個組織中擁有多少儲存體，或是成長速度最快的儲存貯體和字首有哪些。您也可以使用 S3 Storage Lens 指標，識別成本最佳化機會、實作資料保護和安全最佳實務，以及改善應用程式工作負載的效能。例如，您可以識別沒有 S3 生命週期規則的儲存貯體，這些規則可使超過 7 天未完成的分段上傳過期。您也可以識別未遵循資料保護最佳實務的儲存貯體，例如使用 S3 複寫或 S3 版本控制。S3 Storage Lens 也會分析指標，以提供內容相關建議，您可以用來最佳化儲存成本，並套用最佳實務保護您的資料。

S3 Storage Lens 會彙總您的指標，並在 Amazon S3 主控台 Buckets (儲存貯體) 頁面上的 Account snapshot (帳戶快照) 區段中顯示資訊。S3 Storage Lens 也會提供互動式儀表板，您可以用來以視覺化方式呈現洞見與趨勢、標記異常值，以及接收最佳化儲存成本和套用資料保護最佳實務的建議。您的儀表板具有向下切入選項可產生及視覺化組織、帳戶、AWS 區域、儲存類別、儲存貯體、字首或 Storage Lens 群組層級的洞察。您也可以將 CSV 或 Parquet 格式的每日指標匯出傳送到 S3 儲存貯體。您可以使用 Amazon S3 主控台、AWS Command Line Interface (AWS CLI)、AWS SDK 或 Amazon S3 REST API，建立和管理 S3 Storage Lens 儀表板組態。

S3 Storage Lens 概念和術語

本節包含成功了解和使用 Amazon S3 Storage Lens 須知的術語和概念。

主題

- 儀表板組態
- 預設儀表板
- 儀表板
- 帳戶快照
- 指標匯出
- 主要區域
- 保留期間
- 指標類別

- 建議
- 指標選擇
- S3 Storage Lens 和 AWS Organizations

儀表板組態

S3 Storage Lens 需要一個儀表板組態，其中包含代表您為單一儀表板或匯出彙總指標所需的屬性。建立組態時，您可以選擇儀表板名稱和主要區域，但在建立儀表板之後便無法將其變更。您可以選擇性地新增標籤，並設定 CSV 或 Parquet 格式的指標匯出。

在儀表板組態中，您也可以定義儀表板範圍和指標選擇。範圍可以包括組織帳戶的所有儲存體，或依區域、儲存貯體和帳戶篩選的區段。設定指標選擇時，您可以在免費指標與進階指標和建議之間進行選擇。您可以升級至進階指標和建議，但需額外付費。使用進階指標和建議，您可以存取其他指標和功能。這些功能包括進階字首類別、字首層級彙總、內容相關建議，以及 Amazon CloudWatch 發佈。如需有關 S3 Storage Lens 定價的詳細資訊，請參閱 Amazon S3 定價。

預設儀表板

主控台上的 S3 Storage Lens 預設儀表板名稱為 default-account-dashboard。S3 會預先設定此儀表板，以視覺化整個帳戶的摘要洞見和趨勢，並在 Amazon S3 主控台中每日更新它們。您無法修改預設儀表板的設定範圍，但可以將指標選項從免費指標升級至進階指標和建議。您可以設定選用指標匯出，甚至停用儀表板。但是，您無法刪除預設儀表板。

> **Note**
>
> 如果停用預設儀表板，則不再更新它。您將不再於 S3 Storage Lens 儀表板、您的指標匯出或於 S3 儲存貯體頁面上的帳戶快照中，收到任何新的每日指標。如果您的儀表板使用進階指標和建議，則不會再向您收費。您仍然可在儀表板中查看歷史資料，直到資料查詢的 14 天期間過期為止。如果您已啟用進階指標和建議，則此期間為 15 個月。若要存取歷史資料，您可以在期限內重新啟用儀表板。

儀表板

您可以建立其他 S3 Storage Lens 儀表板，並依 AWS 區域、S3 儲存貯體或帳戶 (適用於 AWS Organizations) 限定其範圍。建立或編輯 S3 Storage Lens 儀表板時，您可以定義儀表板範圍和指標選擇。S3 Storage Lens 提供免費指標，以及您可以升級至其中，但需額外付費的進階指標和建議。使用進階指標和建議，您可以存取其他指標和功能，以洞察您的儲存體。其中包括進階字首類別、字首層級

彙總、內容相關建議，以及 Amazon CloudWatch 發佈。如需有關 S3 Storage Lens 定價的詳細資訊，請參閱 Amazon S3 定價。

您也可以停用或刪除儀表板。如果停用儀表板，該儀表板將不再更新，而且您將不再收到任何新的每日指標。您仍然可以查看歷史資料，直到 14 天期間過期為止。如果您已針對該儀表板啟用進階指標和建議，則此期間為 15 個月。若要存取歷史資料，您可以在期限內重新啟用儀表板。

如果刪除儀表板，則會遺失所有儀表板組態設定。您將不再收到任何新的每日指標，也無法存取與該儀表板相關聯的歷史資料。如果想要存取已刪除儀表板的歷史資料，必須在同一個主要區域中建立另一個具有相同名稱的儀表板。

> **Note**
>
> - 您可以使用 S3 Storage Lens，在每個主要區域建立多達 50 個儀表板。
> - 組織層級儀表板只能限於區域範圍。

帳戶快照

S3 Storage Lens Account snapshot (帳戶快照) 會彙總來自預設儀表板的指標，並在 S3 主控台 Buckets (儲存貯體) 頁面上，顯示您的儲存體總量、物件數量和平均物件大小。此帳戶快照可讓您快速存取有關儲存體的洞見，而不必離開 Buckets (儲存貯體) 頁面。帳戶快照也可讓您一鍵式存取您的互動式 S3 Storage Lens 儀表板。

您可以使用儀表板，來視覺化洞見與趨勢、標記異常值，以及接收最佳化儲存成本並套用資料保護最佳實務的建議。您的儀表板具有深度切入選項來產生組織、帳戶、儲存貯體、物件或字首層級的洞見。您也可以透過 CSV 或 Parquet 格式將每日一次指標匯出傳送至 S3 儲存貯體。

您無法修改 default-account dashboard (預設帳戶儀表板) 的儀表板範圍，因為它已連結至 Account snapshot (帳戶快照)。不過，您可以將 default-account-dashboard 中的指標選擇從免費指標升級至付費進階指標和建議。升級後，您可以在 S3 Storage Lens Account snapshot (帳戶快照) 中顯示所有請求、上傳的位元組和下載的位元組。

> **Note**
>
> 如果您停用預設儀表板，您的 Account snapshot (帳戶快照) 不再更新。若要繼續在 Account snapshot (帳戶快照) 中顯示指標，您可以重新啟用 default-account-dashboard。

指標匯出

S3 Storage Lens 指標匯出是一個檔案，其中包含 S3 Storage Lens 組態中識別的所有指標。系統會每日產生 CSV 或 Parquet 格式的資訊，並將其傳送至 S3 儲存貯體。您可以使用您選擇的指標工具，將指標匯出用於進一步分析。匯出指標的 S3 儲存貯體必須與 S3 Storage Lens 組態位於相同的區域。您可以編輯儀表板組態，從 S3 主控台產生 S3 Storage Lens 指標匯出。您也可以使用 AWS CLI 和 AWS SDK 設定指標匯出。

主要區域

主要區域是 AWS 區域，其中存放指定儀表板或組態的所有 S3 Storage Lens 指標。建立 S3 Storage Lens 儀表板組態時，必須選擇主要區域。選擇主要地區後，您無法將其變更。同樣地，如果您要建立 Storage Lens 群組，建議您選擇與 Storage Lens 儀表板相同的主要區域。

> **Note**
>
> 您可以選擇下列其中一個區域作為您的主要區域：
>
> - 美國東部 (維吉尼亞北部) – `us-east-1`
> - 美國東部 (俄亥俄) – `us-east-2`
> - 美國西部 (加利佛尼亞北部) – `us-west-1`
> - 美國西部 (奧勒岡) – `us-west-2`
> - 亞太區域 (孟買) – `ap-south-1`
> - 亞太區域 (首爾) – `ap-northeast-2`
> - 亞太區域 (新加坡) – `ap-southeast-1`
> - 亞太區域 (雪梨) – `ap-southeast-2`
> - 亞太區域 (東京) – `ap-northeast-1`
> - 加拿大 (中部) – `ca-central-1`
> - 中國 (北京) - `cn-north-1`
> - 中國 (寧夏) - `cn-northwest-1`
> - 歐洲 (法蘭克福) – `eu-central-1`
> - 歐洲 (愛爾蘭) – `eu-west-1`
> - 歐洲 (倫敦) – `eu-west-2`
> - 歐洲 (巴黎) – `eu-west-3`
> - 歐洲 (斯德哥爾摩) – `eu-north-1`

- 南美洲 (聖保羅) – `sa-east-1`

保留期間

S3 Storage Lens 指標會保留，因此您可以查看歷史趨勢，並比較一段時間內儲存和活動的差異。您可以將 Amazon S3 Storage Lens 指標用於查詢，因此您可以查看歷史趨勢，並比較一段時間內儲存用量和活動的差異。

所有 S3 Storage Lens 指標都會保留 15 個月。不過，指標僅適用於特定持續時間的查詢，這取決於您的指標選擇。此持續時間無法修改。免費指標有 14 天的時間可用於查詢，進階指標有 15 個月的時間可用於查詢。

指標類別

在免費和進階方案中，S3 Storage Lens 指標會組織成符合重要使用案例的類別，例如成本最佳化和資料保護。免費指標包括摘要、成本最佳化、資料保護、存取管理、效能和事件指標。升級至進階指標和建議時，您可以啟用其他成本最佳化和資料保護指標，用來進一步降低 S3 儲存成本，並確保資料受到保護。您也可以啟用活動指標和詳細狀態碼指標，其可以用來改善應用程式工作負載的效能。

下列清單顯示所有免費和進階指標類別。如需每個類別中包含的個別指標的完整清單，請參閱指標詞彙表。

摘要指標

摘要指標提供 S3 儲存體的一般洞見，包括總儲存位元組和物件計數。

成本最佳化指標

成本最佳化指標可以提供您可以用來管理和最佳化儲存成本的洞見。例如，您可以識別哪些儲存貯體具有超過 7 天未完成的分段上傳。

使用進階指標和建議，您可以啟用進階成本最佳化指標。這些指標包括 S3 生命週期規則計數指標，您可以用來取得每個儲存貯體過期和轉移 S3 生命週期規則計數。

資料保護指標

資料保護指標提供資料保護功能的洞見，例如加密和 S3 版本控制。您可以使用這些指標來識別未遵循資料保護最佳實務的儲存貯體。例如，您可以識別哪些儲存貯體未搭配 AWS Key Management Service 金鑰 (SSE-KMS) 或 S3 版本控制使用預設加密。

使用進階指標和建議，您可以啟用進階資料保護指標。這些指標包括每個儲存貯體複寫規則計數指標。

存取管理指標

存取管理指標提供 S3 物件擁有權的洞見。您可以使用這些指標來查看儲存貯體使用哪些物件擁有權設定。

事件指標

事件指標提供 S3 事件通知的洞見。使用事件指標，您可以查看哪些儲存貯體已設定 S3 事件通知。

表現指標

效能指標提供 S3 Transfer Acceleration 的洞見。使用效能指標，您可以查看哪些儲存貯體已啟用 Transfer Acceleration。

活動指標 (進階)

如果您將儀表板升級至進階指標和建議，則可以啟用活動指標。活動指標提供如何請求儲存體 (例如，所有請求、Get 請求、Put 請求) 的詳細資料、上傳或下載的位元組，以及錯誤。

字首層級活動指標可用來協助您判斷哪些字首不常使用，以便您使用 S3 生命週期轉換為更好的儲存類別。

詳細狀態碼指標 (進階)

如果您將儀表板升級至進階指標和建議，則可以啟用詳細狀態碼指標。詳細狀態碼指標提供 HTTP 狀態碼的洞察，例如 403 禁止和 503 服務無法使用，可用來對存取或效能問題進行疑難排解。例如，您可以查看 403 Forbidden error count (403 禁止錯誤計數) 指標，以識別在未套用正確許可的情況下存取工作負載的工作負載。

字首層級的詳細狀態碼指標可用來依字首進一步了解出現的 HTTP 狀態碼。例如，503 錯誤計數指標可讓您識別在資料擷取期間接收到調節請求的字首。

建議

S3 Storage Lens 提供自動化建議，協助您將儲存空間最佳化。系統會依上下文在 S3 Storage Lens 儀表板中的相關指標旁放置建議。建議與近期發生的事情相關，因此歷史資料不符合提供建議的資格。建議只有在具有相關性時才會顯示。

S3 Storage Lens 建議的形式如下：

- 建議

  建議會提醒您儲存和活動內的趨勢，這些趨勢可能指出儲存成本最佳化機會或資料保護最佳實務。您可以使用《Amazon S3 使用者指南》和 S3 Storage Lens 儀表板中的建議主題，深入了解特定區域、儲存貯體或字首的詳細資訊。

- 標註

  標註是一種建議，會警示您在一段時間內儲存和活動中可能需要進一步關注或監控的異常情況。

  - 異常值標註

    S3 Storage Lens 會根據您最近的 30 天的趨勢，提供異常值的指標標註。異常值是使用標準分數 (也稱為 z 分數) 計算。在此分數中，會從該指標過去 30 天的平均值中減去當天的指標。然後，會將當天的指標除以該指標在過去 30 天的標準差。產生的分數通常介於 -3 和 +3 之間。此數字代表當日指標與平均值的標準差數目。

    S3 Storage Lens 會將分數 >2 或 <-2 的指標視為異常值，因為它們高於或低於正常分佈資料的 95%。

  - 重大變更標註

    重大變更標註適用於預期變更頻率較低的指標。因此，此標註會設定比異常值計算更高的敏感度，與前一天、前一週或前一月相比，通常會落在 +/-20% 的範圍內。

    處理儲存和活動中的標註 – 如果您收到重大變更標註，這不一定是問題。標註可能是儲存中預期會有變更的結果。例如，您最近可能新增了大量新物件、刪除了大量物件，或進行了類似的計劃變更。

    如果您在儀表板上看到重大變更標註，請將它記下來，並判斷最近的情況是否可解釋此現象。如果無法解釋此現象，請使用 S3 Storage Lens 儀表板，深入了解更多詳細資訊，以了解造成波動的特定區域、儲存貯體或前綴。

- 提醒

  提醒提供有關 Amazon S3 如何運作的深入洞見。它們可協助您進一步了解如何使用 S3 功能，來降低儲存成本或套用資料保護最佳實務。

指標選擇

S3 Storage Lens 提供兩種您可以為儀表板和匯出選擇的指標選項：免費指標和進階指標和建議。

- 免費指標

S3 Storage Lens 為所有儀表板和組態提供免費指標。免費指標包含與儲存體相關的指標，例如帳戶中儲存貯體和物件的數目。免費指標還包括使用案例型指標 (例如，成本最佳化和資料保護指標)，您可以用來調查儲存體是否已根據 S3 最佳實務進行設定。所有免費指標均每天收集。資料可供查詢使用 14 天。如需哪些指標可與免費指標搭配使用的詳細資訊，請參閱 Amazon S3 Storage Lens 指標詞彙表。

- 進階指標和建議

  S3 Storage Lens 為所有儀表板和組態提供免費指標，包括升級至進階指標和建議的選項。需支付額外費用。如需詳細資訊，請參閱 Amazon S3 定價。

  進階指標和建議包括免費指標中的所有指標，以及其他指標，例如進階資料保護和成本最佳化指標、活動指標，以及詳細狀態碼指標。進階指標和建議也會提供建議來協助您最佳化儲存體。系統會依上下文在儀表板中的相關指標旁放置建議。

  進階指標和建議包含下列功能：

  - 進階指標 - 產生其他指標。如需進階指標的完整清單，請參閱 指標類別。如需指標的完整清單，請參閱 Amazon S3 Storage Lens 指標詞彙表。
  - Amazon CloudWatch 發佈 - 將 S3 Storage Lens 指標發佈到 CloudWatch，以在 CloudWatch 儀表板中建立統一的運作狀態檢視。您也可以使用 CloudWatch API 操作和功能 (例如警示和觸發動作、指標數學和異常偵測) 來監控 S3 Storage Lens 指標並對其採取動作。如需詳細資訊，請參閱「監控 CloudWatch 中的 S3 Storage Lens 指標」。
  - 字首彙總 - 在字首層級收集指標。啟用字首彙總，可擴充在字首層級包含在儀表板組態中的所有指標。與已設定的閾值相符的字首，才會產生指標。請注意，適用於字首層級的指標可以用於字首彙總，但儲存貯體層級設定和規則計數指標除外。字首層級指標不會發佈至 CloudWatch。
  - Storage Lens 群組彙總 – 在 Storage Lens 群組層級收集指標。啟用進階指標和建議與 Storage Lens 群組彙總之後，您可以指定要在 Storage Lens 儀表板中包含或排除哪些 Storage Lens 群組。至少必須指定一個 Storage Lens 群組。指定的 Storage Lens 群組也必須位於儀表板帳戶中指定的主要區域內。Storage Lens 群組層級指標不會發佈至 CloudWatch。

  所有進階指標均每天收集。資料最多有 15 個月的時間可用於查詢。如需 S3 Storage Lens 彙總之儲存體指標的詳細資訊，請參閱 Amazon S3 Storage Lens 指標詞彙表。

  > **Note**
  >
  > 只有在 Amazon S3 主控台上使用 S3 Storage Lens 儀表板時，才能使用建議。

S3 Storage Lens 和 AWS Organizations

AWS Organizations 是一種 AWS 服務，可協助您將所有 AWS 帳戶 彙總在一個組織階層下。Amazon S3 Storage Lens 可與 AWS Organizations 搭配使用，在 Amazon S3 儲存空間中提供物件儲存和活動的單一檢視。

如需詳細資訊，請參閱「搭配 AWS Organizations 使用 Amazon S3 Storage Lens」。

- 受信任的存取權

  您必須使用您組織的管理帳戶，啟用 S3 Storage Lens 的受信任存取權，才能彙總組織中所有成員帳戶的儲存指標和用量資料。然後，您可以使用管理帳戶或將委派管理員存取權授予組織中的其他帳戶，為組織建立儀表板或匯出。

  您可以隨時停用 S3 Storage Lens 的受信任存取權，停用時，S3 Storage Lens 就無法為您的組織彙總指標。
- 委派的管理員

  您可以使用 AWS Organizations 管理帳戶，或將委派管理員存取權授予組織中的其他帳戶，來為組織建立 S3 Storage Lens 的儀表板和指標。您可以隨時取消註冊委派管理員。取消註冊委派管理員也會自動防止透過該委派管理員建立的所有組織層級儀表板彙總新的儲存指標。

如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的 Amazon S3 Storage Lens 和 AWS Organizations。

Amazon S3 Storage Lens 服務連結角色

除了 AWS Organizations 受信任的存取權外，Amazon S3 Storage Lens 也使用 AWS Identity and Access Management (IAM) 服務連結的角色。服務連結角色是特殊類型的 IAM 角色，此角色可直接連結到 S3 Storage Lens。服務連結的角色是由 S3 Storage Lens 預先定義，其中包含從組織中成員帳戶收集每日儲存和活動指標所需的所有許可。

如需詳細資訊，請參閱在 Amazon S3 Storage Lens 使用服務連結角色。

搭配 AWS Organizations 使用 Amazon S3 Storage Lens

Amazon S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。您可以使用 S3 Storage Lens 指標產生摘要洞見，例如了解您在整個組織中擁有多少儲存體，或是成長速度最快的儲存貯體和字首有哪些。您也可以使用 S3 Storage Lens 指標，識別成本最佳化機會、實作資料保護和安全最佳實務，以及改善應用程式工作負載的效能。例如，您可以識別沒

有 S3 生命週期規則的儲存貯體，這些規則可使超過 7 天未完成的分段上傳過期。您也可以識別未遵循資料保護最佳實務的儲存貯體，例如使用 S3 複寫或 S3 版本控制。S3 Storage Lens 也會分析指標，以提供內容相關建議，您可以用來最佳化儲存成本，並套用最佳實務保護您的資料。

您可以使用 Amazon S3 Storage Lens，收集屬於 AWS Organizations 階層之所有 AWS 帳戶 的儲存指標和用量資料。若要執行此操作，您必須使用 AWS Organizations，而且必須使用 AWS Organizations 管理帳戶啟用 S3 Storage Lens 受信任的存取權。

啟用受信任的存取權之後，您可以將委派的管理員存取權新增至組織中的帳戶。然後，這些帳戶可以建立 S3 Storage Lens 組態和儀表板，以收集整個組織的儲存指標和使用者資料。

如需有關啟用受信任存取權的詳細資訊，請參閱《AWS Organizations 使用者指南》中的 Amazon S3 Storage Lens 和 AWS Organizations。

主題

啟用 S3 Storage Lens 的受信任存取權

透過啟用受信任的存取權，您就可以允許 Amazon S3 Storage Lens 透過 AWS Organizations API 操作存取 AWS Organizations 階層、成員資格和結構。然後，S3 Storage Lens 成為整個組織結構的受信任服務。

每當建立儀表板組態時，S3 Storage Lens 會在您組織的管理或委派管理員帳戶中建立服務連結角色。服務連結角色授予 S3 Storage Lens 執行下列動作的許可：

- 描述組織
- 列出帳戶
- 驗證組織的 AWS 服務 存取清單
- 取得組織的委派管理員

然後，S3 Storage Lens 可以確保其擁有存取權，以收集組織中帳戶的跨帳戶指標。如需詳細資訊，請參閱在 Amazon S3 Storage Lens 使用服務連結角色。

啟用受信任存取權之後，您就可以將委派管理員存取權指派給組織中的帳戶。將帳戶標示為服務的委派管理員時，帳戶會收到所有唯讀組織 API 操作的存取授權。此存取權會提供組織成員和結構的委派管理員可見性，讓他們也可以建立 S3 Storage Lens 儀表板。

> **Note**
>
> 只有管理帳戶可以啟用 Amazon S3 Storage Lens 的受信任存取權。

停用 S3 Storage Lens 的受信任存取權

停用受信任存取權後，您可以限制 S3 Storage Lens 只能在帳戶層級上運作。此外，每個帳戶持有人只能看到其帳戶範圍內 (而不是整個組織) 的 S3 Storage Lens。任何需要受信任存取權的儀表板都不會再更新，但會保留其歷史資料，直到資料可用於查詢的期間結束。

> **Note**
>
> - 停用 S3 Storage Lens 的受信任存取權也會使所有組織層級儀表板自動停止收集和彙總儲存空間指標。
> - 您的管理和委派管理員帳戶仍然可以在資料可用於查詢時段期間，查看現有組織層級儀表板的歷史資料。

註冊 S3 Storage Lens 的委派管理員

您可以使用組織的管理帳戶或委派管理員帳戶，來建立組織層級儀表板。委派管理員帳戶允許管理帳戶以外的其他帳戶建立組織層級儀表板。僅組織的管理帳戶可以將其他帳戶註冊為組織的委派管理員，以及取消註冊委派管理員。

若要使用 Amazon S3 主控台註冊委派管理員，請參閱 正在註冊 S3 Storage Lens 的委派管理員。

您也可以使用 AWS Organizations REST API、AWS CLI 或 SDK，透過管理帳戶註冊委派管理員。如需詳細資訊，請參閱《AWS Organizations API 參考》 中的 RegisterDelegatedAdministrator。

> **Note**
>
> 在您可以使用 AWS Organizations REST API、AWS CLI 或 SDK，指定委派管理員之前，您必須先呼叫 EnableAWSOrganizationsAccess 操作。

取消註冊 S3 Storage Lens 的委派管理員

您也可以取消註冊委派管理員帳戶。委派管理員帳戶允許管理帳戶以外的其他帳戶建立組織層級儀表板。只有組織的管理帳戶可以透過組織委派管理員的身分取消註冊帳戶。

若要使用 S3 主控台取消註冊委派管理員，請參閱 正在取消註冊 S3 Storage Lens 的委派管理員。

您也可以使用 AWS Organizations REST API、AWS CLI 或 SDK，透過管理帳戶取消註冊委派管理員。如需詳細資訊，請參閱《AWS Organizations API 參考》 中的 DeregisterDelegatedAdministrator。

> **Note**
>
> - 取消註冊委派管理員也會自動防止透過該委派管理員建立的所有組織層級儀表板彙總新的儲存指標。
> - 已取消註冊的委派管理員帳戶仍然可以查看在資料可用於查詢時所建立的儀表板歷史資料。

Amazon S3 Storage Lens 許可

Amazon S3 Storage Lens 需要 AWS Identity and Access Management (IAM) 中的新許可，才能授權對 S3 Storage Lens 的存取權。若要授予這些許可，您可以使用身分型 IAM 政策。您可以將此政策連接至 IAM 使用者、群組或角色，為他們授予許可。此類許可能夠包含啟用或停用 S3 Storage Lens，或存取任何 S3 Storage Lens 儀表板或組態的能力。

IAM 使用者或角色必須屬於建立或擁有儀表板或組態的帳戶，除非以下兩個條件均成立：

- 您的帳戶是 AWS Organizations 的成員。
- 您有權以委派管理員的身分使用管理帳戶建立組織層級儀表板。

> **Note**
>
> - 您無法使用帳戶的根使用者憑證，來檢視 Amazon S3 Storage Lens 儀表板。若要存取 S3 Storage Lens 儀表板，必須將必要的 IAM 許可授予新的或現有的 IAM 使用者。然後，使用這些使用者憑證登入，以存取 S3 Storage Lens 儀表板。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的安全性最佳實務。

- 在 Amazon S3 主控台上使用 S3 Storage Lens 可能需要多個許可。例如，若要在主控台上編輯儀表板，您需要下列許可：
  - `s3:ListStorageLensConfigurations`
  - `s3:GetStorageLensConfiguration`
  - `s3:PutStorageLensConfiguration`

主題

設定使用 S3 Storage Lens 的帳戶許可

若要建立及管理 S3 Storage Lens 儀表板和 Storage Lens 儀表板組態，您必須具備下列許可 (具體取決於您要執行的動作)：

Amazon S3 Storage Lens 相關的 IAM 許可

動作	IAM 許可
在 Amazon S3 主控台中建立或更新 S3 Storage Lens 儀表板。	`s3:ListStorageLensConfigurations` `s3:GetStorageLensConfiguration` `s3:GetStorageLensConfigurationTagging` `s3:PutStorageLensConfiguration` `s3:PutStorageLensConfigurationTagging`
在 Amazon S3 主控台上取得 S3 Storage Lens 儀表板的標籤。	`s3:ListStorageLensConfigurations` `s3:GetStorageLensConfigurationTagging`

動作	IAM 許可
在 Amazon S3 主控台上檢視 S3 Storage Lens 儀表板。	`s3:ListStorageLensConfigurations` `s3:GetStorageLensConfiguration` `s3:GetStorageLensDashboard`
刪除 Amazon S3 主控台上 S3 Storage Lens 儀表板。	`s3:ListStorageLensConfigurations` `s3:GetStorageLensConfiguration` `s3:DeleteStorageLensConfiguration`
使用 AWS CLI 或 AWS SDK 建立或更新 S3 Storage Lens 組態。	`s3:PutStorageLensConfiguration` `s3:PutStorageLensConfigurationTagging`
使用 AWS CLI 或 AWS SDK 取得 S3 Storage Lens 組態的標籤。	`s3:GetStorageLensConfigurationTagging`
使用 AWS CLI 或 AWS SDK 檢視 S3 Storage Lens 組態。	`s3:GetStorageLensConfiguration`
使用 AWS CLI 或 AWS SDK 刪除 S3 Storage Lens 組態。	`s3:DeleteStorageLensConfiguration`

Note

- 您可以在 IAM 政策中使用資源標籤來管理許可。
- 具有這些許可的 IAM 使用者或角色可以從儲存貯體和字首查看指標，這些 IAM 使用者/角色可能沒有從這些儲存貯體和字首直接讀取或列出物件的許可。
- 對於已啟用字首層級指標的 S3 Storage Lens 儀表板，如果選取的字首路徑與物件索引鍵相符，儀表板可能會將物件索引鍵顯示為另一個字首。

- 對於在您帳戶中儲存貯體存放的指標匯出，使用 IAM 政策中現有的 `s3:GetObject` 許可來授予許可。同樣地，對於 AWS Organizations 實體，組織管理帳戶或委派管理員帳戶可以使用 IAM 政策，來管理組織層級儀表板和組態的存取許可。

設定使用 S3 Storage Lens 群組的帳戶許可

您可以使用 S3 Storage Lens 群組，根據字首、尾碼、物件標籤、物件大小或物件存留期，了解儲存在儲存貯體內的分佈情形。您可以將 Storage Lens 群組連接至儀表板，以檢視其彙總指標。

若要使用 Storage Lens 群組，您需要特定許可。如需詳細資訊，請參閱「the section called “Storage Lens 群組許可”」。

設定搭配 AWS Organizations 使用 Amazon S3 Storage Lens 的許可

您可以使用 Amazon S3 Storage Lens，收集屬於 AWS Organizations 階層之所有帳戶的儲存指標和用量資料。以下是將 Organizations 與 S3 Storage Lens 搭配使用的相關動作和許可。

使用 S3 Storage Lens 的 AWS Organizations 相關 IAM 許可

動作	IAM 許可
為您的組織啟用 S3 Storage Lens 的受信任存取權。	`organizations:EnableAWSServiceAccess`
針對組織停用 S3 Storage Lens 的受信任存取權。	`organizations:DisableAWSServiceAccess`
註冊委派管理員，為您的組織建立 S3 Storage Lens 儀表板或組態。	`organizations:RegisterDelegatedAdministrator`
取消註冊委派管理員，以便其無法再針對您的組織建立 S3 Storage Lens 儀表板或組態。	`organizations:DeregisterDelegatedAdministrator`
建立 S3 Storage Lens 全組織組態的額外許可。	`organizations:DescribeOrganization` `organizations:ListAccounts`

動作	IAM 許可
	`organizations:ListAWSServiceAccessForOrganization` `organizations:ListDelegatedAdministrators` `iam:CreateServiceLinkedRole`

使用 Amazon S3 Storage Lens 檢視指標

S3 Storage Lens 會彙總您的指標，並在 Amazon S3 主控台 Buckets (儲存貯體) 頁面上的 Account snapshot (帳戶快照) 區段中顯示資訊。S3 Storage Lens 也會提供互動式儀表板，您可以用來以視覺化方式呈現洞見與趨勢、標記異常值，以及接收最佳化儲存成本和套用資料保護最佳實務的建議。您的儀表板具有向下切入選項可產生及視覺化組織、帳戶、AWS 區域、儲存類別、儲存貯體、字首或 Storage Lens 群組層級的洞察。您也可以將 CSV 或 Parquet 格式的每日指標匯出傳送到 S3 儲存貯體。

依預設，所有儀表板都是使用免費指標設定的，您可以使用其中包括的指標，來了解 S3 儲存體的用量和活動、最佳化儲存成本，以及實作資料保護和存取管理最佳實務。免費指標會向下彙總至儲存貯體層級。使用免費指標，資料可供查詢使用多達 14 天。

進階指標和建議包含下列其他功能，您可以用來進一步洞察跨儲存體的用量和活動，以及最佳化儲存體的最佳實務：

- 內容相關建議 (只能在主控台中使用)
- 進階指標 (包括依儲存貯體彙總的活動指標)
- 字首彙總
- Storage Lens 群組彙總
- Storage Lens 群組彙總
- Amazon CloudWatch 發佈

進階指標資料可供查詢使用 15 個月。使用具有進階指標的 S3 Storage Lens 需支付額外費用。如需詳細資訊，請參閱 Amazon S3 定價。如需免費和進階指標的詳細資訊，請參閱 指標選擇。

主題

在儀表板上檢視 S3 Storage Lens 指標

在 Amazon S3 主控台，S3 Storage Lens 提供互動式預設儀表板，可讓您用來視覺化資料中的洞察與趨勢。您也可以使用此儀表板來標記極端值，以及接收最佳化儲存成本並套用資料保護最佳實務的建議。您的儀表板具有向下切入選項，用以產生帳戶、儲存貯體、AWS 區域、字首或 Storage Lens 群組層級的洞察。如果您已啟用 S3 Storage Lens 以使用 AWS Organizations，您也可以產生組織層級的洞察 (例如，所有屬於 AWS Organizations 階層之帳戶的資料)。您的儀表板一律會載入具有可用指標的最新日期。

主控台上的 S3 Storage Lens 預設儀表板名稱為 default-account-dashboard。Amazon S3 會預先設定此儀表板，以視覺化整個帳戶的摘要洞察和趨勢，並在 Amazon S3 主控台每日加以更新。您無法修改預設儀表板的設定範圍，但可以將指標選項從免費指標升級為付費進階指標和建議。使用進階指標和建議，您可以存取其他指標和功能。這些功能包括進階字首類別、字首層級彙總、內容相關建議，以及 Amazon CloudWatch 發佈。

您可以停用預設儀表板，但無法將其刪除。如果停用預設儀表板，則不再更新它。您也將不再於 S3 Storage Lens 中或儲存貯體頁面上的帳戶快照中收到任何新的每日指標。您仍然可在預設儀表板中查看歷史資料，直到資料查詢的 14 天期間過期為止。如果您已啟用進階指標和建議，則此期間為 15 個月。若要存取此資料，您可以在期限內重新啟用儀表板。

您可以建立其他 S3 Storage Lens 儀表板，並依 AWS 區域、S3 儲存貯體或帳戶限定其範圍。如果您已啟用 Storage Lens 以使用 AWS Organizations，您也可以依組織限定儀表板的範圍。建立或編輯 S3 Storage Lens 儀表板時，您可以定義儀表板範圍和指標選擇。

您可以停用或刪除您建立的任何其他儀表板。

- 如果停用儀表板，該儀表板將不再更新，而且您將不再收到任何新的每日指標。您仍然可以查看免費指標的歷史資料，直到 14 天期間過期為止。如果您已針對該儀表板啟用進階指標和建議，則此期間為 15 個月。若要存取此資料，您可以在期限內重新啟用儀表板。
- 如果刪除儀表板，則會遺失所有儀表板組態設定。您將不再收到任何新的每日指標，也無法存取與該儀表板相關聯的歷史資料。如果想要存取已刪除儀表板的歷史資料，必須在同一個主要區域中建立另一個具有相同名稱的儀表板。

主題

檢視 Amazon S3 Storage Lens 儀表板

下列程序說明如何在 S3 主控台中檢視 S3 Storage Lens 儀表板。如需使用案例型逐步解說以了解如何使用儀表板將成本最佳化、實作最佳實務，以及改善存取 S3 儲存貯體的應用程式效能，請參閱 Amazon S3 Storage Lens 指標使用案例。

> **Note**
>
> 您無法使用帳戶的根使用者憑證，來檢視 Amazon S3 Storage Lens 儀表板。若要存取 S3 Storage Lens 儀表板，必須將必要的 AWS Identity and Access Management (IAM) 許可授予新的或現有的 IAM 使用者。然後，使用這些使用者憑證登入，以存取 S3 Storage Lens 儀表板。如需詳細資訊，請參閱《IAM 使用者指南》中的 Amazon S3 Storage Lens 許可 及 IAM 中的安全最佳實務。

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板。

   您的儀表板即會在 S3 Storage Lens 中開啟。Snapshot for date (日期的快照) 區段會顯示 S3 Storage Lens 收集指標的最新日期。您的儀表板一律會載入具有可用指標的最新日期。
4. (選用) 若要變更 S3 Storage Lens 儀表板的日期，請在右上角的日期選取器中選擇新的日期。
5. (選用) 若要套用臨時篩選條件以進一步限制儀表板資料的範圍，請執行下列動作：

   a. 展開篩選條件區段。

   b. 若要依特定帳戶、AWS 區域、儲存類別或儲存貯體、字首或 Storage Lens 群組進行篩選，請選擇篩選依據的選項。

   > **Note**
   >
   > 字首篩選條件和 Storage Lens 群組篩選條件無法同時套用。

c. 若要更新篩選條件，請選擇 Apply (套用)。

d. 若要移除篩選條件，請按一下篩選條件旁的 X。

6. 在 S3 Storage Lens 儀表板的任何區段中，若要查看特定指標的資料，請針對 Metric (指標) 選擇指標名稱。

7. 在 S3 Storage Lens 儀表板的任何圖表或視覺效果中，您可以向下切入至更深層的彙總，方法為使用帳戶、AWS 區域、儲存類別、儲存貯體、字首或 Storage Lens 群組索引標籤。如需範例，請參閱 瞭解冷 Amazon S3 儲存貯體。

瞭解您的 S3 Storage Lens 儀表板

您的 S3 Storage Lens 儀表板具有主要 Overview (概觀) 索引標籤，以及最多五個代表各彙總層級的其他索引標籤：

- 帳戶
- AWS 區域
- 儲存類別
- 儲存貯體
- 字首
- Storage Lens 群組

在 Overview (概觀) 索引標籤上，儀表板資料會彙總為三個不同區段：Snapshot for date (日期的快照)、Trends and distributions(趨勢和分佈)，以及 Top N overview (前 N 個概觀)。

如需 S3 Storage Lens 儀表板的詳細資訊，請參閱下列各節。

快照

Snapshot for date (日期的快照) 區段會顯示 S3 Storage Lens 已為所選日期收集的摘要指標。這些摘要指標包括下列指標：

- 總儲存空間 - 已使用的儲存空間總量 (以位元組為單位)。
- 物件計數 - 您 AWS 帳戶 的物件總數。
- 平均物件大小 - 平均的物件大小。
- 作用中儲存貯體 - 您的帳戶中，作用中儲存體用量 > 0 位元組的作用中儲存貯體總數。

- 帳戶 - 儲存體在範圍內的帳戶數目。除非您使用 AWS Organizations，且您的 S3 Storage Lens 具有有效服務連結角色的受信任存取權，否則此值為 1。如需詳細資訊，請參閱「讓 Amazon S3 Storage Lens 使用服務連結角色」。
- 儲存貯體 - 帳戶中的儲存貯體總數。

指標資料

對於快照中出現的每個指標，您可以看到下列資料：

- 指標名稱 - 指標的名稱。
- 指標類別 - 指標組織成的類別。
- 日期總計 - 所選日期的總計數。
- % 變更 - 自上一個快照日期以來的百分比變更。
- 30 天趨勢 - 顯示 30 天期間內指標變更的趨勢線。
- 建議 - 以快照中提供的資料為基礎的內容相關建議。這些建議可與進階指標和建議搭配使用。如需詳細資訊，請參閱「建議」。

指標類別

您可以選擇性地更新儀表板 Snapshot for date (日期的快照) 區段，以顯示其他類別的指標。如果想要查看其他指標的快照資料，您可以從下列 Metrics categories (指標類別) 中進行選擇：

- 成本最佳化
- 資料保護
- 活動 (可與進階指標搭配使用)
- 存取管理
- 效能
- 事件

Snapshot for date (日期的快照) 區段只會針對每個類別顯示指標的選取項目。若要查看特定類別的所有指標，請在 Trends and distributions (趨勢和分佈) 或 Top N overview (前 N 個概觀) 區段中選擇指標。如需指標類別的詳細資訊，請參閱 指標類別。如需 S3 Storage Lens 指標的完整清單，請參閱 Amazon S3 Storage Lens 指標詞彙表。

趨勢和分佈

Overview (概觀) 索引標籤的第二個區段是 Trends and distribution (趨勢和分佈)。在 Trends and distributions (趨勢和分佈) 區段中，您可以選擇兩個指標，在您定義的日期範圍內進行比較。Trends and distributions (趨勢和分佈) 區段會顯示一段時間後兩個指標之間的關係。此區段會顯示圖表，您可以用來查看正在追蹤的兩個趨勢之間的 Storage class (儲存體類別) 和 Region (區域) 分佈。您可以選擇性地向下切入至其中一個圖表的資料點，以進行更深入的分析。

如需使用 Trends and distributions (趨勢和分佈) 區段的演練，請參閱 識別未使用伺服器端加密搭配 AWS KMS 進行預設加密 (SSE-KMS) 的儲存貯體。

前 N 個概觀

S3 Storage Lens 儀表板的第三區段是 Top N overview (前 N 個概觀) (以遞增或遞減方式排序) 。此區段會顯示您在最高排名的帳戶、AWS 區域、儲存貯體、字首或 Storage Lens 群組間選取的指標。如果您已啟用 S3 Storage Lens 以使用 AWS Organizations，您也可以查看在整個組織中選取的指標。

如需使用 Top N overview (前 N 個概觀) 區段的演練，請參閱 識別您的最大 S3 儲存貯體。

依選項向下切入並分析

為了提供流暢的分析體驗，S3 Storage Lens 儀表板提供了動作功能表，會在您選擇任一圖表值時出現。若要使用此功能表，請選擇任一圖表值以查看相關聯的指標值，然後在出現的方塊中從兩個選項進行選擇：

- Drill down (向下切入) 動作會將所選值作為篩選條件，套用至儀表板的所有索引標籤。然後，您可以向下切入該值以進行更深入分析。
- 分析依據動作會將您帶至您所選取的維度索引標籤，並將該值索引標籤作為篩選條件套用。這些索引標籤包括帳戶、AWS 區域、儲存類別、儲存貯體、字首 (針對已啟用進階指標和字首彙總的儀表板) 和 Storage Lens 群組 (針對已啟用進階指標和 Storage Lens 群組彙總的儀表板) 透過分析依據，您可以在新維度的內容中檢視資料，以進行更深入的分析。

若成果產生不合乎邏輯的結果，或沒有任何值，表示向下切入和分析依據動作可能已停用。向下切入和分析依據動作所套用的篩選條件都會優先於儀表板的所有索引標籤間任何現有的篩選條件。您也可以視需要移除篩選條件。

標籤

維度層級索引標籤提供特定維度內所有值的詳細檢視。例如，AWS 區域 索引標籤會顯示所有 AWS 區域 的指標，而儲存貯體索引標籤則會顯示所有儲存貯體的指標。每個維度頁籤都包含由四個區段組成的相同配置：

- 趨勢圖會顯示過去 30 天內，選取指標在維度中的前 N 個項目。依預設，此圖表會顯示前 10 個項目，但您可將其減少為至少 3 個項目，或增加為最多 50 個項目。
- 一個直方圖，顯示所選日期和指標的垂直長條圖。如果您有大量項目要顯示在此圖表中，可能會需要水平捲動。
- 一個氣泡分析圖，會繪製維度內的所有項目。此圖表代表 x 軸上的第一個指標和 y 軸上的第二個指標。第三個指標由氣泡的大小表示。
- 一個指標網格檢視，包含維度中每列所列的各個項目。資料欄代表每個可用的指標，並排列在指標類別標籤中，以方便導覽。

使用資料匯出檢視 Amazon S3 Storage Lens 指標

每天會以 CSV 或 Apache Parquet 格式的指標匯出檔案產生 Amazon S3 Storage Lens 指標，並將其放置在您帳戶的 S3 儲存貯體中。從那裡，您可以將指標匯出導入到您選擇的分析工具中，例如 Amazon QuickSight 和 Amazon Athena，您可以在其中分析儲存使用情況和活動趨勢。

主題

- 使用 AWS KMS key 來加密指標匯出
- 什麼是 S3 Storage Lens 匯出資訊清單？
- 瞭解 Amazon S3 Storage Lens 匯出結構描述

使用 AWS KMS key 來加密指標匯出

若要授予 Amazon S3 Storage Lens 使用客戶受管金鑰加密指標匯出的許可，您必須使用金鑰政策。若要更新您的金鑰政策，以使用 KMS 金鑰來加密 S3 Storage Lens 指標匯出，請依照以下步驟執行。

授予 S3 Storage Lens 使用 KMS 金鑰加密資料的許可

1. 使用擁有客戶管理金鑰的登入。 AWS Management Console AWS 帳戶
2. 開啟主 AWS KMS 控台，網址為 https://console.aws.amazon.com/kms。
3. 若要變更 AWS 區域，請使用頁面右上角的「地區」選取器。

4. 在左側導覽窗格中，選擇 Customer managed keys (客戶受管金鑰)。
5. 在客戶受管金鑰下，選擇您要用來加密指標匯出的金鑰。 AWS KMS keys 是區域特定的，且必須與指標匯出目的地 S3 儲存貯體位於相同的區域。
6. 在 Key policy (金鑰政策) 下，選擇 Switch to policy view (切換至政策檢視)。
7. 若要更新金鑰政策，請選擇 Edit (編輯)。
8. 在 Edit key policy (編輯金鑰政策) 下，將下列金鑰政策新增至現有的金鑰政策。若要使用此政策，請以您的資訊取代 *user input placeholders*。

```
{
    "Sid": "Allow Amazon S3 Storage Lens use of the KMS key",
     "Effect": "Allow",
    "Principal": {
        "Service": "storage-lens.s3.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
       "StringEquals": {
           "aws:SourceArn": "arn:aws:s3:us-east-1:source-account-id:storage-
lens/your-dashboard-name",
           "aws:SourceAccount": "source-account-id"
        }
     }
}
```

9. 選擇儲存變更。

如需建立客戶受管金鑰和使用金鑰政策的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的下列主題：

- 入門
- 在中使用金鑰原則 AWS KMS

您也可以使用金 AWS KMS PUT鑰政策 API 作業 (PutKeyPolicy)， AWS CLI將金鑰政策複製到客戶管理金鑰，您要使用 REST API 和 SDK 來加密指標匯出時使用。

什麼是 S3 Storage Lens 匯出資訊清單？

有鑑於資料彙總量龐大，S3 Storage Lens 每日指標匯出可以分割成多個檔案。資訊清單檔案 manifest.json 會描述當日指標匯出檔案的位置。每次交付新的匯出時，都會伴隨新的資訊清單。manifest.json 檔案內所包含的每個資訊清單檔案，都會提供匯出的中繼資料和其他基本資訊。

資訊清單資訊包含以下屬性：

- sourceAccountId - 組態擁有者的帳戶 ID。
- configId - 儀表板的唯一識別碼。
- destinationBucket - 指標匯出所放置之目的地儲存貯體的 Amazon Resource Name (ARN)。
- reportVersion - 匯出的版本。
- reportDate - 報告的日期。
- reportFormat - 報告的格式。
- reportSchema - 報告的結構描述。
- reportFiles - 目標儲存貯體中匯出報告檔案的實際清單。

以下是 manifest.json 檔案中 CSV 格式匯出的資訊清單範例。

```
{
   "sourceAccountId":"123456789012",
   "configId":"my-dashboard-configuration-id",
   "destinationBucket":"arn:aws:s3:::destination-bucket",
   "reportVersion":"V_1",
   "reportDate":"2020-11-03",
   "reportFormat":"CSV",

 "reportSchema":"version_number,configuration_id,report_date,aws_account_number,aws_region,stor
   "reportFiles":[
        {
            "key":"DestinationPrefix/StorageLens/123456789012/my-dashboard-
configuration-id/V_1/reports/dt=2020-11-03/a38f6bc4-2e3d-4355-ac8a-e2fdcf3de158.csv",
            "size":1603959,
            "md5Checksum":"2177e775870def72b8d84febe1ad3574"
        }
      ]
```

```
}
```

以下是 manifest.json 檔案中 Parquet 格式匯出的清單檔案範例。

```
{
   "sourceAccountId":"123456789012",
   "configId":"my-dashboard-configuration-id",
   "destinationBucket":"arn:aws:s3:::destination-bucket",
   "reportVersion":"V_1",
   "reportDate":"2020-11-03",
   "reportFormat":"Parquet",
   "reportSchema":"message s3.storage.lens { required string version_number;
 required string configuration_id; required string report_date; required string
 aws_account_number; required string aws_region; required string storage_class;
 required string record_type; required string record_value; required string
 bucket_name; required string metric_name; required long metric_value; }",
   "reportFiles":[
      {
         "key":"DestinationPrefix/StorageLens/123456789012/my-dashboard-configuration-
id/V_1/reports/dt=2020-11-03/bd23de7c-b46a-4cf4-bcc5-b21aac5be0f5.par",
         "size":14714,
         "md5Checksum":"b5c741ee0251cd99b90b3e8eff50b944"
      }
}
```

您可以在 Amazon S3 主控台中將指標匯出設定為儀表板組態的一部分，AWS CLI或使用 Amazon S3 REST API 和開發套件來產生。

瞭解 Amazon S3 Storage Lens 匯出結構描述

下表包含 S3 Storage Lens 指標匯出的結構描述。

屬性名稱	資料類型	資料欄名稱	描述
VersionNumber	字串	version_number	正在使用 S3 Storage Lens 指標版本。
ConfigurationId	字串	configuration_id	S3 Storage Lens 組態 configuration_id 。

屬性名稱	資料類型	資料欄名稱	描述
ReportDate	字串	report_date	追蹤指標的日期。
AwsAccountNumber	字串	aws_account_number	你的 AWS 帳戶 號碼
AwsRegion	字串	aws_region	追蹤量度的目標。AWS 區域
StorageClass	字串	storage_class	儲存貯體的儲存類別有問題。
RecordType	ENUM	record_type	正在回報的成品類型 (「帳戶」、「儲存貯體」或「前綴」)。
RecordValue	字串	record_value	RecordType 成品的價值。 **Note** record_value 為 URL 編碼。
BucketName	字串	bucket_name	正在回報的儲存貯體名稱。
MetricName	字串	metric_name	正在回報的指標名稱。
MetricValue	Long	metric_value	正在回報的指標值。

S3 Storage Lens 指標匯出的範例

以下是以此結構描述為基礎的 S3 Storage Lens 指標匯出範例。

> **Note**
>
> 您可以在 `record_type` 欄中尋找 `STORAGE_LENS_GROUP_BUCKET` 或 `STORAGE_LENS_GROUP_ACCOUNT` 值，以識別 Storage Lens 群組的指標。`record_value` 欄會顯示 Storage Lens 群組的 Amazon Resource Name (ARN)，例如 `arn:aws:s3:us-east-1:123456789012:storage-lens-group/slg-1`。

version_r	configuration_id	report_date	aws_account_number	aws_region	storage_class	record_type	record_value	bucket_name	metric_name	metric_value
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			StorageBytes	2478830621
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectCount	1598962
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ReplicatedStorageBytes	20000
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ReplicatedObjectCount	20
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			EncryptedStorageBytes	2478828742
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			EncryptedObjectCount	1598961
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			DeleteMarkerObjectCount	1500
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectLockEnabledStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectLockEnabledObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			CurrentVersionStorageBytes	2478830621
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			CurrentVersionObjectCount	1598962
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			NonCurrentVersionStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			NonCurrentVersionObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			IncompleteMultipartUploadStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			IncompleteMultipartUploadObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	StorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	ObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	ReplicatedStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	ReplicatedObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	EncryptedStorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	EncryptedObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	DeleteMarkerObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	ObjectLockEnabledStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	ObjectLockEnabledObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	CurrentVersionStorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	CurrentVersionObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	NonCurrentVersionStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	NonCurrentVersionObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	IncompleteMultipartUploadStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	IncompleteMultipartUploadObjectCount	0

以下是 Storage Lens 群組資料的 S3 Storage Lens 指標匯出範例。

version_numbe	configuration_id	report_date	aws_account_num	aws_region	storage_class	record_type	record_value	bucket_name	metric_name	metric_value
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		StorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		EncryptedStorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		EncryptedObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		DeleteMarkerObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectLockEnabledStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectLockEnabledObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		CurrentVersionStorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		CurrentVersionObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		NonCurrentVersionStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		NonCurrentVersionObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMultipartUploadStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMultipartUploadObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		DeleteMarkerStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedStorageBytesSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedObjectCountSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMPUStorageBytesOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMPUObjectCountOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	StorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	EncryptedStorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	EncryptedObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	DeleteMarkerObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectLockEnabledStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectLockEnabledObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	CurrentVersionStorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	CurrentVersionObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	NonCurrentVersionStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	NonCurrentVersionObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMultipartUploadStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMultipartUploadObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	DeleteMarkerStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedStorageBytesSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedObjectCountSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMPUStorageBytesOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMPUObjectCountOlderThan7Days	0

監控 CloudWatch 中的 S3 Storage Lens 指標

您可以將 S3 Storage Lens 指標發佈到 Amazon CloudWatch，以在 CloudWatch 儀表板中建立統一的運作狀態檢視。您也可以使用 CloudWatch 功能 (例如警示和觸發動作、指標數學和異常偵測) 來監控 S3 Storage Lens 指標並對其採取動作。此外，CloudWatch API 操作可讓應用程式 (包括第三方供應商) 存取您的 S3 Storage Lens 指標。如需有關 CloudWatch 功能的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》。

您可以使用 Amazon S3 主控台、Amazon S3 REST API、AWS CLI 和 AWS SDK 為新的或現有儀表板組態啟用 CloudWatch 發佈選項。升級至 S3 Storage Lens 進階指標和建議儀表板可以使用 CloudWatch 發佈選項。如需 S3 Storage Lens 進階指標和建議定價，請參閱 Amazon S3 定價。不會產生額外的 CloudWatch 指標發佈費用；不過，儀表板、警示和 API 呼叫等其他 CloudWatch 費用則適用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

S3 Storage Lens 指標會在擁有 S3 Storage Lens 組態的帳戶中發佈至 CloudWatch。在進階指標和建議中啟用 CloudWatch 發佈選項之後，您可以在 CloudWatch 中存取組織、帳戶和儲存貯體層級的指標。在 CloudWatch 中不提供字首層級的指標。

> **Note**
>
> S3 Storage Lens 指標是每日指標，每天向 CloudWatch 發佈一次。在 CloudWatch 中查詢 S3 Storage Lens 指標時，查詢的時段必須為 1 天 (86400 秒)。在每日 S3 Storage Lens 指標出現在 Amazon S3 主控台的 S3 Storage Lens 儀表板中之後，這些相同指標可能會需要幾個小時才會出現在 CloudWatch 中。首次啟用 S3 Storage Lens 指標的 CloudWatch 發佈選項時，您的指標最多可能需要 24 小時才能發佈到 CloudWatch。

啟用 CloudWatch 發佈選項後，您可以使用下列 CloudWatch 功能來監控和分析您的 S3 Storage Lens 資料：

- 儀表板 – 使用 CloudWatch 儀表板來建立自訂的 S3 Storage Lens 儀表板。將您的 CloudWatch 儀表板與無法直接存取 AWS 帳戶 的人員、跨團隊、與利害關係人，以及與組織外部人員共用。
- 警示和觸發的動作 – 設定警示，以監視指標，並在超出閾值時採取動作。例如，您可以設定警示，當 Incomplete Multipart Upload Bytes (未完成的分段上傳位元組) 指標連續三天超過 1 GB 時，傳送 Amazon SNS 通知。
- 異常偵測 – 啟用異常偵測以持續分析指標、判斷正常基準以及顯露異常情況。您可以建立以指標預期值為基礎的異常偵測警示。例如，您可以監控 Object Lock Enabled Bytes (已啟用物件鎖定的位元組) 指標的異常情況，以偵測未經授權即移除物件鎖定設定的情況。

- 指標數學 – 您還可以使用指標數學，以查詢多個 S3 Storage Lens 指標，並使用數學表達式根據這些指標來建立新的時間序列。例如，您可以建立新指標，透過將 `StorageBytes` 除以 `ObjectCount` 來取得平均物件大小。

如需有關 S3 Storage Lens 指標之 CloudWatch 發佈選項的詳細資訊，請參閱下列主題。

主題

- S3 Storage Lens 指標和維度
- 啟用 S3 Storage Lens 的 CloudWatch 發佈
- 使用 CloudWatch 中的 S3 Storage Lens 指標

S3 Storage Lens 指標和維度

若要將 S3 Storage Lens 指標傳送至 CloudWatch，您必須在 S3 Storage Lens 進階指標和建議中啟用 CloudWatch 發佈選項。在啟用進階指標之後，您可以使用 CloudWatch 儀表板來監控 S3 Storage Lens 指標與其他應用程式指標，並建立統一的運作狀態檢視。您可以使用維度，依組織、帳戶、儲存貯體、儲存類別、區域和指標組態 ID 在 CloudWatch 中篩選 S3 Storage Lens 指標。

S3 Storage Lens 指標會在擁有 S3 Storage Lens 組態的帳戶中發佈至 CloudWatch。在進階指標和建議中啟用 CloudWatch 發佈選項之後，您可以在 CloudWatch 中存取組織、帳戶和儲存貯體層級的指標。在 CloudWatch 中不提供字首層級的指標。

> **Note**
>
> S3 Storage Lens 指標是每日指標，每天向 CloudWatch 發佈一次。在 CloudWatch 中查詢 S3 Storage Lens 指標時，查詢的時段必須為 1 天 (86400 秒)。在每日 S3 Storage Lens 指標出現在 Amazon S3 主控台的 S3 Storage Lens 儀表板中之後，這些相同指標可能會需要幾個小時才會出現在 CloudWatch 中。首次啟用 S3 Storage Lens 指標的 CloudWatch 發佈選項時，您的指標最多可能需要 24 小時才能發佈到 CloudWatch。

如需有關 CloudWatch 中 S3 Storage Lens 指標和維度的詳細資訊，請參閱下列主題。

主題

- 指標
- 維度

指標

S3 Storage Lens 指標會在 CloudWatch 中作為指標提供。S3 Storage Lens 指標會發佈至 `AWS/S3/Storage-Lens` 命名空間。此命名空間僅適用於 S3 Storage Lens 指標。Amazon S3 儲存貯體、請求和複寫指標會發佈至 `AWS/S3` 命名空間。

S3 Storage Lens 指標會在擁有 S3 Storage Lens 組態的帳戶中發佈至 CloudWatch。在進階指標和建議中啟用 CloudWatch 發佈選項之後，您可以在 CloudWatch 中存取組織、帳戶和儲存貯體層級的指標。在 CloudWatch 中不提供字首層級的指標。

在 S3 Storage Lens 中，指標只會彙總並存放在指定的主要區域。S3 Storage Lens 指標也會發佈至您在 S3 Storage Lens 組態中指定主要區域中的 CloudWatch。

如需 S3 Storage Lens 指標的完整清單，包括 CloudWatch 中這些可用指標的清單，請參閱 Amazon S3 Storage Lens 指標詞彙表。

> **Note**
>
> CloudWatch 中 S3 Storage Lens 指標的有效統計數字為平均值。如需 CloudWatch 中統計數字的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 CloudWatch 統計數字定義。

CloudWatch 中的 S3 Storage Lens 指標資料精細程度

S3 Storage Lens 提供組織、帳戶、儲存貯體和字首精細程度的指標。S3 Storage Lens 會將組織、帳戶和儲存貯體層級的 S3 Storage Lens 指標發佈至 CloudWatch。在 CloudWatch 中不提供字首層級的 S3 Storage Lens 指標。

如需 CloudWatch 中可用 S3 Storage Lens 指標精細程度的詳細資訊，請參閱下列清單：

- 組織 – 組織中跨成員帳戶彙總的指標。S3 Storage Lens 會將成員帳戶的指標發佈到管理帳戶中的 CloudWatch。
  - 組織和帳戶 – 組織中成員帳戶的指標。
  - 組織和儲存貯體 – 組織中成員帳戶的 Amazon S3 儲存貯體指標。
- 帳戶 (非組織層級) – 在您帳戶中跨儲存貯體彙總的指標。
- 儲存貯體 (非組織層級) – 特定儲存貯體的指標。在 CloudWatch 中，S3 Storage Lens 會將這些指標發佈到建立 S3 Storage Lens 組態的 AWS 帳戶中。S3 Storage Lens 只會針對非組織組態發佈這些指標。

維度

當 S3 Storage Lens 傳送資料至 CloudWatch 時，維度會連接至每個指標。維度是描述指標特性的類別。您可以使用維度來篩選 CloudWatch 傳回的結果。

例如，CloudWatch 中的所有 S3 Storage Lens 指標都有 `configuration_id` 維度。您可以使用此維度來區分與特定 S3 Storage Lens 組態相關聯的指標。`organization_id` 會識別組織層級指標。如需 CloudWatch 中維度的詳細資訊，請參閱《CloudWatch 使用者指南》中的維度。

S3 Storage Lens 指標可以使用不同的維度，具體取決於指標的精細程度。例如，您可以使用 `organization_id` 維度來依 AWS Organizations ID 篩選組織層級指標。不過，您無法將此維度用於儲存貯體和帳戶層級指標。如需詳細資訊，請參閱「使用維度篩選指標」。

若要查看 S3 Storage Lens 組態可用的維度，請參閱下表。

維度	Description (描述)	儲存貯體	帳戶	組織	組織和儲存貯體	組織和帳戶
`configuration_id`	指標中報告的 S3 Storage Lens 組態儀表板名稱	.	.	.	.	.
`metrics_version`	S3 Storage Lens 指標的版本。指標版本的值固定為 `1.0`。	.	.	.	.	.
`organization_id`	指標的 AWS Organizations ID	.	.	.	.	.
`aws_account_number`	與指標相關聯的 AWS 帳戶	.	.	.	.	.
`aws_region`	指標的 AWS 區域	.	.	.	.	.

維度	Description (描述)	儲存貯體	帳戶	組織	組織和儲存貯體	組織和帳戶
bucket_name	指標中報告的 S3 儲存貯體名稱	.	.	.	.	.
storage_class	指標中報告之儲存貯體的儲存體類別	.	.	.	.	.
record_type	指標的精細程度：組織、帳戶、儲存貯體	. 儲存貯體	. 帳戶	. 儲存貯體	. 帳戶	. 組織

啟用 S3 Storage Lens 的 CloudWatch 發佈

您可以將 S3 Storage Lens 指標發佈到 Amazon CloudWatch，以在 CloudWatch 儀表板中建立統一的運作狀態檢視。您也可以使用 CloudWatch 功能 (例如警示和觸發動作、指標數學和異常偵測) 來監控 S3 Storage Lens 指標並對其採取動作。此外，CloudWatch API 操作可讓應用程式 (包括第三方供應商) 存取您的 S3 Storage Lens 指標。如需有關 CloudWatch 功能的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》。

S3 Storage Lens 指標會在擁有 S3 Storage Lens 組態的帳戶中發佈至 CloudWatch。在進階指標和建議中啟用 CloudWatch 發佈選項之後，您可以在 CloudWatch 中存取組織、帳戶和儲存貯體層級的指標。在 CloudWatch 中不提供字首層級的指標。

您可以使用 S3 主控台、Amazon S3 REST API、AWS CLI 和 AWS SDK 為新的或現有儀表板組態啟用 CloudWatch 支援。CloudWatch 發佈選項適用於升級至 S3 Storage Lens 進階指標和建議的儀表板。如需 S3 Storage Lens 進階指標和建議定價，請參閱 Amazon S3 定價。不會產生額外的 CloudWatch 指標發佈費用；不過，儀表板、警示和 API 呼叫等其他 CloudWatch 費用則適用。

若要為 S3 Storage Lens 指標啟用 CloudWatch 發佈選項，請參閱下列主題。

> **Note**
>
> S3 Storage Lens 指標是每日指標，每天向 CloudWatch 發佈一次。在 CloudWatch 中查詢 S3 Storage Lens 指標時，查詢的時段必須為 1 天 (86400 秒)。在每日 S3 Storage Lens 指標出現在 Amazon S3 主控台的 S3 Storage Lens 儀表板中之後，這些相同指標可能會需要幾個小時才會出現在 CloudWatch 中。首次啟用 S3 Storage Lens 指標的 CloudWatch 發佈選項時，您的指標最多可能需要 24 小時才能發佈到 CloudWatch。
> 目前，S3 Storage Lens 指標無法透過 CloudWatch 串流取用。

使用 S3 主控台

更新 S3 Storage Lens 儀表板時，您無法變更儀表板名稱或主要區域。您也無法變更預設儀表板的範圍，其範圍限定在整個帳戶的儲存體。

更新 S3 Storage Lens 儀表板以啟用 CloudWatch 發佈

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 S3 Storage Lens、Dashboards (儀表板)。
3. 選擇您要編輯的儀表板，然後選擇 Edit (編輯)。
4. 在 Metrics selection (指標選擇) 下，選擇 Advanced metrics and recommendations (進階指標和建議)。

   進階指標和建議需支付額外費用。進階指標和建議包含資料查詢的 15 個月時段、字首層級彙總的用量指標、依儲存貯體彙總的活動指標、CloudWatch 發佈選項，以及協助您最佳化儲存成本，並套用資料保護最佳實務的內容相關建議。如需詳細資訊，請參閱 Amazon S3 定價。
5. 在 Select Advanced metrics and recommendations features (選取進階指標和建議功能) 下，選取 CloudWatch publishing (CloudWatch 發佈)。

   > **Important**
   >
   > 如果您的組態啟用用量指標的字首彙總，則字首層級指標將不會發佈至 CloudWatch。僅儲存貯體、帳戶和組織層級 S3 Storage Lens 指標發佈至 CloudWatch。
6. 選擇 Save changes (儲存變更)。

建立啟用 CloudWatch 支援的新 S3 Storage Lens 儀表板

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 選擇 Create dashboard (建立儀表板)。
4. 在 General (一般) 下，定義下列組態選項：

   a. 針對 Dashboard name (儀表板名稱)，輸入儀表板名稱。

      儀表板名稱必須少於 65 個字元，且不得包含特殊字元或空格。建立儀表板後，您就無法變更儀表板名稱。

   b. 選擇儀表板的 Home Region (主要區域)。

      此儀表板範圍中所有包含區域的指標都會集中存放在此指定的主要區域。在 CloudWatch 中，主要區域也提供 S3 Storage Lens 指標。建立儀表板後，您就無法變更主要區域。

5. (選用) 若要新增標籤，請選擇 Add tag (新增標籤)，然後輸入標籤的 Key (鍵) 和 Value (值)。

   > **Note**
   >
   > 您最多可為儀表板組態中新增 50 個標籤。

6. 定義您的組態範圍：

   a. 如果您要建立組織層級的組態，請選擇要包含在組態中的帳戶：Include all accounts in your configuration (將全部帳戶包含在組態中) 或者 Limit the scope to your signed-in account (將範圍限制在您的登入帳戶)。

      > **Note**
      >
      > 建立包含所有帳戶的組織層級組態時，您只能包含或排除區域，而不是儲存貯體。

   b. 選擇您想要 S3 Storage Lens 在儀表板組態中包含的區域和儲存貯體，方法為執行下列動作：

      - 若要包含所有區域，請選擇 Include Regions and buckets (包含區域和儲存貯體)。
      - 若要包含特定區域，請清除 Include all Regions (包含所有區域)。在 Choose Regions to include (選擇要包含的區域) 下，選擇您想要 S3 Storage Lens 包含在儀表板中的區域。

- 若要包含特定的儲存貯體，請清除 Include all buckets (包含所有儲存貯體)。在 Choose buckets to include (選擇要包含的儲存貯體) 下，選擇您想要 S3 Storage Lens 包含在儀表板中的儲存貯體。

> **Note**
>
> 您最多可以選擇 50 個儲存貯體。

7. 針對 Metrics selection (指標選取項目)，選擇 Advanced metrics and recommendations (進階指標和建議)。

   如需有關進階指標和建議定價的詳細資訊，請參閱 Amazon S3 定價。

8. 在 Advanced metrics and recommendations features (進階指標和建議功能) 下，選取您想要啟用的選項：

   - Advanced metrics (進階指標)
   - CloudWatch 發佈

   > **Important**
   >
   > 如果為您的 S3 Storage Lens 組態啟用字首彙總，則字首層級指標將不會發佈至 CloudWatch。僅儲存貯體、帳戶和組織層級 S3 Storage Lens 指標發佈至 CloudWatch。

   - 字首彙總

   > **Note**
   >
   > 如需進階指標和建議功能的詳細資訊，請參閱 指標選擇。

9. 如果您已啟用 Advanced metrics (進階指標)，請選取要在 S3 Storage Lens 儀表板中顯示的 Advanced metrics categories (進階指標類別)：

   - 活動指標
   - Detailed status code metrics (詳細狀態碼指標)
   - Advanced cost optimization metrics (進階成本最佳化指標)
   - Advanced data protection metrics (進階資料保護指標)

如需指標類別的詳細資訊，請參閱 指標類別。如需指標的完整清單，請參閱Amazon S3 Storage Lens 指標詞彙表。

10. (選用) 設定您的指標匯出。

    如需如何設定指標匯出的詳細資訊，請參閱步驟 建立 Amazon S3 Storage Lens 儀表板。

11. 選擇 Create dashboard (建立儀表板)。

使用 AWS CLI

以下 AWS CLI 範例會使用 S3 Storage Lens 組織層級的進階指標和建議組態來啟用 CloudWatch 發佈選項。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control put-storage-lens-configuration --account-id=555555555555 --config-
id=your-configuration-id --region=us-east-1 --storage-lens-configuration=file://./
config.json

config.json
{
  "Id": "SampleS3StorageLensConfiguration", //Use this property to identify your S3
 Storage Lens configuration.
  "AwsOrg": { //Use this property when enabling S3 Storage Lens for AWS Organizations.
    "Arn": "arn:aws:organizations::123456789012:organization/o-abcdefgh"
  },
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "DetailedStatusCodesMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
```

```
      "ActivityMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "AdvancedCostOptimizationMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "DetailedStatusCodesMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "PrefixLevel":{
        "StorageMetrics":{
          "IsEnabled":true, //Mark this as false if you want only free metrics.
          "SelectionCriteria":{
            "MaxDepth":5,
            "MinStorageBytesPercentage":1.25,
            "Delimiter":"/"
          }
        }
      }
    }
  },
  "Exclude": { //Replace with "Include" if you prefer to include Regions.
    "Regions": [
      "eu-west-1"
    ],
    "Buckets": [ //This attribute is not supported for AWS Organizations-level
configurations.
      "arn:aws:s3:::source_bucket1"
    ]
  },
  "IsEnabled": true, //Whether the configuration is enabled
  "DataExport": { //Details about the metrics export
    "S3BucketDestination": {
      "OutputSchemaVersion": "V_1",
      "Format": "CSV", //You can add "Parquet" if you prefer.
      "AccountId": "111122223333",
      "Arn": "arn:aws:s3:::destination-bucket-name", // The destination bucket for your
metrics export must be in the same Region as your S3 Storage Lens configuration.
      "Prefix": "prefix-for-your-export-destination",
      "Encryption": {
        "SSES3": {}
      }
    },
    "CloudWatchMetrics": {
```

```
      "IsEnabled": true //Mark this as false if you want to export only free metrics.
    }
  }
}
```

使用適用於 Java 的 AWS SDK

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.CloudWatchMetrics;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateAndUpdateDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
```

```
        String exportAccountId = "Destination Account ID";
        String exportBucketArn = "arn:aws:s3:::destBucketName"; // The destination
 bucket for your metrics export must be in the same Region as your S3 Storage Lens
 configuration.
        String awsOrgARN = "arn:aws:organizations::123456789012:organization/o-
abcdefgh";
        Format exportFormat = Format.CSV;

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
 AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
 DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
 PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
 AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
 DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            Include include = new Include()
                    .withBuckets(Arrays.asList("arn:aws:s3:::bucketName"))
                    .withRegions(Arrays.asList("us-west-2"));

            StorageLensDataExportEncryption exportEncryption = new
 StorageLensDataExportEncryption()
                    .withSSES3(new SSES3());
```

```
            S3BucketDestination s3BucketDestination = new S3BucketDestination()
                    .withAccountId(exportAccountId)
                    .withArn(exportBucketArn)
                    .withEncryption(exportEncryption)
                    .withFormat(exportFormat)
                    .withOutputSchemaVersion(OutputSchemaVersion.V_1)
                    .withPrefix("Prefix");
            CloudWatchMetrics cloudWatchMetrics = new CloudWatchMetrics()
                    .withIsEnabled(true);
            StorageLensDataExport dataExport = new StorageLensDataExport()
                    .withCloudWatchMetrics(cloudWatchMetrics)
                    .withS3BucketDestination(s3BucketDestination);

            StorageLensAwsOrg awsOrg = new StorageLensAwsOrg()
                    .withArn(awsOrgARN);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withInclude(include)
                    .withDataExport(dataExport)
                    .withAwsOrg(awsOrg)
                    .withIsEnabled(true);

            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
```

```
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用 REST API

若要使用 Amazon S3 REST API 來啟用 CloudWatch 發佈選項，您可以使用 PutStorageLensConfiguration。

後續步驟

啟用 CloudWatch 發佈選項後，您可以在 CloudWatch 中存取您的 S3 Storage Lens 指標。您也可以利用 CloudWatch 功能，在 CloudWatch 中監控和分析您的 S3 Storage Lens 資料。如需詳細資訊，請參閱下列主題：

- S3 Storage Lens 指標和維度
- 使用 CloudWatch 中的 S3 Storage Lens 指標

使用 CloudWatch 中的 S3 Storage Lens 指標

您可以將 S3 Storage Lens 指標發佈到 Amazon CloudWatch，以在 CloudWatch 儀表板中建立統一的運作狀態檢視。您也可以使用 CloudWatch 功能 (例如警示和觸發動作、指標數學和異常偵測) 來監控 S3 Storage Lens 指標並對其採取動作。此外，CloudWatch API 操作可讓應用程式 (包括第三方供應商) 存取您的 S3 Storage Lens 指標。如需有關 CloudWatch 功能的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》。

您可以使用 Amazon S3 主控台、Amazon S3 REST API、AWS CLI 和 AWS SDK 為新的或現有儀表板組態啟用 CloudWatch 發佈選項。CloudWatch 發佈選項適用於升級至 S3 Storage Lens 進階指標和建議的儀表板。如需 S3 Storage Lens 進階指標和建議定價，請參閱 Amazon S3 定價。不會產生額外的 CloudWatch 指標發佈費用；不過，儀表板、警示和 API 呼叫等其他 CloudWatch 費用則適用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

S3 Storage Lens 指標會在擁有 S3 Storage Lens 組態的帳戶中發佈至 CloudWatch。在進階指標和建議中啟用 CloudWatch 發佈選項之後，您可以在 CloudWatch 中存取組織、帳戶和儲存貯體層級的指標。在 CloudWatch 中不提供字首層級的指標。

> **Note**
>
> S3 Storage Lens 指標是每日指標，每天向 CloudWatch 發佈一次。在 CloudWatch 中查詢 S3 Storage Lens 指標時，查詢的時段必須為 1 天 (86400 秒)。在每日 S3 Storage Lens 指標出現在 Amazon S3 主控台的 S3 Storage Lens 儀表板中之後，這些相同指標可能會需要幾個小時才會出現在 CloudWatch 中。首次啟用 S3 Storage Lens 指標的 CloudWatch 發佈選項時，您的指標最多可能需要 24 小時才能發佈到 CloudWatch。
> 目前，S3 Storage Lens 指標無法透過 CloudWatch 串流取用。

如需在 CloudWatch 中使用 S3 Storage Lens 指標的詳細資訊，請參閱下列主題。

主題

- 使用 CloudWatch 儀表板
- 設定警示、觸發動作和使用異常偵測
- 使用維度篩選指標
- 使用指標數學計算新指標
- 在圖形中使用搜尋表達式

使用 CloudWatch 儀表板

您可以使用 CloudWatch 儀表板來監控 S3 Storage Lens 指標與其他應用程式指標，並建立統一的運作狀態檢視。儀表板是 CloudWatch 主控台中的可自訂首頁，您可於單一檢視中監控您的資源。

CloudWatch 具有廣泛的許可控制，不支援將存取限制為指標或維度的特定集。您帳戶或組織中具有 CloudWatch 存取權的使用者將可存取啟用 CloudWatch 支援選項之所有 S3 Storage Lens 組態的指標。您無法像在 S3 Storage Lens 中一樣管理特定儀表板的許可。如需 CloudWatch 許可的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的管理 CloudWatch 資源的存取許可。

如需使用 CloudWatch 儀表板和設定許可的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的使用 Amazon CloudWatch 儀表板和共用 CloudWatch 儀表板。

設定警示、觸發動作和使用異常偵測

您可以設定 CloudWatch 警示，以便在 CloudWatch 中監視 S3 Storage Lens 指標，並在超出閾值時採取動作。例如，您可以設定警示，當 Incomplete Multipart Upload Bytes (未完成的分段上傳位元組) 指標連續三天超過 1 GB 時，傳送 Amazon SNS 通知。

您也可以啟用異常偵測，以持續分析您的 S3 Storage Lens 指標、判斷正常基準以及顯露異常情況。您可以建立以指標預期值為基礎的異常偵測警示。例如，您可以監控 Object Lock Enabled Bytes (已啟用物件鎖定的位元組) 指標的異常情況，以偵測未經授權即移除物件鎖定設定的情況。

如需詳細資訊和範例，請參閱《Amazon CloudWatch 使用者指南》中的使用 Amazon CloudWatch 警示和從圖表上的指標建立警示。

使用維度篩選指標

您可以使用維度來篩選 CloudWatch 主控台中的 S3 Storage Lens 指標。例如，您可以依 `configuration_id`、`aws_account_number`、`aws_region`、`bucket_name` 等進行篩選。

S3 Storage Lens 支援每個帳戶的多種儀表板組態。這意味著不同的組態可以包含相同的儲存貯體。當這些指標在 CloudWatch 發佈時，儲存貯體在 CloudWatch 中會有重複的指標。若只要在 CloudWatch 中檢視特定 S3 Storage Lens 組態的指標，您可以使用 `configuration_id` 維度。依 `configuration_id` 篩選時，您只會看到與您所識別之組態相關聯的指標。

如需依組態 ID 篩選的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的搜尋可用的指標。

使用指標數學計算新指標

您可以使用指標數學，以查詢多個 S3 Storage Lens 指標，並使用數學表達式根據這些指標來建立新的時間序列。例如，您可以從「物件計數」中減去「加密物件」，為未加密物件建立新指標。您也可以建立一個指標，將 `StorageBytes` 除以 `ObjectCount` 而取得平均物件大小，或者將 `BytesDownloaded` 除以 `StorageBytes` 而取得一天存取的位元組數。

如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的使用指標數學。

在圖形中使用搜尋表達式

使用 S3 Storage Lens 指標，您可以建立搜尋表達式。例如，您可以針對所有名為 IncompleteMultipartUploadStorageBytes 的指標建立搜尋表達式，然後將 SUM 新增至該表達式。使用此搜尋表達式，您可以在單一指標中查看儲存體所有維度的未完成分段上傳位元組總數。

此範例顯示您將用來為所有名為 IncompleteMultipartUploadStorageBytes 的指標建立搜尋表達式的語法。

```
SUM(SEARCH('{AWS/S3/Storage-
Lens,aws_account_number,aws_region,configuration_id,metrics_version,record_type,storage_class}
 MetricName="IncompleteMultipartUploadStorageBytes"', 'Average',86400))
```

如需此語法的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 CloudWatch 搜尋表達式語法。若要使用搜尋表達式建立 CloudWatch 圖表，請參閱《Amazon CloudWatch 使用者指南》中的使用搜尋表達式建立 CloudWatch 圖表。

Amazon S3 Storage Lens 指標使用案例

您可以使用 Amazon S3 Storage Lens 儀表板，視覺化洞見與趨勢、標記異常值，以及接收建議。S3 Storage Lens 指標會組織成符合重要使用案例的類別。您可以使用這些指標執行下列動作：

- 識別成本最佳化機會
- 套用資料保護最佳實務
- 套用存取管理最佳實務
- 改善應用程式工作負載的效能

例如，使用成本最佳化指標，您可以識別哪些機會可降低 Amazon S3 儲存成本。您可以識別哪些儲存貯體具有超過 7 天的分段上傳，或哪些儲存貯體正在累積非目前版本。

同樣地，您可以使用資料保護指標，來識別組織內未遵循資料保護最佳實務的儲存貯體。例如，您可以識別哪些儲存貯體未使用 AWS Key Management Service 金鑰 (SSE-KMS) 進行預設加密或未啟用 S3 版本控制。

使用 S3 Storage Lens 存取管理指標，您可以識別 S3 物件擁有權的儲存貯體設定，以便可將存取控制清單 (ACL) 許可遷移至儲存貯體政策並停用 ACL。

如果已啟用 S3 Storage Lens 進階指標，您可以使用詳細的狀態碼指標，來取得成功或失敗請求的計數，您可以用來針對存取或效能問題進行疑難排解。

使用進階指標，您還可以存取其他成本最佳化和資料保護指標，可用來識別哪些機會可進一步降低整體 S3 儲存成本，並更好地符合保護資料的最佳實務。例如，進階成本最佳化指標包括生命週期規則計數，您可以用來您可以識別沒有生命週期規則的儲存貯體，這些規則可使超過 7 天未完成的分段上傳過期。進階資料保護指標包括複寫規則計數。

如需指標類別的詳細資訊，請參閱 指標類別。如需 S3 Storage Lens 指標的完整清單，請參閱 Amazon S3 Storage Lens 指標詞彙表。

主題

- 使用 Amazon S3 Storage Lens 最佳化儲存成本
- 使用 S3 Storage Lens 保護您的資料
- 使用 S3 Storage Lens 稽核物件擁有權設定

- 使用 S3 Storage Lens 指標來改善效能

使用 Amazon S3 Storage Lens 最佳化儲存成本

您可以使用 S3 Storage Lens 成本最佳化指標來降低 S3 儲存體的整體成本。成本最佳化指標可協助您確認您已有效地且根據最佳實務設定 Amazon S3 成本。例如，您可以識別下列成本最佳化機會：

- 超過 7 天未完成分段上傳的儲存貯體
- 累積眾多非目前版本的儲存貯體
- 沒有生命週期規則來中止未完成分段上傳的儲存貯體
- 沒有生命週期規則使非目前版本物件過期的儲存貯體
- 沒有生命週期規則來將物件轉移到不同儲存體類別的儲存貯體

然後，您可以使用此資料，將其他生命週期規則新增至儲存貯體。

下列範例示範如何在 S3 Storage Lens 儀表板中使用成本最佳化指標，以最佳化您的儲存成本。

主題

- 識別您的最大 S3 儲存貯體
- 瞭解冷 Amazon S3 儲存貯體
- 尋找不完整的分段上傳
- 減少保留的非目前版本數量
- 識別沒有生命週期規則的儲存貯體，並檢閱生命週期規則計數

識別您的最大 S3 儲存貯體

您支付在 S3 儲存貯體中存放物件的費用。向您收取費用的費率取決於物件大小、存放物件多久以及其儲存體方案。使用 S3 Storage Lens，您可以集中檢視帳戶中的所有儲存貯體。若要查看組織所有帳戶中的所有儲存貯體，您可以設定 AWS Organizations 層級 S3 Storage Lens 儀表板。從此儀表板檢視中，您可以識別最大的儲存貯體。

步驟 1：識別您的最大儲存貯體

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。

3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板。

   儀表板開啟時，您可以看到 S3 Storage Lens 收集指標的最新日期。您的儀表板一律會載入至具有可用指標的最新日期。

4. 若要透過所選日期範圍的 Total storage (儲存體總數) 指標來查看最大儲存貯體的排名，請向下捲動至 Top N overview for date (日期的前 N 個概觀) 區段，。

   您可以切換排序順序以顯示最小儲存貯體。您也可以調整 Metric (指標) 選擇，依據任何可用指標對儲存貯體進行排名。Top N overview for date (日期的前 N 個概觀) 區段也會顯示前一天或前一週的變更百分比以及火花線，來視覺化趨勢。此趨勢是 14 天趨勢 (若為免費指標)，以及 30 天趨勢 (若為進階指標和建議)。

   > **Note**
   >
   > 有了 S3 Storage Lens 進階指標和建議，指標可供查詢使用 15 個月。如需更多詳細資訊，請參閱 指標選擇。

5. 如需有關儲存貯體的詳細洞見，請向上捲動至頁面頂端，然後選擇 Bucket (儲存貯體) 索引標籤。

   在 Bucket (儲存貯體) 索引標籤上，您可以查看詳細資訊，例如最近的成長率、平均物件大小、最大的字首以及物件數量。

步驟 2：導覽到您的儲存貯體並進行調查

在識別了您的最大 S3 儲存貯體之後，您可以導覽至 S3 主控台內的每個儲存貯體，以檢視儲存貯體中的物件、了解其相關聯的工作負載，以及識別其內部擁有者。您可以聯絡儲存貯體擁有者，以了解是否預期此成長，或此成長是否需要進一步的監控與控制。

瞭解冷 Amazon S3 儲存貯體

如果已啟用 S3 Storage Lens 進階指標，您可以使用活動指標來瞭解您的 S3 儲存貯體有多久未使用。「冷」儲存貯體是不再存取其儲存的儲存貯體 (或非常少存取)。缺乏活動通常表示不經常存取儲存貯體的物件。

諸如 GET 請求和下載位元組數等指標指示每天存取儲存貯體的頻率。若要瞭解存取模式的一致性，以及找出完全不再存取的儲存貯體，您可以在數個月內趨勢化此資料。計算為下載位元組/儲存總量的擷取率指標指示每日存取的儲存貯體中的儲存比例。

> **Note**
>
> 如果同一個物件在一天內下載多次，則會複製下載位元組。

先決條件

若要在 S3 Storage Lens 儀表板中查看活動指標，您必須啟用 S3 Storage Lens Advanced metrics and recommendations (進階指標和建議)，然後選取 Activity metrics (活動指標)。如需更多詳細資訊，請參閱 建立和更新 Amazon S3 Storage Lens 儀表板。

步驟 1：識別作用中儲存貯體

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板。
4. 選擇 Bucket (儲存貯體) 索引標籤，然後向下捲動至 Bubble analysis by buckets for date (日期的依儲存貯體進行氣泡分析) 區段。

   在 Bubble analysis by buckets for date (日期的依儲存貯體進行氣泡分析) 區段中，您可以使用任意三個指標在多個維度上繪製儲存貯體，以表示氣泡的 X-axis (X 軸)、Y-axis (Y 軸) 以及 Size (大小)。
5. 若要尋找變冷的儲存貯體，請針對 X-axis (X 軸)、Y-axis (Y 軸) 和 Size (大小)，選擇 Total storage (儲存體總量)、% retrieval rate (% 擷取率) 和 Average object size (平均物件大小) 指標。
6. 在 Bubble analysis by buckets for date (日期的依儲存貯體進行氣泡分析) 區段中，尋找任何擷取率為零 (或接近零) 且相對儲存體大小更大的儲存貯體，並選擇代表儲存貯體的氣泡。

   將出現一個方塊，其中包含更精細洞見的選項。執行下列任意一項：

   a. 若要更新 Bucket (儲存貯體) 索引標籤，僅顯示所選儲存貯體的指標，請選擇 Drill down (向下切入)，然後選擇 Apply (套用)。

   b. 若要依帳戶、AWS 區域、儲存體類別或儲存貯體彙總您的儲存貯體層級資料，請選擇 Analyze by (分析依據)，然後對 Dimension (維度) 做出選擇。例如，若要依儲存體類別彙總，請選擇 Storage class (儲存體類別) 作為 Dimension (維度)。

要查找變冷的儲存貯體，請使用儲存總量、擷取率 (%) 以及平均物件大小指標進行氣泡分析。尋找任何擷取率為零 (或接近零) 且相對儲存容量較大的儲存貯體。

儀表板的 Bucket (儲存貯體) 索引標籤會更新，以顯示所選彙總或篩選條件的資料。如果您依儲存體類別或另一個維度彙總，則該新索引標籤會在儀表板中開啟 (例如，Storage class (儲存體類別) 索引標籤)。

步驟 2：調查冷儲存貯體

從這裡，您可以識別帳戶或組織中冷儲存貯體的擁有者，並確定是否仍然需要該儲存體。然後，您可以透過設定這些儲存貯體的生命週期過期組態，或在其中一個 Amazon S3 Glacier 儲存體類別 中封存資料來最佳化成本。

為了避免發生冷儲存貯體問題，可以使用 S3 生命週期組態對您的儲存貯體自動轉換資料，或者可以啟用使用 S3 Intelligent-Tiering 自動封存。

您也可以使用步驟 1 來識別熱儲存貯體。然後，您可以確保這些儲存貯體使用正確的 S3 儲存體類別，以確保它們在效能和成本方面能夠以最有效率的方式提供請求。

尋找不完整的分段上傳

您可以使用分段上傳，將非常大的物件 (最多 5 TB) 作為一組不同組件進行上傳，以改善輸送量並加快網路問題復原的速度。如果分段上傳程序未完成，未完成的部分仍會留在儲存貯體中 (處於無法使用狀態)。這些未完成的部分會產生儲存成本，直到上傳程序完成或將未完成的部分移除為止。如需更多詳細資訊，請參閱 使用分段上傳來上傳和複製物件。

透過 S3 Storage Lens，您可以識別帳戶或整個組織中不完整分段上傳位元組的數量，包括超過 7 天未完成的分段上傳。如需未完成分段上傳指標的完整清單，請參閱 Amazon S3 Storage Lens 指標詞彙表。

作為最佳實務，建議您設定生命週期規則，使超過特定天數未完成的分段上傳過期。當您建立生命週期規則，使未完成的分段上傳過期時，我們建議您使用 7 天作為不錯的起點。

步驟 1：檢閱未完成分段上傳的整體趨勢

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。

3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板。
4. 在 Snapshot for date (日期的快照) 區段中，於 Metrics categories (指標類別) 下，選擇 Cost optimization (成本最佳化)。

   Snapshot for date (日期的快照) 區段會更新，以顯示 Cost optimization (成本最佳化) 指標，其中包括 Incomplete multipart upload bytes greater than 7 days old (超過 7 天未完成的分段上傳位元組)。

   在 S3 Storage Lens 儀表板的任何圖表中，您可以看到未完成分段上傳的指標。您可以使用這些指標，進一步評估未完成分段上傳位元組對儲存體的影響，包括它們對整體成長趨勢的貢獻。您也可以使用 Account (帳戶)、AWS 區域、Bucket (儲存貯體) 或 Storage class (儲存體類別) 索引標籤，向下切入至更深層的彙總，以進行更深入的資料分析。如需範例，請參閱 瞭解冷 Amazon S3 儲存貯體。

步驟 2：識別具有最多未完成分段上傳位元組，但沒有生命週期規則來中止未完成分段上傳的儲存貯體

先決條件

若要在 S3 Storage Lens 儀表板中查看 Abort incomplete multipart upload lifecycle rule count (中止未完成分段上傳生命週期規則計數) 指標，您必須啟用 S3 Storage Lens Advanced metrics and recommendations (進階指標和建議)，然後選取 Advanced cost optimization metrics (進階成本最佳化指標)。如需更多詳細資訊，請參閱 建立和更新 Amazon S3 Storage Lens 儀表板。

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板。
4. 若要識別累積超過 7 天未完成分段上傳的特定儲存貯體，請移至 Top N overview for date (日期的前 N 個概觀) 區段。

   依預設，Top N overview for date (日期的前 N 個概觀) 區段會顯示前 3 個儲存貯體的指標。您可以在 Top N (前 N 個) 欄位中增加或減少儲存貯體數目。Top N overview for date (日期的前 N 個概觀) 區段也會顯示前一天或前一週的變更百分比以及火花線，來視覺化趨勢。(此趨勢是 14 天趨勢 (若為免費指標)，以及 30 天趨勢 (若為進階指標和建議)。)

> **Note**
>
> 有了 S3 Storage Lens 進階指標和建議，指標可供查詢使用 15 個月。如需更多詳細資訊，請參閱 指標選擇。

5. 針對 Metric (指標)，請在 Cost optimization (成本最佳化) 類別中選擇 Incomplete multipart upload bytes greater than 7 days old (超過 7 天未完成的分段上傳位元組)。

   在 Top number buckets (前幾個儲存貯體) 下，您可以看到哪些儲存貯體具有最多超過 7 天未完成的分段上傳儲存體位元組。

6. 若要檢視未完成分段上傳的更詳細儲存貯體層級指標，請捲動至頁面頂端，然後選擇 Bucket (儲存貯體) 索引標籤。

7. 向下捲動至 Buckets (儲存貯體) 區段。針對 Metrics categories (指標類別)，請選取 Cost optimization (成本最佳化)。然後清除 Summary (摘要)。

   Buckets (儲存貯體) 清單會更新，以顯示所顯示儲存貯體的所有可用 Cost optimization (成本最佳化) 指標。

8. 若要篩選 Buckets (儲存貯體) 清單，僅顯示特定成本最佳化指標，請選擇偏好設定圖示 ( )。

9. 清除所有成本最佳化指標的切換，直到留下選取的 Incomplete multipart upload bytes greater than 7 days old (超過 7 天未完成的分段上傳位元組) 和 Abort incomplete multipart upload lifecycle rule count (中止未完成分段上傳生命週期規則計數) 為止。

10. (選用) 在 Page size (頁面大小) 下，選擇要在清單中顯示的儲存貯體數目。

11. 選擇 Confirm (確認)。

    Buckets (儲存貯體) 清單會更新，以顯示未完成分段上傳和生命週期規則計數的儲存貯體層級指標。您可以使用此資料，來識別哪些儲存貯體具有最多超過 7 天未完成的分段上傳位元組，且缺少生命週期規則來中止未完成的分段上傳。然後，您可以導覽至 S3 主控台中的這些儲存貯體，然後新增生命週期規則來刪除已放棄的未完成分段上傳。

步驟 3：新增生命週期規則，以在 7 天後刪除未完成的分段上傳

若要自動管理不完整的分段上傳，您可以使用 S3 主控台建立生命週期組態，以在指定天數後，使儲存貯體中未完成的分段上傳位元組過期。如需更多詳細資訊，請參閱 設定儲存貯體生命週期組態，以刪除不完整的分段上傳。

減少保留的非目前版本數量

啟用時，S3 版本控制會保留相同物件的多個不同複本，您可以用來在物件遭到意外刪除或覆寫時快速復原資料。如果已啟用 S3 版本控制，但未設定生命週期規則來轉移非目前版本或使其過期，則可能會累積大量先前的非目前版本，這可能會影響儲存成本。如需更多詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。

步驟 1：識別具有最多非目前物件版本的儲存貯體

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板。
4. 在 Snapshot for date (日期的快照) 區段中，於 Metric categories (指標類別) 下，選擇 Cost optimization (成本最佳化)。

   Snapshot for date (日期的快照) 區段會更新，以顯示 Cost optimization (成本最佳化) 指標，其中包括 % noncurrent version bytes (% 非目前版本位元組) 的指標。% noncurrent version bytes (非目前版本位元組百分比) 指標代表在儀表板的範圍內和針對所選日期，儲存體位元組總數對非目前版本所佔的比例。

   > **Note**
   >
   > 如果您的 % noncurrent version bytes (非目前版本位元組百分比) 大於帳戶層級儲存體的 10%，您可能存放了太多物件版本。

5. 若要識別累積大量非目前版本的特定儲存貯體，請執行下列動作：

   a. 向下捲動至 Top N overview for date (日期的前 N 個概觀) 區段。針對 Top N (前 N 個)，輸入您要查看其資料的儲存貯體數目。

   b. 針對 Metric (指標)，選擇 % noncurrent version bytes (% 非目前版本位元組)。

      在 Top number buckets (前幾個儲存貯體) 下，您可以看到哪些儲存貯體 (針對您指定的數目) 具有最高的 % noncurrent version bytes (% 非目前版本位元組)。Top N overview for date (日期的前 N 個概觀) 區段也會顯示前一天或前一週的變更百分比以及火花線，來視覺化趨勢。此趨勢是 14 天趨勢 (若為免費指標)，以及 30 天趨勢 (若為進階指標和建議)。

> **Note**
>
> 有了 S3 Storage Lens 進階指標和建議，指標可供查詢使用 15 個月。如需更多詳細資訊，請參閱 指標選擇。

c. 若要檢視非目前物件版本的更詳細儲存貯體層級指標，請捲動至頁面頂端，然後選擇 Bucket (儲存貯體) 索引標籤。

   在 S3 Storage Lens 儀表板的任何圖表或視覺效果中，您可以向下切入至更深層的彙總，方法為使用 Account (帳戶)、AWS 區域、Storage class (儲存體類別) 或 Bucket (儲存貯體) 索引標籤。如需範例，請參閱 瞭解冷 Amazon S3 儲存貯體。

d. 在 Buckets (儲存貯體) 區段中，針對 Metric categories (指標類別)，選取 Cost optimization (成本最佳化)。然後，清除 Summary (摘要)。

   您現在可以看到 % noncurrent version bytes (% 非目前版本位元組) 指標，以及與非目前版本相關的其他指標。

步驟 2：識別缺少用於管理非目前版本的轉移和過期生命週期規則的儲存貯體

先決條件

若要在 S3 Storage Lens 儀表板中查看 Noncurrent version transition lifecycle rule count (非目前版本轉移生命週期規則計數) 和 Noncurrent version expiration lifecycle rule count (非目前版本過期生命週期規則計數) 指標，您必須啟用 S3 Storage Lens Advanced metrics and recommendations (進階指標和建議)，然後選取 Advanced cost optimization metrics (進階成本最佳化指標)。如需更多詳細資訊，請參閱 建立和更新 Amazon S3 Storage Lens 儀表板。

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板。
4. 在 Storage Lens 儀表板中，選擇 Bucket (儲存貯體) 索引標籤。
5. 向下捲動至 Buckets (儲存貯體) 區段。針對 Metrics categories (指標類別)，請選取 Cost optimization (成本最佳化)。然後清除 Summary (摘要)。

   Buckets (儲存貯體) 清單會更新，以顯示所顯示儲存貯體的所有可用 Cost optimization (成本最佳化) 指標。

6. 若要篩選 Buckets (儲存貯體) 清單，僅顯示特定成本最佳化指標，請選擇偏好設定圖示 ()。
7. 清除所有成本最佳化指標的切換，直到只留下以下選取的項目為止：
   - % noncurrent version bytes (% 非目前版本位元組)
   - Noncurrent version transition lifecycle rule count (非目前版本轉移生命週期規則計數)
   - Noncurrent version expiration lifecycle rule count (非目前版本過期生命週期規則計數)
8. (選用) 在 Page size (頁面大小) 下，選擇要在清單中顯示的儲存貯體數目。
9. 選擇 Confirm (確認)。

   Buckets (儲存貯體) 清單會更新，以顯示非目前版本位元組和非目前版本生命週期規則計數的指標。您可以使用此資料，來識別哪些儲存貯體具有高百分比的非目前版本位元組，但缺少轉移和過期生命週期規則。然後，您可以導覽至 S3 主控台中的這些儲存貯體，並將生命週期規則新增至這些儲存貯體。

步驟 3：新增生命週期規則來轉移非目前物件版本或使其過期

在確定了哪些儲存貯體需要進一步調查之後，您可以導覽至 S3 主控台內的儲存貯體，並新增生命週期規則，以在指定天數後使非目前版本過期。或者，若要在保留非目前版本的同時降低成本，您可以設定生命週期規則，將非目前版本轉移至其中一個 Amazon S3 Glacier 儲存體類別。如需更多詳細資訊，請參閱 範例 6：為已啟用版本控制的儲存貯體指定生命週期規則。

識別沒有生命週期規則的儲存貯體，並檢閱生命週期規則計數

S3 Storage Lens 提供 S3 生命週期規則計數指標，您可以用來識別缺少生命週期規則的儲存貯體。若要尋找沒有生命週期規則的儲存貯體，您可以使用 Total buckets without lifecycle rules (沒有生命週期規則的儲存貯體總數) 指標。沒有 S3 生命週期組態的儲存貯體可能具有您不再需要或者可以遷移到成本較低的儲存體類別的儲存體。您也可以使用生命週期規則計數指標，來識別缺少特定類型生命週期規則 (例如過期或轉移規則) 的儲存貯體。

先決條件

若要在 S3 Storage Lens 儀表板中查看生命週期規則計數，以及 Total buckets without lifecycle rules (沒有生命週期規則的儲存貯體總數) 指標，您必須啟用 S3 Storage Lens Advanced metrics and recommendations (進階指標和建議)，然後選取 Advanced cost optimization metrics (進階成本最佳化指標)。如需更多詳細資訊，請參閱 建立和更新 Amazon S3 Storage Lens 儀表板。

步驟 1：識別沒有生命週期規則的儲存貯體

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板。
4. 若要識別沒有生命週期規則的特定儲存貯體，請向下捲動至 Top N overview for date (日期的前 N 個概觀) 區段。

   依預設，Top N overview for date (日期的前 N 個概觀) 區段會顯示前 3 個儲存貯體的指標。在 Top N (前 N 個) 欄位中，您可以增加儲存貯體的數目。Top N overview for date (日期的前 N 個概觀) 區段也會顯示前一天或前一週的變更百分比以及火花線，來視覺化趨勢。此趨勢是 14 天趨勢 (若為免費指標)，以及 30 天趨勢 (若為進階指標和建議)。

   > **Note**
   >
   > 有了 S3 Storage Lens 進階指標和建議，指標可供查詢使用 15 個月。如需更多詳細資訊，請參閱 指標選擇。

5. 針對 Metric (儲存貯體)，從 Cost optimization (成本最佳化) 類別中選擇 Total buckets without lifecycle rules (沒有生命週期規則的儲存貯體總數)。
6. 檢閱 Total buckets without lifecycle rules (沒有生命週期規則的儲存貯體總數) 的下列資料:

   - Top number accounts (前幾個帳戶) - 查看哪些帳戶具有最多沒有生命週期規則的儲存貯體。
   - Top number Regions (前幾個區域) - 依區域檢視沒有生命週期規則的儲存貯體明細。
   - Top number buckets (前幾個儲存貯體) - 查看哪些儲存貯體沒有生命週期規則。

   在 S3 Storage Lens 儀表板的任何圖表或視覺效果中，您可以向下切入至更深層的彙總，方法為使用 Account (帳戶)、AWS 區域、Storage class (儲存體類別) 或 Bucket (儲存貯體) 索引標籤。如需範例，請參閱 瞭解冷 Amazon S3 儲存貯體。

   在識別了哪些儲存貯體沒有生命週期規則之後，您也可以檢閱儲存貯體的特定生命週期規則計數。

步驟 2：檢閱儲存貯體的生命週期規則計數

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板。
4. 在 S3 Storage Lens 儀表板中，選擇 Bucket (儲存貯體) 索引標籤。
5. 向下捲動至 Buckets (儲存貯體) 區段。在 Metrics categories (指標類別) 下，選取 Cost optimization (成本最佳化)。然後清除 Summary (摘要)。

   Buckets (儲存貯體) 清單會更新，以顯示所顯示儲存貯體的所有可用 Cost optimization (成本最佳化) 指標。
6. 若要篩選 Buckets (儲存貯體) 清單，僅顯示特定成本最佳化指標，請選擇偏好設定圖示 ()。
7. 清除所有成本最佳化指標的切換，直到只留下以下選取的項目為止：
   - Transition lifecycle rule count (轉移生命週期規則計數)
   - Expiration lifecycle rule count (過期生命週期規則計數)
   - Noncurrent version transition lifecycle rule count (非目前版本轉移生命週期規則計數)
   - Noncurrent version expiration lifecycle rule count (非目前版本過期生命週期規則計數)
   - Abort incomplete multipart upload lifecycle rule count (中止未完成分段上傳生命週期規則計數)
   - Total lifecycle rule count (生命週期規則總計數)
8. (選用) 在 Page size (頁面大小) 下，選擇要在清單中顯示的儲存貯體數目。
9. 選擇 Confirm (確認)。

   Buckets (儲存貯體) 清單會更新，以顯示儲存貯體的生命週期規則計數指標。您可以使用此資料，來識別沒有生命週期規則的儲存貯體，或缺少特定類型生命週期規則的儲存貯體，例如過期或轉移規則。然後，您可以導覽至 S3 主控台中的這些儲存貯體，並將生命週期規則新增至這些儲存貯體。

步驟 3：新增生命週期規則

在識別了沒有生命週期規則的儲存貯體之後，您可以新增生命週期規則。如需詳細資訊，請參閱在值區上設定生命週期組態及S3 生命週期組態範例。

使用 S3 Storage Lens 保護您的資料

您可以使用 Amazon S3 Storage Lens 資料保護指標，來識別尚未套用資料保護最佳實務的儲存貯體。您可以使用這些指標採取行動，並套用符合最佳實務的標準設定，以在您的帳戶或組織中跨儲存貯體保

護您的資料。例如，您可以使用資料保護指標，來識別未使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 進行預設加密的儲存貯體，或使用 AWS 第 2 版簽署程序 (SigV2) 的請求。

下列使用案例提供策略，用於使用 S3 Storage Lens 儀表板來識別異常值以及跨 S3 儲存貯體套用資料保護最佳實務。

主題

- 識別未使用伺服器端加密搭配 AWS KMS 進行預設加密 (SSE-KMS) 的儲存貯體
- 識別已啟用 S3 版本控制的儲存貯體
- 識別使用 AWS 第 2 版簽署程序 (SigV2) 的請求
- 計算每個儲存貯體的複寫規則總數
- 識別物件鎖定位元組的百分比

識別未使用伺服器端加密搭配 AWS KMS 進行預設加密 (SSE-KMS) 的儲存貯體

使用 Amazon S3 預設加密，您可以設定 S3 儲存貯體的預設加密行為。如需更多詳細資訊，請參閱 the section called “設定預設儲存貯體加密”。

您可以使用 SSE-KMS enabled bucket count (已啟用 SSE-KMS 的儲存貯體計數) 和 % SSE-KMS enabled buckets (% 已啟用 SSE-KMS 的儲存貯體) 指標，識別使用伺服器端加密搭配 AWS KMS 金鑰 (SSE-KMS) 進行預設加密的儲存貯體。S3 Storage Lens 也提供未加密位元組、未加密物件、加密位元組和加密物件的指標。如需指標的完整清單，請參閱Amazon S3 Storage Lens 指標詞彙表。

您可以在一般加密指標的內容中分析 SSE-KMS 加密指標，以識別未使用 SSE-KMS 的儲存貯體。如果您想要針對帳戶或組織中的所有儲存貯體使用 SSE-KMS，則可以更新這些儲存貯體的預設加密設定，以使用 SSE-KMS。除了 SSE-KMS 之外，您還可以使用伺服器端加密搭配 Amazon S3 受管金鑰 (SSE-S3) 或客戶提供的金鑰 (SSE-C)。如需更多詳細資訊，請參閱 使用加密來保護資料。

步驟 1：識別哪些儲存貯體正在使用 SSE-KMS 進行預設加密

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板名稱。
4. 在 Trends and distributions (趨勢和分佈) 區段中，選擇 % SSE-KMS enabled bucket count (% 已啟用 SSE-KMS 的儲存貯體計數) 作為主要指標，以及選擇 % encrypted bytes (% 加密位元組) 作為次要指標。

Trend for date (<日期> 的趨勢) 圖表會更新，以顯示 SSE-KMS 和加密位元組的趨勢。

5. 若要檢視 SSE-KMS 的更精細儲存貯體層級洞見：

   a. 在圖表上選擇一點。將出現一個方塊，其中包含更精細洞見的選項。

   b. 選擇 Buckets (儲存貯體) 維度。接著選擇 Apply (套用)。

6. 在 Distribution by buckets for date (<日期> 依儲存貯體的分佈) 圖表中，選擇 SSE-KMS enabled bucket count (已啟用 SSE-KMS 的儲存貯體計數) 指標。

7. 您現在可以看到哪些儲存貯體已啟用 SSE-KMS，以及哪些儲存貯體未啟用它。

步驟 2：更新儲存貯體預設加密設定

確定了哪些儲存貯體在 % encrypted bytes (% 加密位元組) 的內容中使用 SSE-KMS，您就可以識別未使用 SSE-KMS 的儲存貯體。然後，您可以選擇性地導覽至 S3 主控台內的這些儲存貯體，並更新其預設加密設定，以使用 SSE-KMS 或 SSE-S3。如需更多詳細資訊，請參閱 設定預設加密。

識別已啟用 S3 版本控制的儲存貯體

啟用時，S3 版本控制功能會保留相同物件的多個版本，以便在物件意外刪除或覆寫時快速復原資料。您可以使用 Versioning-enabled bucket count (已啟用版本控制的儲存貯體計數) 指標，查看哪些儲存貯體使用 S3 版本控制。然後，您可以在 S3 主控台中採取動作，針對其他儲存貯體啟用 S3 版本控制。

步驟 1：識別已啟用 S3 版本控制的儲存貯體

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

2. 在導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。

3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板名稱。

4. 在 Trends and distributions (趨勢和分佈) 區段中，選擇 Versioning-enabled bucket count (已啟用版本控制的儲存貯體計數) 作為主要指標，以及選擇 Buckets (儲存貯體) 作為次要指標。

   Trend for date (日期的趨勢) 圖表會更新，以顯示已啟用 S3 版本控制的儲存貯體的趨勢。在趨勢線下方，您可以看到 Storage class distribution (儲存體類別分佈) 和 Region distribution (區域分佈) 子區段。

5. 若要檢視您在 Trend for date (<日期> 的趨勢) 圖表中看到的任何儲存貯體的更精細洞見，以便您可以執行更深入的分析，請執行下列動作：

a. 在圖表上選擇一點。將出現一個方塊，其中包含更精細洞見的選項。

b. 選擇要套用至資料以進行更深入分析的維度：Account (帳戶)、AWS 區域、Storage class (儲存體類別) 或 Bucket (儲存貯體)。接著選擇 Apply (套用)。

6. 在 Bubble analysis by buckets for date (<日期> 依儲存貯體的氣泡分析) 區段中，選擇 Versioning-enabled bucket count (已啟用版本控制的儲存貯體計數)、Buckets (儲存貯體) 和 Active buckets (作用中儲存貯體) 指標。

   Bubble analysis by buckets for date (日期的依儲存貯體進行氣泡分析) 區段會更新，以顯示您所選指標的資料。您可以使用此資料，來查看哪些儲存貯體已在儲存貯體總計數的內容中啟用 S3 版本控制。在 Bubble analysis by buckets for date (日期的依儲存貯體進行氣泡分析) 區段中，您可以使用任意三個指標在多個維度上繪製儲存貯體，以表示氣泡的 X-axis (X 軸)、Y-axis (Y 軸) 以及 Size (大小)。

步驟 2：啟用 S3 版本控制

在識別了已啟用 S3 版本控制的儲存貯體之後，您可以識別從未啟用 S3 版本控制或已暫停版本控制的儲存貯體。然後，您可以選擇性地在 S3 主控台中針對這些儲存貯體啟用版本控制。如需更多詳細資訊，請參閱 在儲存貯體上啟用版本控制。

識別使用 AWS 第 2 版簽署程序 (SigV2) 的請求

您可以使用 All unsupported signature requests (所有不支援的簽章請求) 指標，來識別使用 AWS 第 2 版簽署程序 (SigV2) 的請求。此資料可協助您識別正在使用 SigV2 的特定應用程式。然後，您可以將這些應用程序遷移至 AWS 第 4 版簽署程序 (SigV4)。

SigV4 是所有新 S3 應用程式的建議簽署方法。SigV4 提供了改善的安全性，並在所有 AWS 區域 中受到支援。如需詳細資訊，請參閱 Amazon S3 更新 - SigV2 棄用期間延長和修改。

先決條件

若要在 S3 Storage Lens 儀表板中查看 All unsupported signature requests (所有不支援的簽章請求)，您必須啟用 S3 Storage Lens Advanced metrics and recommendations (進階指標和建議)，然後選取 Advanced data protection metrics (進階資料保護指標)。如需更多詳細資訊，請參閱 建立和更新 Amazon S3 Storage Lens 儀表板。

步驟 1：依 AWS 帳戶、區域和儲存貯體檢查 SigV2 簽署趨勢

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板名稱。
4. 若要識別具有使用 SigV2 之請求的特定儲存貯體、帳戶和區域：

   a. 在 Top N overview for date (<日期> 的前 N 個概觀) 下的 Top N (前 N 個) 中，輸入您要查看其資料的儲存貯體數目。

   b. 對於 Metric (指標)，請從 Data protection (資料保護) 類別中選擇 All unsupported signature requests (所有不支援的簽章請求)。

      Top N overview for date (日期的前 N 個概觀) 會更新，依帳戶、AWS 區域和儲存貯體顯示 SigV2 請求的資料。Top N overview for date (日期的前 N 個概觀) 區段也會顯示前一天或前一週的變更百分比以及火花線，來視覺化趨勢。此趨勢是 14 天趨勢 (若為免費指標)，以及 30 天趨勢 (若為進階指標和建議)。

      > **Note**
      >
      > 有了 S3 Storage Lens 進階指標和建議，指標可供查詢使用 15 個月。如需更多詳細資訊，請參閱 指標選擇。

步驟 2：識別應用程式透過 SigV2 請求存取的儲存貯體

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板名稱。
4. 在 Storage Lens 儀表板中，選擇 Bucket (儲存貯體) 索引標籤。
5. 向下捲動至 Buckets (儲存貯體) 區段。在 Metrics categories (指標類別) 下，選擇 Data protection (資料保護)。然後清除 Summary (摘要)。

   Buckets (儲存貯體) 清單會更新，以顯示所顯示儲存貯體的所有可用 Data protection (資料保護) 指標。
6. 若要篩選 Buckets (儲存貯體) 清單，僅顯示特定資料保護指標，請選擇偏好設定圖示 ( )。
7. 清除所有資料保護指標的切換，直到只留下以下選取的指標為止：

- All unsupported signature requests (所有不支援的簽章請求)
- % all unsupported signature requests (% 所有不支援的簽章請求)

8. (選用) 在 Page size (頁面大小) 下，選擇要在清單中顯示的儲存貯體數目。
9. 選擇 Confirm (確認)。

   Buckets (儲存貯體) 清單會更新，以顯示 SigV2 請求的儲存貯體層級指標。您可以使用此資料來識別具有 SigV2 請求的特定儲存貯體。然後，您可以使用此資訊，將應用程式遷移到 SigV4。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的身分[身分驗證請求 (AWS Signature 第 4 版)](#)。

計算每個儲存貯體的複寫規則總數

S3 複寫可讓物件跨 Amazon S3 儲存貯體進行自動非同步複製。設定用於物件複寫的儲存貯體可由相同 AWS 帳戶 或不同帳戶擁有。如需更多詳細資訊，請參閱 [複製物件概觀](#)。

您可以使用 S3 Storage Lens 複寫規則計數指標，取得針對複寫所設定之儲存貯體的每個儲存貯體詳細資訊。此資訊包括儲存貯體和區域內部以及它們之間的複寫規則。

先決條件

若要在 S3 Storage Lens 儀表板中查看複寫規則計數指標，您必須啟用 S3 Storage Lens Advanced metrics and recommendations (進階指標和建議)，然後選取 Advanced data protection metrics (進階資料保護指標)。如需更多詳細資訊，請參閱 [建立和更新 Amazon S3 Storage Lens 儀表板](#)。

步驟 1：計算每個儲存貯體的複寫規則總數

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板名稱。
4. 在 Storage Lens 儀表板中，選擇 Bucket (儲存貯體) 索引標籤。
5. 向下捲動至 Buckets (儲存貯體) 區段。在 Metrics categories (指標類別) 下，選擇 Data protection (資料保護)。然後清除 Summary (摘要)。
6. 若要篩選 Buckets (儲存貯體) 清單以僅顯示複寫規則計數指標，請選擇偏好設定圖示 ()。
7. 清除所有資料保護指標的切換，直到只留下選取的複寫規則計數指標為止：

- Same-Region Replication rule count (相同區域複寫規則計數)
- Cross-Region Replication rule count (跨區域複寫規則計數)
- Same-account replication rule count (相同帳戶複寫規則計數)
- Cross-account replication rule count (跨帳戶複寫規則計數)
- Total replication rule count (複寫規則數總計)

8. (選用) 在 Page size (頁面大小) 下，選擇要在清單中顯示的儲存貯體數目。
9. 選擇 Confirm (確認)。

步驟 2：新增複寫規則

在具有每個儲存貯體複寫規則計數之後，您可以選擇性地建立其他複寫規則。如需更多詳細資訊，請參閱 設定即時複製的範例。

識別物件鎖定位元組的百分比

搭配 S3 物件鎖定，您可以使用單寫多讀 (WORM) 模型來存放物件。您可以使用物件鎖定，協助您讓物件在固定時間或無限期免於遭到刪除或覆寫。只有在您建立儲存貯體並同時啟用 S3 版本控制時，才能啟用物件鎖定。不過，您可以編輯個別物件版本的保留期間，或針對已啟用物件鎖定的儲存貯體套用法務保存措施。如需更多詳細資訊，請參閱 使用 S3 物件鎖定。

您可以在 S3 Storage Lens 中使用物件鎖定指標，查看您帳戶或組織的 % Object Lock bytes (% 物件鎖定位元組) 指標。您可以使用此資訊來識別帳戶或組織中未遵循資料保護最佳實務的儲存貯體。

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板名稱。
4. 在 Snapshot (快照) 區段的 Metrics categories (指標類別) 下，選擇 Data protection (資料保護)。

   Snapshot (快照) 區段會更新，以顯示資料保護指標，包括 % Object Lock bytes (% 物件鎖定位元組) 指標。您可以查看帳戶或組織的物件鎖定位元組的整體百分比。
5. 若要查看每個儲存貯體的 % Object Lock bytes (% 物件鎖定位元組)，請向下捲動至 Top N overview (前 N 個概觀) 區段。

   若要取得物件鎖定的物件層級資料，您也可以使用 Object Lock object count (物件鎖定物件計數) 和 % Object Lock objects (% 物件鎖定物件) 指標。

6. 針對 Metric (指標)，從 Data protection (資料保護) 類別中選擇 % Object Lock bytes (% 物件鎖定位元組)。

   依預設，Top N overview for date (<日期> 的前 N 個概觀) 區段會顯示前 3 個儲存貯體的指標。在 Top N (前 N 個) 欄位中，您可以增加儲存貯體的數目。Top N overview for date (日期的前 N 個概觀) 區段也會顯示前一天或前一週的變更百分比以及火花線，來視覺化趨勢。此趨勢是 14 天趨勢 (若為免費指標)，以及 30 天趨勢 (若為進階指標和建議)。

   > **Note**
   >
   > 有了 S3 Storage Lens 進階指標和建議，指標可供查詢使用 15 個月。如需更多詳細資訊，請參閱 指標選擇。

7. 檢閱 % Object Lock bytes (% 物件鎖定位元組) 的下列資料：

   - Top number accounts (前 <數目> 個帳戶) - 查看哪些帳戶具有最高和最低的 % Object Lock bytes (% 物件鎖定位元組)。
   - Top number Regions (前幾個區域) - 依區域檢視 % Object Lock bytes (% 物件鎖定位元組) 的明細。
   - Top number buckets (前幾個儲存貯體) - 查看哪些儲存貯體具有最高和最低的 % Object Lock bytes (% 物件鎖定位元組)。

使用 S3 Storage Lens 稽核物件擁有權設定

Amazon S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來停用存取控制清單 (ACL)，以及控制儲存貯體中物件的擁有權。如果您將物件擁有權設為儲存貯體擁有者強制執行，則可以停用存取控制清單 (ACL)，並取得儲存貯體中每個物件的擁有權。此方法可簡化存放在 Amazon S3 中之資料的存取管理。

預設情況下，當另一個 AWS 帳戶 會將物件上傳到您的 S3 儲存貯體時，該帳戶 (物件寫入者) 擁有物件、擁有其存取權，並且可以透過 ACL 授權其他使用者存取該物件。您可以使用「物件所有權」變更此預設行為。

Amazon S3 中的大多數新式使用案例不再需要使用 ACL。因此，建議您停用 ACL，除非在異常情況下必須個別控制每個物件的存取。透過將物件擁有權設為儲存貯體擁有者強制執行，您可以停用 ACL，並依賴政策進行存取控制。如需詳細資訊，請參閱「控制物件的擁有權並停用儲存貯體的 ACL」。

使用 S3 Storage Lens 存取管理指標，您可以識別未停用 ACL 的儲存貯體。在識別這些儲存貯體之後，您可以將 ACL 許可遷移至政策，並停用這些儲存貯體的 ACL。

主題

步驟 1：識別物件擁有權設定的一般趨勢

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板名稱。
4. 在 Snapshot for date (日期的快照) 區段中，於 Metrics categories (指標類別) 下，選擇 Access management (存取管理)。

   Snapshot for date (日期的快照) 區段即會更新，以顯示 % Object Ownership bucket owner enforced (% 物件擁有權儲存貯體擁有者強制執行) 指標。您可以查看帳戶或組織中針對物件擁有權使用儲存貯體擁有者強制執行設定，以停用 ACL 之儲存貯體的整體百分比。

步驟 2：識別物件擁有權設定的儲存貯體層級趨勢

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板名稱。
4. 若要檢視更詳細的儲存貯體層級指標，請選擇 Bucket (儲存貯體) 索引標籤。
5. 在 Distribution by buckets for date (依日期的儲存貯體進行分佈) 區段中，選擇 % Object Ownership bucket owner enforced (% 物件擁有權儲存貯體擁有者強制執行) 指標。

   此圖表會更新以顯示 % Object Ownership bucket owner enforced (% 物件擁有權儲存貯體擁有者強制執行) 的每個儲存貯體明細。您可以查看哪些儲存貯體使用儲存貯體擁有者強制執行設定，以停用 ACL。
6. 若要在內容中檢視儲存貯體擁有者強制執行設定，請向下捲動至 Buckets (儲存貯體) 區段。針對 Metrics categories (指標類別)，選取 Access management (存取管理)。然後清除 Summary (摘要)。

Buckets (儲存貯體) 清單會顯示所有三個物件擁有權設定的資料：儲存貯體擁有者強制執行、儲存貯體擁有者偏好，以及物件寫入器。

7. 若要篩選 Buckets (儲存貯體) 清單，僅顯示特定物件擁有權設定的指標，請選擇偏好設定圖示 ()。
8. 清除您不想要查看的指標。
9. (選用) 在 Page size (頁面大小) 下，選擇要在清單中顯示的儲存貯體數目。
10. 選擇 Confirm (確認)。

步驟 3：將您的物件擁有權設定更新為儲存貯體擁有者強制執行，以停用 ACL

在識別了針對物件擁有權使用物件寫入器和儲存貯體有者偏好設定的儲存貯體之後，您可以將 ACL 許可遷移至儲存貯體政策。在完成了遷移 ACL 許可之後，您可以接著將物件擁有權設定更新為儲存貯體擁有者強制執行，以停用 ACL。如需詳細資訊，請參閱「停用 ACL 的先決條件」。

使用 S3 Storage Lens 指標來改善效能

如果已啟用 S3 Storage Lens 進階指標，您可以使用詳細的狀態碼指標，來取得成功或失敗請求的計數。您可以使用此資訊，針對存取或效能問題進行疑難排解。詳細的狀態碼指標會顯示 HTTP 狀態碼的計數，例如 403 禁止和 503 服務無法使用。您可以跨 S3 儲存貯體、帳戶和組織檢查詳細狀態碼指標的整體趨勢。然後，您可以向下切入至儲存貯體層級指標，以識別目前正在存取這些儲存貯體並導致錯誤的工作負載。

例如，您可以查看 403 Forbidden error count (403 禁止錯誤計數) 指標，以識別在未套用正確許可的情況下存取工作負載的工作負載。在識別了這些工作負載之後，您可以在 S3 Storage Lens 之外進行深入探索，針對 403 禁止錯誤進行疑難排解。

此範例說明如何使用 403 Forbidden error count (403 禁止錯誤計數) 和 % 403 Forbidden errors (% 403 禁止錯誤計數) 指標，針對 403 禁止錯誤執行趨勢分析。您可以使用這些指標，來識別在未套用正確許可的情況下存取工作負載的工作負載。您可以針對任何其他 Detailed status code metrics (詳細狀態碼指標) 執行類似的趨勢分析。如需詳細資訊，請參閱 Amazon S3 Storage Lens 指標詞彙表。

先決條件

若要在 S3 Storage Lens 儀表板中查看 Detailed status code metrics (詳細狀態碼指標) 指標，您必須啟用 S3 Storage Lens Advanced metrics and recommendations (進階指標和建議)，然後選取 Detailed status code metrics (詳細狀態碼指標)。如需詳細資訊，請參閱 建立和更新 Amazon S3 Storage Lens 儀表板。

主題

步驟 1：針對個別 HTTP 狀態碼執行趨勢分析

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板名稱。
4. 在 Trends and distributions (趨勢和分佈) 區段中，針對 Primary metric (主要指標)，從 Detailed status codes (詳細狀態碼) 類別中選擇 403 Forbidden error count (403 禁止錯誤計數)。針對 Secondary metric (次要指標)，選擇 % 403 Forbidden errors (% 403 禁止錯誤)。
5. 向下捲動至 Top N overview for date (日期的前 N 個概觀) 區段。針對 Metrics (指標)，從 Detailed status codes (詳細狀態碼) 類別中選擇 403 Forbidden error count (403 禁止錯誤計數) 或 % 403 Forbidden errors (% 403 禁止錯誤)。

   Top N overview for date (日期的前 N 個概觀) 區段會更新，依帳戶、AWS 區域和儲存貯體顯示熱門 403 禁止錯誤計數。

步驟 2：依儲存貯體分析錯誤計數

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要檢視的儀表板名稱。
4. 在 Storage Lens 儀表板中，選擇 Bucket (儲存貯體) 索引標籤。
5. 向下捲動至 Buckets (儲存貯體) 區段。針對 Metrics categories (指標類別)，選取 Detailed status code (詳細狀態碼) 指標。然後清除 Summary (摘要)。

   Buckets (儲存貯體) 清單會更新，以顯示所有可用的詳細狀態碼指標。您可以使用此資訊，來查看哪些儲存貯體具有很大比例的特定 HTTP 狀態碼，以及哪些狀態碼通用於儲存貯體之間。

6. 若要篩選 Buckets (儲存貯體) 清單，僅顯示特定詳細狀態碼指標，請選擇偏好設定圖示 ()。
7. 清除您不想要在 Buckets (儲存貯體) 清單中檢視之任何詳細狀態碼指標的切換。
8. (選用) 在 Page size (頁面大小) 下，選擇要在清單中顯示的儲存貯體數目。
9. 選擇 Confirm (確認)。

   Buckets (儲存貯體) 清單會顯示您指定之儲存貯體數目的錯誤計數指標。您可以使用此資訊，來識別遇到許多錯誤的特定儲存貯體，並依儲存貯體針對錯誤進行疑難排解。

步驟 3：針對錯誤進行疑難排解

在識別具有高比例特定 HTTP 狀態碼的儲存貯體之後，您可以針對這些錯誤進行疑難排解。如需詳細資訊，請參閱下列內容：

- 當我嘗試在 Amazon S3 中上傳檔案時，為什麼收到 403 禁止錯誤？
- 當我嘗試在 Amazon S3 中修改儲存貯體政策時，為什麼收到 403 禁止錯誤？
- 如何針對 Amazon S3 儲存貯體中的 403 禁止錯誤進行疑難排解，其中所有資源都來自相同的 AWS 帳戶？
- 如何針對 Amazon S3 中的 HTTP 500 或 503 錯誤進行疑難排解？

Amazon S3 Storage Lens 指標詞彙表

Amazon S3 Storage Lens 指標詞彙表提供 S3 Storage Lens 的完整免費和進階指標清單。

S3 Storage Lens 提供所有儀表板和組態的免費指標，其中包括升級至進階指標和建議的選項。

- 免費指標包含與儲存用量相關的指標，例如帳戶中儲存貯體和物件的數目。免費指標還包括使用案例型指標，例如，成本最佳化和資料保護指標。系統會每天收集所有免費指標，且資料可供查詢多達 14 天。
- 進階指標和建議包括免費指標中的所有指標，以及其他指標，例如進階資料保護和成本最佳化指標。進階指標還包括其他指標類別，例如活動指標和詳細狀態碼指標。進階指標資料可供查詢使用 15 個月。

  若您使用具有進階指標和建議的 S3 Storage Lens 需支付額外費用。如需詳細資訊，請參閱 Amazon S3 定價。如需進階指標和建議功能的詳細資訊，請參閱 指標選擇。

> **Note**
>
> 對於 Storage Lens 群組，僅提供免費方案儲存指標。進階方案指標無法在 Storage Lens 群組層級使用。

指標名稱

下表中的 Metric name (指標名稱) 欄提供 S3 主控台中每個 S3 Storage Lens 的名稱。CloudWatch and export (CloudWatch 和匯出) 欄提供 Amazon CloudWatch 中每個指標的名稱，以及您可以在 S3 Storage Lens 儀表板中設定的指標匯出檔案。

衍生指標公式

指標匯出和 CloudWatch 發佈選項無法使用衍生指標。不過，您可以使用 Derived metrics formula (衍生指標公式) 欄中顯示的指標公式進行運算。

解釋 Amazon S3 Storage Lens 指標單位倍數的前綴符號 (K、M、G 等)

S3Storage Lens 指標單位倍數是使用前綴符號寫入的。這些前綴符號符合採用國際計量局 (BIPM) 標準的國際單位系統 (SI) 符號。這些符號也會在統一計量單位代碼 (UCUM) 中使用。如需詳細資訊，請參閱 SI 前綴符號清單。

> **Note**
>
> - S3 儲存體位元組的測量單位是二進位 GB，其中 1 GB 是 2^{30} 個位元組，1 TB 是 2^{40} 個位元組，而 1 PB 是 2^{50} 個位元組。根據國際電工委員會 (IEC) 的定義，此測量單位也稱為 GiB。
> - 依據物件的生命週期組態，當物件的生命週期接近結尾時，Amazon S3 會將此物件排入佇列等待移除，並會以非同步方式進行移除物件。因此，過期日期與 Amazon S3 移除物件的日期之間，可能會有所延遲。S3 Storage Lens 不包含已過期但尚未移除之物件的指標。如需 S3 生命週期中過期動作的詳細資訊，請參閱 即將到期的物件。

S3 Storage Lens 指標詞彙表

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
儲存體總量	StorageBytes	儲存體總量，包括不完整的分段上傳、物件中繼資料和刪除標記	免費	總結	N	-
物件計數	ObjectCount	物件總計數	免費	總結	N	-
平均物件大小	-	平均物件大小	免費	總結	Y	sum(StorageBytes)/sum(ObjectCount)
作用中儲存貯體	-	作用中儲存用量 > 0 位元組的儲存貯體總數	免費	總結	Y	-
儲存貯體	-	儲存貯體總數	免費	總結	Y	-
帳戶	-	儲存體在範圍內的帳戶數量	免費	總結	Y	-
目前版本位元組	CurrentVersionStorageBytes	物件目前版本的位元組數目	免費	成本最佳化	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 目前版本位元組	-	範圍中物件目前版本的位元組百分比	免費	成本最佳化	Y	sum(CurrentVersionStorageBytes)/sum(StorageBytes)
目前版本物件計數	CurrentVersionObjectCount	最新版本物件的計數	免費	成本最佳化	N	-
% 目前版本物件	-	目前版本範圍內的物件百分比	免費	成本最佳化	Y	sum(CurrentVersionObjectCount)/sum(ObjectCount)
非目前版本位元組	NonCurrentVersionStorageBytes	非目前版本位元組的數目	免費	成本最佳化	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 非目前版本位元組	-	範圍中非目前版本的位元組百分比	免費	成本最佳化	Y	sum(NonCurrentVersionStorageBytes)/sum(StorageBytes)
非目前版本物件計數	NonCurrentVersionObjectCount	非目前物件版本的計數	免費	成本最佳化	N	-
% 非目前版本物件	-	非目前版本範圍內的物件百分比	免費	成本最佳化	Y	sum(NonCurrentVersionObjectCount)/sum(ObjectCount)
刪除標記位元組	DeleteMarkerStorageBytes	範圍中刪除標記的位元組數目	免費	成本最佳化	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 刪除標記位元組	-	範圍中刪除標記的位元組百分比	免費	成本最佳化	Y	sum(DeleteMarkerStorageBytes)/ sum(StorageBytes)
刪除標記物件計數	DeleteMarkerObjectCount	具有刪除標記的物件總數	免費	成本最佳化	N	-
% 刪除標記物件	-	具有刪除標記之範圍中的物件百分比	免費	成本最佳化	Y	sum(DeleteMarkerObjectCount)/ sum(ObjectCount)
未完成分段上傳位元組	IncompleteMultipartUploadStorageBytes	範圍中未完成分段上傳的位元組總數	免費	成本最佳化	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 未完成分段上傳位元組	-	範圍中未完成分段上傳結果的位元組百分比	免費	成本最佳化	Y	sum(IncompleteMultipartUploadStorageBytes)/ sum(StorageBytes)
未完成分段上傳物件計數	IncompleteMultipartUploadObjectCount	不完整分段上傳範圍內的物件數	免費	成本最佳化	N	-
% 未完成分段上傳物件	-	不完整分段上傳範圍內的物件百分比	免費	成本最佳化	Y	sum(IncompleteMultipartUploadObjectCount)/ sum(ObjectCount)
超過 7 天的未完成分段上傳儲字體位元組	IncompleteMPUStorageBytesOlderThan7Days	範圍中超過 7 天的未完成分段上傳的位元組總數	免費	成本最佳化	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
超過 7 天的 % 未完成分段上傳儲字體位元組	-	超過 7 天的未完成分段上傳的位元組百分比	免費	成本最佳化	Y	sum(IncompleteMPUStorageBytesOlderThan7Days)/ sum(StorageBytes)
超過 7 天的未完成分段上傳物件計數	IncompleteMPUObjectCountOlderThan7Days	超過 7 天的未完成分段上傳的物件數目	免費	成本最佳化	N	-
超過 7 天的 % 未完成分段上傳物件計數	-	超過 7 天的未完成分段上傳的物件百分比	免費	成本最佳化	Y	sum(IncompleteMPUObjectCountOlderThan7Days)/ sum(ObjectCount)
轉移生命週期規則計數	TransitionLifecycleRuleCount	將物件轉移至另一個儲存體類別的生命週期規則計數	Adva (進階)	成本最佳化	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
每個儲存貯體的平均轉移生命週期規則	-	將物件轉移至另一個儲存體類別的生命週期規則平均數	Adva (進階)	成本最佳化	Y	sum(TransitionLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
過期生命週期規則計數	ExpirationLifecycleRuleCount	使物件過期的生命週期規則計數	Adva (進階)	成本最佳化	N	-
每個儲存貯體的平均過期生命週期規則	-	使物件過期的生命週期規則平均數	Adva (進階)	成本最佳化	Y	sum(ExpirationLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
非目前版本轉移生命週期規則計數	NoncurrentVersionTransitionLifecycleRuleCount	將非目前物件版本轉移至另一個儲存體類別的生命週期規則計數	Adva (進階)	成本最佳化	N	

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
每個儲存貯體的平均非目前版本轉移生命週期規則	-	將非目前物件版本轉移至另一個儲存體類別的生命週期規則平均數	Adva (進階)	成本最佳化	Y	sum(NoncurrentVersionTransitionLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
非目前版本過期生命週期規則計數	NoncurrentVersionExpirationLifecycleRuleCount	使非目前物件版本過期的生命週期規則計數	Adva (進階)	成本最佳化	N	-
每個儲存貯體的平均非目前版本過期生命週期規則	-	使非目前物件版本過期的生命週期規則平均數	Adva (進階)	成本最佳化	Y	sum(NoncurrentVersionExpirationLifecycleRuleCount)/sum(DistinctNumberOfBuckets)

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
中止未完成分段上傳生命週期規則計數	AbortIncompleteMPULifecycleRuleCount	刪除未完成分段上傳的生命週期規則計數	Adva (進階)	成本最佳化	N	-
每個儲存貯體的平均中止未完成分段上傳生命週期規則	-	刪除未完成分段上傳的生命週期規則平均數	Adva (進階)	成本最佳化	Y	sum(AbortIncompleteMPULifecycleRuleCount)/sum(DistinctNumberOfBuckets)
過期物件刪除標記生命週期規則計數	ExpiredObjectDeleteMarkerLifecycleRuleCount	移除過期物件刪除標記的生命週期規則計數	Adva (進階)	成本最佳化	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
每個儲存貯體的平均過期物件刪除標記生命週期規則	-	移除過期物件刪除標記的生命週期規則平均數	Adva (進階)	成本最佳化	Y	sum(ExpiredObjectDeleteMarkerLifecycleRuleCount)/ sum(DistinctNumberOfBuckets)
生命週期規則總計數	TotalLifecycleRuleCount	生命週期規則的總計數	Adva (進階)	成本最佳化	N	-
每個儲存貯體的平均生命週期規則計數	-	生命週期規則平均數	Adva (進階)	成本最佳化	Y	sum(TotalLifecycleRuleCount)/ sum(DistinctNumberOfBuckets)
加密位元組	EncryptedStorageBytes	加密位元組總數	免費	資料保護	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 加密位元組	-	已加密的位元組總數百分比	免費	資料保護	Y	sum(EncryptedObjectCount)/sum(StorageBytes)
加密物件計數	EncryptedObjectCount	未加密的物件總計數	免費	資料保護	N	-
% 加密物件	-	已加密的物件百分比	免費	資料保護	Y	sum(EncryptedStorageBytes)/sum(ObjectCount)
未加密位元組	UnencryptedStorageBytes	未加密的位元組數目	免費	資料保護	Y	sum(StorageBytes) - sum(EncryptedStorageBytes)

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 未加密位元組	-	未加密的位元組百分比	免費	資料保護	Y	sum(UnencryptedStorageBytes)/sum(StorageBytes)
未加密物件計數	UnencryptedObjectCount	未加密的物件總計數	免費	資料保護	Y	sum(ObjectCount) - sum(EncryptedObjectCount)
% 未加密物件	-	未加密物件的百分比	免費	資料保護	Y	sum(UnencryptedStorageBytes)/sum(ObjectCount)
複寫的儲存體位元組來源	ReplicatedStorageBytesSource	從來源儲存貯體複寫的位元組總數	免費	資料保護	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 複寫的位元組來源	-	從來源儲存貯體複寫的位元組總數百分比	免費	資料保護	Y	sum(ReplicatedStorageBytesSource)/sum(StorageBytes)
複寫的物件計數來源	ReplicatedObjectCountSource	從來源儲存貯體複寫的物件計數	免費	資料保護	N	-
% 複寫的物件來源	-	從來源儲存貯體複寫的物件總數百分比	免費	資料保護	Y	sum(ReplicatedStorageObjectCount)/sum(ObjectCount)
複寫儲存體位元組目的地	ReplicatedStorageBytes	複寫至目的地儲存貯體的位元組總數	免費	資料保護	Y	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 複寫的位元組目的地	-	複寫至目的地儲存貯體的位元組總數百分比	免費	資料保護	Y	sum(ReplicatedStorageBytes) / sum(StorageBytes)
複寫的物件計數目的地	ReplicatedObjectCount	複寫到目的地儲存貯體的物件計數	免費	資料保護	Y	-
% 複寫的物件目的地	-	複寫至目的地儲存貯體的物件總數百分比	免費	資料保護	Y	sum(ReplicatedObjectCount)/ sum(ObjectCount)

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
物件鎖定位元組	ObjectLockEnabledStorageBytes	已啟用物件鎖定的儲存體位元組總計數	免費	資料保護	Y	sum(UnencryptedStorageBytes)/ sum(ObjectLockEnabledStorageCount) - sum(ObjectLockEnabledStorageBytes)
% 物件鎖定位元組	-	已啟用物件鎖定的儲存體位元組百分比	免費	資料保護	Y	sum(ObjectLockEnabledStorageBytes)/ sum(StorageBytes)
物件鎖定物件計數	ObjectLockEnabledObjectCount	物件鎖定物件的總計數	免費	資料保護	Y	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 物件鎖定物件	-	已啟用物件鎖定的物件總數百分比	免費	資料保護	Y	sum(ObjectLockEnabledObjectCount)/sum(ObjectCount)
已啟用版本控制的儲存貯體計數	VersioningEnabledBucketCount	已啟用 S3 版本控制的儲存貯體計數	免費	資料保護	N	-
% 已啟用版本控制的儲存貯體	-	已啟用 S3 版本控制的儲存貯體百分比	免費	資料保護	Y	sum(VersioningEnabledBucketCount)/sum(DistinctNumberOfBuckets)
已啟用 MFA 刪除的儲存貯體計數	MFADeleteEnabledBucketCount	已啟用 MFA (多重因素認證) 刪除的儲存貯體計數	免費	資料保護	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 已啟用 MFA 刪除的儲存貯體	-	已啟用 MFA (多重因素認證) 刪除的儲存貯體百分比	免費	資料保護	Y	sum(MFADeleteEnabledBucketCount)/ sum (Distinct NumberOfBuckets)
已啟用 SSE-KMS 的儲存貯體計數	SSEKMSEnabledBucketCount	使用伺服器端加密搭配 AWS Key Management Service 金鑰 (SSE-KMS) 進行預設儲存貯體加密的儲存貯體計數	免費	資料保護	N	-
% 已啟用 SSE-KMS 的儲存貯體	-	使用 SSE-KMS 進行預設儲存貯體加密的儲存貯體百分比	免費	資料保護	Y	sum(SSEKMSEnabledBucketCount)/ sum(DistinctNumberOfBuckets)
所有不支援的簽章請求	AllUnsupportedSignatureRequests	使用不支援的 AWS 簽章版本的請求總數	Adva (進階)	資料保護	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 所有不支援的簽章請求	-	使用不支援的 AWS 簽章版本的請求百分比	Adva (進階)	資料保護	Y	sum(AllUnsupportedSignatureRequests) / sum(AllRequests)
所有不支援的 TLS 請求	AllUnsupportedTLSRequests	使用不支援的 Transport Layer Security (TLS) 版本的請求數目	Adva (進階)	資料保護	N	-
% 所有不支援的 TLS 請求	-	使用不支援的 TLS 版本的請求百分比	Adva (進階)	資料保護	Y	sum(AllUnsupportedTLSRequests)/ sum(AllRequests)
所有 SSE-KMS 請求	AllSSEKMSRequests	指定 SSE-KMS 的請求總數	Adva (進階)	資料保護	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 所有 SSE-KMS 請求	-	指定 SSE-KMS 的請求百分比	Adva (進階)	資料保護	Y	sum(AllSSEKMSRequests)/sum(AllRequests)
相同區域複寫規則計數	SameRegionReplicationRuleCount	相同區域複寫 (SRR) 的複寫規則計數	Adva (進階)	資料保護	N	-
每個儲存貯體的平均相同區域複寫規則	-	SRR 的複寫規則平均數	Adva (進階)	資料保護	Y	sum(SameRegionReplicationRuleCount)/sum(DistinctNumberOfBuckets)
跨區域複寫規則計數	CrossRegionReplicationRuleCount	跨區域複寫 (CRR) 的複寫規則計數	Adva (進階)	資料保護	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
每個儲存貯體的平均跨區域複寫規則	-	CRR 的複寫規則平均數	Adva (進階)	資料保護	Y	sum(CrossRegionReplicationRuleCount) / sum(DistinctNumberOfBuckets)
相同帳戶複寫規則計數	SameAccountReplicationRuleCount	相同帳戶內複寫的複寫規則計數	Adva (進階)	資料保護	N	-
每個儲存貯體的平均相同帳戶複寫規則	-	相同帳戶內複寫的複寫規則平均數	Adva (進階)	資料保護	Y	sum(SameAccountReplicationRuleCount) / sum(DistinctNumberOfBuckets)
跨帳戶複寫規則計數	CrossAccountReplicationRuleCount	跨帳戶複寫的複寫規則計數	Adva (進階)	資料保護	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
每個儲存貯體的平均跨帳戶複寫規則	-	跨帳戶複寫的複寫規則平均數	Adva (進階)	資料保護	Y	sum(CrossAccountReplicationRuleCount)/ sum(DistinctNumberOfBuckets)
無效的目的地複寫規則計數	InvalidDestinationReplicationRuleCount	複寫目的地無效的複寫規則計數	Adva (進階)	資料保護	N	-
每個儲存貯體的平均無效目的地複寫規則	-	複寫目的地無效的複寫規則平均數	Adva (進階)	資料保護	Y	sum(InvalidReplicationRuleCount)/ sum (DistinctNumberOfBuckets)
複寫規則數總計	-	複寫規則數總計	Adva (進階)	資料保護	Y	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
每個儲存貯體的平均複寫規則計數	-	平均複寫規則數總計	Adva (進階)	資料保護	Y	sum(all replication rule count metrics)/ sum(DistinctNumberOfBuckets)
物件擁有權儲存貯體擁有者強制執行的儲存貯體計數	ObjectOwnershipBucketOwnerEnforcedBucketCount	針對物件擁有權使用儲存貯體擁有者強制執行設定，來停用存取控制清單 (ACL) 的儲存貯體總計數	免費	存取管理	N	-
% 物件擁有權儲存貯體擁有者強制執行的儲存貯體	-	針對物件擁有權使用儲存貯體擁有者強制執行設定，來停用 ACL 的儲存貯體百分比	免費	存取管理	Y	sum(ObjectOwnershipBucketOwnerEnforcedBucketCount)/ sum (Distinct NumberOfBuckets)

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
物件擁有權儲存貯體擁有者偏好的儲存貯體計數	ObjectOwnershipBucketOwnerPreferredBucketCount	針對物件擁有權使用儲存貯體擁有者偏好設定的儲存貯體總計數	免費	存取管理	N	-
% 物件擁有權儲存貯體擁有者偏好的儲存貯體	-	針對物件擁有權使用儲存貯體擁有者偏好設定的儲存貯體百分比	免費	存取管理	Y	sum(ObjectOwnershipBucketOwnerPreferredBucketCount)/sum(DistinctNumberOfBuckets)
物件擁有權物件寫入器儲存貯體計數	ObjectOwnershipObjectWriterBucketCount	針對物件擁有權使用物件寫入器設定的儲存貯體總計數	免費	存取管理	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 物件擁有權物件寫入器儲存貯體	-	針對物件擁有權使用物件寫入器設定的儲存貯體百分比	免費	存取管理	Y	sum(ObjectOwnershipObjectWriterBucketCount)/sum(DistinctNumberOfBuckets)
已啟用 Transfer Acceleration 的儲存貯體計數	TransferAccelerationEnabledBucketCount	已啟用 Transfer Acceleration 的儲存貯體總計數	免費	效能	N	-
% 已啟用 Transfer Acceleration 的儲存貯體	-	已啟用 Transfer Acceleration 的儲存貯體百分比	免費	效能	Y	sum(TransferAccelerationEnabledBucketCount)/sum(DistinctNumberOfBuckets)
已啟用事件通知的儲存貯體計數	EventNotificationEnabledBucketCount	已啟用事件通知的儲存貯體總計數	免費	事件	N	

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
% 已啟用事件通知的儲存貯體	-	已啟用事件通知的儲存貯體百分比	免費	事件	Y	sum(EventNotificationEnabledBucketCount)/sum(DistinctNumberOfBuckets)
所有請求	AllRequests	提出的 請求總數	Adva(進階)	活動	N	-
GET 請求	GetRequests	提出的 GET 請求總數	Adva(進階)	活動	N	-
Put 請求	PutRequests	提出的 PUT 請求總數	Adva(進階)	活動	N	-
Head 請求	HeadRequests	提出的 HEAD 請求總數	Adva(進階)	活動	N	-
Delete 請求	DeleteRequests	提出的 DELETE 請求總數	Adva(進階)	活動	N	-

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
LIST 請求	ListRequests	提出的 LIST 請求總數	Adva (進階)	活動	N	-
Post 請求	PostRequests	提出的 POST 請求總數	Adva (進階)	活動	N	-
Select 請求	SelectRequests	S3 Select 請求的總數	Adva (進階)	活動	N	-
Select 掃描的位元組	SelectScannedBytes	已掃描的 S3 Select 位元組數	Adva (進階)	活動	N	-
Select 傳回的位元組	SelectReturnedBytes	已傳回的 S3 Select 位元組數	Adva (進階)	活動	N	-
下載的位元組數	BytesDownloaded	已下載的位元組數	Adva (進階)	活動	N	-
% 擷取率	-	已下載的位元組百分比	Adva (進階)	活動	Y	sum(BytesDownloaded)/sum(StorageBytes)

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
上傳的位元組數	BytesUploaded	已上傳的位元組數	Adva (進階)	活動	N	-
% 導入率	-	已上傳的位元組百分比	Adva (進階)	活動	Y	sum(BytesUploaded) / sum(StorageBytes)
4xx 錯誤	4xxErrors	HTTP 4xx 狀態碼的總數	Adva (進階)	活動	N	-
5xx 錯誤	5xxErrors	HTTP 5xx 狀態碼的總數	Adva (進階)	活動	N	-
錯誤總數	-	所有 4xx 和 5xx 錯誤的總和	Adva (進階)	活動	Y	sum(4xxErrors) + sum(5xxErrors)
% 錯誤率	-	4xx 和 5xx 錯誤總數占要求總數的百分比	Adva (進階)	活動	Y	sum(TotalErrors)/ sum(TotalRequests)

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
200 OK 狀態計數	200OKStatusCount	200 OK 狀態碼的總數	Adva (進階)	詳細狀態碼	N	-
% 200 OK 狀態	-	200 OK 狀態碼總數占請求總數的百分比	Adva (進階)	詳細狀態碼	Y	sum(200OKStatusCount)/sum(AllRequests)
206 部分內容狀態計數	206PartialContentStatusCount	206 部分內容狀態碼的總數	Adva (進階)	詳細狀態碼	N	-
% 206 部分內容狀態	-	206 部分內容狀態碼總數占請求總數的百分比	Adva (進階)	詳細狀態碼	Y	sum(206PartialContentStatusCount)/sum(AllRequests)

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
400 錯誤的請求錯誤計數	400BadRequestErrorCount	400 錯誤的請求狀態碼的總數	Adva (進階)	詳細狀態碼	N	-
% 400 錯誤的請求錯誤	-	400 錯誤的請求狀態碼總數占請求總數的百分比	Adva (進階)	詳細狀態碼	Y	sum(400BadRequestErrorCount)/ sum(All Requests)
403 禁止錯誤計數	403ForbiddenErrorCount	403 禁止狀態碼的總數	Adva (進階)	詳細狀態碼	N	-
% 403 禁止錯誤	-	403 禁止狀態碼總數占請求總數的百分比	Adva (進階)	詳細狀態碼	Y	sum(403ForbiddenErrorCount) / sum(AllRequests)

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
404 找不到錯誤計數	404NotFoundErrorCount	404 找不到狀態碼的總數	Adva (進階)	詳細狀態碼	N	-
% 404 找不到錯誤	-	404 找不到狀態碼總數占請求總數的百分比	Adva (進階)	詳細狀態碼	Y	sum(404NotFoundErrorCount)/ sum(AllRequests)
500 內部伺服器錯誤計數	500InternalServerErrorCount	500 內部伺服器錯誤狀態碼的總數	Adva (進階)	詳細狀態碼	N	-
% 500 內部伺服器錯誤	-	500 內部伺服器錯誤狀態碼總數占請求總數的百分比	Adva (進階)	詳細狀態碼	Y	sum(500InternalServerErrorCount)/ sum (AllRequests)

指標名稱	CloudWatch 和匯出	描述	Tier[1]	Cate	衍生	衍生指標公式
503 服務無法使用錯誤計數	503ServiceUnavailableErrorCount	503 服務無法使用狀態碼的總數	Adva (進階)	詳細狀態碼	N	-
% 503 服務無法使用錯誤	-	503 服務無法使用狀態碼總數占請求總數的百分比	Adva (進階)	詳細狀態碼	Y	sum(503ServiceUnavailableErrorCount) / sum(AllRequests)

[1] 所有免費方案儲存指標都可在 Storage Lens 群組層級使用。進階方案指標無法在 Storage Lens 群組層級使用。

[2] 規則計數指標和儲存貯體設定指標無法在字首層級使用。

透過主控台和 API 使用 Amazon S3 Storage Lens

Amazon S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。您可以使用 S3 Storage Lens 指標產生摘要洞見，例如了解您在整個組織中擁有多少儲存體，或是成長速度最快的儲存貯體和字首有哪些。您也可以使用 S3 Storage Lens 指標，識別成本最佳化機會、實作資料保護和安全最佳實務，以及改善應用程式工作負載的效能。例如，您可以識別沒有 S3 生命週期規則的儲存貯體，這些規則可使超過 7 天未完成的分段上傳過期。您也可以識別未遵循資料保護最佳實務的儲存貯體，例如使用 S3 複寫或 S3 版本控制。S3 Storage Lens 也會分析指標，以提供內容相關建議，您可以用來最佳化儲存成本，並套用最佳實務保護您的資料。

S3 Storage Lens 會彙總您的指標，並在 Amazon S3 主控台 Buckets (儲存貯體) 頁面上的 Account snapshot (帳戶快照) 區段中顯示資訊。S3 Storage Lens 也會提供互動式儀表板，您可以用來以視覺

化方式呈現洞見與趨勢、標記異常值，以及接收最佳化儲存成本和套用資料保護最佳實務的建議。您的儀表板具有向下切入選項可產生及視覺化組織、帳戶、AWS 區域、儲存類別、儲存貯體、字首或 Storage Lens 群組層級的洞察。您也可以將 CSV 或 Parquet 格式的每日指標匯出傳送到 S3 儲存貯體。

以下區段包含建立、更新和檢視 S3 Storage Lens 組態的範例，以及執行與此功能相關操作的範例。如果您透過 AWS Organizations 使用 S3 Storage Lens，這些範例中也包含相關使用案例。在範例中，以您專屬的變數值取代任何變數值。

主題

在主控台中使用 Amazon S3 Storage Lens

Amazon S3 Storage Lens 是一種雲端儲存體分析功能，您可以用來了解整個組織使用物件儲存體的情況及其活動情形。您可以使用 S3 Storage Lens 指標產生摘要洞見，例如了解您在整個組織中擁有多少儲存體，或是成長速度最快的儲存貯體和字首有哪些。您也可以使用 S3 Storage Lens 指標，識別成本最佳化機會、實作資料保護和安全最佳實務，以及改善應用程式工作負載的效能。例如，您可以識別沒有 S3 生命週期規則的儲存貯體，這些規則可使超過 7 天未完成的分段上傳過期。您也可以識別未遵循資料保護最佳實務的儲存貯體，例如使用 S3 複寫或 S3 版本控制。S3 Storage Lens 也會分析指標，以提供內容相關建議，您可以用來最佳化儲存成本，並套用最佳實務保護您的資料。

S3 Storage Lens 會彙總您的指標，並在 Amazon S3 主控台 Buckets (儲存貯體) 頁面上的 Account snapshot (帳戶快照) 區段中顯示資訊。S3 Storage Lens 也會提供互動式儀表板，您可以用來以視覺化方式呈現洞見與趨勢、標記異常值，以及接收最佳化儲存成本和套用資料保護最佳實務的建議。您的儀表板具有向下切入選項可產生及視覺化組織、帳戶、 AWS 區域、儲存類別、儲存貯體、字首或 Storage Lens 群組層級的洞察。您也可以將 CSV 或 Parquet 格式的每日指標匯出傳送到 S3 儲存貯體。

> **Note**
>
> 對儀表板組態的任何更新最多可能需要 48 小時才能正確顯示或視覺化。

主題

- 正在停用或刪除 Amazon S3 Storage Lens 儀表板
- 使用建立組織層級儀 AWS Organizations 表板

建立和更新 Amazon S3 Storage Lens 儀表板

S3 Storage Lens 會彙總您的指標，並在 Amazon S3 主控台 Buckets (儲存貯體) 頁面上的 Account snapshot (帳戶快照) 區段中顯示資訊。S3 Storage Lens 也會提供互動式儀表板，您可以用來以視覺化方式呈現洞見與趨勢、標記異常值，以及接收最佳化儲存成本和套用資料保護最佳實務的建議。您的儀表板具有向下切入選項可產生及視覺化組織、帳戶、 AWS 區域、儲存類別、儲存貯體、字首或 Storage Lens 群組層級的洞察。您也可以將 CSV 或 Parquet 格式的每日指標匯出傳送到 S3 儲存貯體。

Amazon S3 儲存鏡頭的預設儀表板為default-account-dashboard。此儀表板是由 Amazon S3 預先設定，可協助您在主控台上以視覺化方式呈現整個帳戶彙總的免費和進階指標的摘要洞見和趨勢。您無法修改預設儀表板的組態範圍，但您可以將指標選項從免費指標升級為付費進階指標和建議、設定選用指標匯出，甚至停用預設儀表板。無法刪除預設儀表板。

您也可以建立額外的 S3 Storage Lens 自訂儀表板，這些儀表板可以將範圍設定為您的組織， AWS Organizations 或設定在帳戶內的特定區域或儲存貯體。

建立 Amazon S3 Storage Lens 儀表板

使用下列步驟在 Amazon S3 主控台上建立 Amazon S3 Storage Lens 儀表板。

步驟 1：定義儀表板範圍

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在頁面頂端的導覽列中，選擇目前顯示的 AWS 區域名稱。接下來，選擇要切換到的區域。
3. 在左導覽窗格的 S3 Storage Lens 下，選擇儀表板。
4. 選擇 Create dashboard (建立儀表板)。
5. 在 Dashboard (儀表板) 頁面的 General (一般) 區段中，執行以下動作：

    a. 檢視儀表板的「居住地區」。「主區域」是儲 AWS 區域 存此儲存鏡頭儀表板的組態和指標的位置。

    b. 輸入儀表板名稱。

    儀表板名稱必須少於 65 個字元，且不得包含特殊字元或空格。

> **Note**
>
> 建立儀表板後，您就無法變更此儀表板名稱。

c. 您可以選擇將 Tags (標籤) 新增至儀表板。您可以使用標籤來管理儀表板的權限，並追蹤 S3 Storage Lens 的成本。

如需詳細資訊，請參閱《IAM 使用者指南》中的使用資源標籤控制存取權，以及《AWS Billing 使用者指南》中的 AWS產生的成本分配標籤。

> **Note**
>
> 您最多可為儀表板組態中新增 50 個標籤。

6. 在 Dashboard scope (儀表板範圍) 區段中，執行以下動作：

a. 選擇您希望 S3 Storage Lens 在儀表板中包含或排除的區域和儲存貯體。

b. 選擇您希望 S3 Storage Lens 包含或排除的選取區域儲存貯體。您可以包含或排除儲存貯體，但不能同時包含和排除。在您建立組織層級儀表板時，無法使用此選項。

> **Note**
>
> - 您可以包含或排除區域和儲存貯體。在組織中跨成員帳戶建立組織層級儀表板時，此選項會僅限於「區域」。
> - 您最多可以選擇 50 個要包含或排除的儲存貯體。

步驟 2：設定指標選擇

1. 在 Metrics selection (指標選取) 區段中，選取您要為此儀表板彙總的指標類型。

- 若要包含儲存貯體層級彙總的免費指標，且其可供查詢使用 14 天，請選擇 Free Metrics (免費指標)。
- 若要啟用進階指標和其他進階選項，請選擇 Advanced metrics and recommendations (進階指標和建議)。這些選項包括進階前綴彙總、Amazon CloudWatch 發佈和情境式建議。資料有 15 個

月的時間可用於查詢。進階指標和建議會有額外的費用。如需詳細資訊，請參閱 Amazon S3 定價。

如需進階指標和免費指標的詳細資訊，請參閱 指標選擇。

2. 在 Advanced metrics and recommendations features (進階指標和建議功能) 下，選取您想要啟用的選項：

   - Advanced metrics (進階指標)
   - CloudWatch 出版
   - 字首彙總

> **Important**
>
> 如果您為 S3 儲存鏡頭組態啟用前置詞彙總，則不會將前綴層級指標發佈到。 CloudWatch 只有儲存貯體、帳戶和組織層級 S3 儲存鏡頭指標會發佈到。 CloudWatch

3. 如果您已啟用 Advanced metrics (進階指標)，請選取要在 S3 Storage Lens 儀表板中顯示的 Advanced metrics categories (進階指標類別)：

   - 活動指標
   - Detailed status code metrics (詳細的狀態碼指標)
   - Advanced cost optimization metrics (進階成本最佳化指標)
   - Advanced data protection metrics (進階資料保護指標)

   如需指標類別的詳細資訊，請參閱 指標類別。如需指標的完整清單，請參閱Amazon S3 Storage Lens 指標詞彙表。

4. 如果您選擇啟用字首彙總，請設定下列項目：

   a. 請為此儀表板選擇最小字首閥值大小。

      例如，字首閾值 5% 表示組成儲存貯體總儲存體大小 5% 或更大的字首將會彙總。

   b. 選擇字首深度。

      此設定表示可評估字首的最多層級數目。字首深度必須小於 10。

   c. 輸入字首分隔符號字元。

此值用來識別每個字首層級。Amazon S3 中的預設值是 / 字元，但您的儲存結構可能會使用其他分隔符號字元。

(選用) 步驟 3：儀表板的匯出指標

1. 在 Metrics export (指標匯出) 區段中，若要建立每日將放置在您選擇的目的地儲存貯體中的指標匯出，請選擇 Enable (啟用)。

   指標匯出採用 CSV 或 Apache Parquet 格式。它代表與 S3 Storage Lens 儀表板資料相同的資料範圍，而不提供建議。

2. 如果您已啟用指標匯出，請選擇每日指標匯出的輸出格式：CSV 或 Apache Parquet。

   Parquet 是 Hadoop 的開放原始碼檔案格式，以平面直欄式格式儲存巢狀資料。

3. 選擇指標匯出的目的地 S3 儲存貯體。

   您可以在 S3 Storage Lens 儀表板的目前帳戶中選擇一個儲存貯體。或者， AWS 帳戶 如果您具有目的地值區權限和目的地值區擁有者的帳戶 ID，則可以選擇另一個。

4. 選擇目的地 S3 儲存貯體 (格式：`s3://bucket-name/prefix`)。

   儲存貯體必須位於 S3 Storage Lens 儀表板的主要區域中。S3 主控台會顯示將由 Amazon S3 新增至目的地儲存貯體政策的 Destination bucket permission (目的地儲存貯體許可)。Amazon S3 會更新目的地儲存貯體上的儲存貯體政策，以允許 S3 將資料置於該儲存貯體中。

5. (選用) 若要啟用伺服器端加密進行指標匯出，請選擇 Specify an encryption key (指定加密金鑰)。然後，選擇加密類型：Amazon S3 受管金鑰 (SSE-S3) 或 AWS Key Management Service 金鑰 (SSE-KMS)。

   您可以在 Amazon S3 受管金鑰 (SSE-S3) 和 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 之間進行選擇。

6. (選擇性) 若要指定金 AWS KMS 鑰，您必須選擇 KMS 金鑰或輸入金鑰 Amazon 資源名稱 (ARN)。

   如果您選擇客戶受管金鑰，您必須授予 S3 Storage Lens 許可，才能在 AWS KMS 金鑰政策中加密。如需詳細資訊，請參閱 使用 AWS KMS key 來加密指標匯出。

7. 選擇 Create dashboard (建立儀表板)。

若要進一步了解您的儲存，您可以建立一或多個 S3 Storage Lens 群組，並將其連接至儀表板。S3 Storage Lens 群組是根據字首、尾碼、物件標籤、物件大小、物件存留期為物件定義的自訂篩選條件，或這些篩選條件的組合。

您可以使用 S3 Storage Lens 群組取得對大型共用儲存貯體 (例如資料湖) 的精細可見度，以做出更明智的商業決策。例如，您可以將儲存用量劃分為個別專案的特定物件群組以及單一或多個儲存貯體內的成本中心，以簡化儲存配置並最佳化成本報告。

若要使用 S3 Storage Lens 群組，您必須升級儀表板以使用進階指標和建議。如需關於 S3 Storage Lens 群組的詳細資訊，請參閱 the section called “使用 Amazon S3 Storage Lens 群組”。

正在更新 Amazon S3 Storage Lens 儀表板

使用下列步驟在 Amazon S3 主控台上更新 Amazon S3 Storage Lens 儀表板。

步驟 1：更新儀表板範圍

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左導覽窗格中，選擇 Storage Lens, Dashboards (Storage Lens、儀表板)。
3. 選擇您要編輯的儀表板，然後選擇 Edit (編輯)。

   Edit dashboard (編輯儀表板) 頁面即會開啟。

   > **Note**
   >
   > 您無法變更下列項目：
   >
   > - 儀表板名稱
   > - 主要區域
   > - 預設儀表板的儀表板範圍，其範圍限定在整個帳戶的儲存體

4. (選用) 在儀表板組態頁面的 General (一般) 區段中，請更新標籤並將其新增至儀表板。

   您可以使用標籤來管理儀表板的許可，並追蹤 S3 Storage Lens 的成本。如需詳細資訊，請參閱《IAM 使用者指南》中的使用資源標籤控制存取權，以及《AWS Billing 使用者指南》中的 AWS 產生的成本分配標籤。

> **Note**
>
> 您最多可為儀表板組態中新增 50 個標籤。

5. 在 Dashboard scope (儀表板範圍) 區段中，執行以下動作：

   a. 更新您希望 S3 Storage Lens 在儀表板中包含或排除的區域和儲存貯體。

   > **Note**
   >
   > - 您可以包含或排除區域和儲存貯體。在組織中跨成員帳戶建立組織層級儀表板時，此選項會僅限於「區域」。
   > - 您最多可以選擇 50 個要包含或排除的儲存貯體。

   b. 更新您希望 S3 Storage Lens 包含或排除的選定區域儲存貯體。您可以包含或排除儲存貯體，但不能同時包含和排除。建立組織層級儀表板時，不會出現此選項。

步驟 2：更新指標選擇

1. 在 Metrics selection (指標選取) 區段中，選取您要為此儀表板彙總的指標類型。

   - 若要包含儲存貯體層級彙總的免費指標，且其可供查詢使用 14 天，請選擇 Free Metrics (免費指標)。
   - 若要啟用進階指標和其他進階選項，請選擇 Advanced metrics and recommendations (進階指標和建議)。這些選項包括進階前綴彙總、Amazon CloudWatch 發佈和情境式建議。資料有 15 個月的時間可用於查詢。進階指標和建議會有額外的費用。如需詳細資訊，請參閱 Amazon S3 定價。

     如需進階指標和免費指標的詳細資訊，請參閱 指標選擇。

2. 在 Advanced metrics and recommendations features (進階指標和建議功能) 下，選取您想要啟用的選項：

   - Advanced metrics (進階指標)
   - CloudWatch 出版
   - 字首彙總

> **Important**
>
> 如果您為 S3 儲存鏡頭組態啟用前置詞彙總，則不會將前綴層級指標發佈到。 CloudWatch 只有儲存貯體、帳戶和組織層級 S3 儲存鏡頭指標會發佈到。 CloudWatch

3. 如果您已啟用 Advanced metrics (進階指標)，請選取要在 S3 Storage Lens 儀表板中顯示的 Advanced metrics categories (進階指標類別)：

   - 活動指標
   - Detailed status code metrics (詳細的狀態碼指標)
   - Advanced cost optimization metrics (進階成本最佳化指標)
   - Advanced data protection metrics (進階資料保護指標)

   如需指標類別的詳細資訊，請參閱 指標類別。如需指標的完整清單，請參閱Amazon S3 Storage Lens 指標詞彙表。

4. 如果您選擇啟用字首彙總，請設定下列項目：

   a. 請為此儀表板選擇最小字首閥值大小。

      例如，字首閾值 5% 表示組成儲存貯體總儲存體大小 5% 或更大的字首將會彙總。

   b. 選擇字首深度。

      此設定表示可評估字首的最多層級數目。字首深度必須小於 10。

   c. 輸入字首分隔符號字元。

      這是用來識別每個字首層級的值。Amazon S3 中的預設值是 / 字元，但您的儲存結構可能會使用其他分隔符號字元。

(選用) 步驟 3：儀表板的匯出指標

1. 在 Metrics export (指標匯出) 區段中，若要建立每日將放置在您選擇的目的地儲存貯體中的指標匯出，請選擇 Enable (啟用)。若要停用指標匯出，請選擇 Disable (停用)。

   指標匯出採用 CSV 或 Apache Parquet 格式。它代表與 S3 Storage Lens 儀表板資料相同的資料範圍，而不提供建議。

2. 若已啟用，請選擇每日指標匯出的輸出格式：CSV 或 Apache Parquet。

Parquet 是 Hadoop 的開放原始碼檔案格式，以平面直欄式格式儲存巢狀資料。

3. 選擇指標匯出的目的地 S3 儲存貯體。

   您可以在 S3 Storage Lens 儀表板的目前帳戶中選擇一個儲存貯體。或者， AWS 帳戶 如果您具有目的地值區權限和目的地值區擁有者的帳戶 ID，則可以選擇另一個。

4. 選擇目的地 S3 儲存貯體 (格式：`s3://bucket-name/prefix`)。

   儲存貯體必須位於 S3 Storage Lens 儀表板的主要區域中。S3 主控台會顯示將由 Amazon S3 新增至目的地儲存貯體政策的 Destination bucket permission (目的地儲存貯體許可)。Amazon S3 會更新目的地儲存貯體上的儲存貯體政策，以允許 S3 將資料置於該儲存貯體中。

5. (選用) 若要啟用伺服器端加密進行指標匯出，請選擇 Specify an encryption key (指定加密金鑰)。然後，選擇加密類型：Amazon S3 受管金鑰 (SSE-S3) 或 AWS Key Management Service 金鑰 (SSE-KMS)。

   您可以在 Amazon S3 受管金鑰 (SSE-S3) 和 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 之間進行選擇。

6. (選擇性) 若要指定金 AWS KMS 鑰，您必須選擇 KMS 金鑰或輸入金鑰 Amazon 資源名稱 (ARN)。在 AWS KMS 金鑰下，使用下列其中一種方式指定 KMS 金鑰：

   - 若要從可用的 KMS 金鑰清單中選擇，請選擇從 AWS KMS keys中選擇，然後從可用金鑰清單中選擇您的 KMS 金鑰。

     AWS 受管金鑰 (aws/s3) 和您的客戶管理金鑰都會出現在此清單中。如需詳細了解客戶受管金鑰，請參閱《AWS Key Management Service 開發人員指南》中的客戶金鑰和 AWS 金鑰。

     > **Note**
     >
     > 不支援使用 S3 儲存鏡頭進行 SSE-KMS 加密的 AWS 受管金鑰 (aws/S3)。

   - 若要輸入 KMS 金鑰 ARN，請選擇輸入 AWS KMS key ARN，然後在出現的欄位中輸入您的 KMS 金鑰 ARN。
   - 若要在 AWS KMS 主控台中建立新的客戶管理金鑰，請選擇 [建立 KMS 金鑰]。

     如果您選擇客戶受管金鑰，您必須授予 S3 Storage Lens 許可，才能在 AWS KMS 金鑰政策中加密。如需詳細資訊，請參閱 使用 AWS KMS key 來加密指標匯出。

     如需有關建立金鑰的詳細資訊 AWS KMS key，請參閱AWS Key Management Service 開發人員指南中的建立金鑰。

7. 選擇儲存變更。

若要進一步了解您的儲存，您可以建立一或多個 S3 Storage Lens 群組，並將其連接至儀表板。S3 Storage Lens 群組是根據字首、尾碼、物件標籤、物件大小、物件存留期為物件定義的自訂篩選條件，或這些篩選條件的組合。

您可以使用 S3 Storage Lens 群組取得對大型共用儲存貯體 (例如資料湖) 的精細可見度，以做出更明智的商業決策。例如，您可以將儲存用量劃分為個別專案的特定物件群組以及單一或多個儲存貯體內的成本中心，以簡化儲存配置並最佳化成本報告。

若要使用 S3 Storage Lens 群組，您必須升級儀表板以使用進階指標和建議。如需關於 S3 Storage Lens 群組的詳細資訊，請參閱 the section called “使用 Amazon S3 Storage Lens 群組”。

正在停用或刪除 Amazon S3 Storage Lens 儀表板

S3 Storage Lens 會彙總您的指標，並在 Amazon S3 主控台 Buckets (儲存貯體) 頁面上的 Account snapshot (帳戶快照) 區段中顯示資訊。S3 Storage Lens 也會提供互動式儀表板，您可以用來以視覺化方式呈現洞見與趨勢、標記異常值，以及接收最佳化儲存成本和套用資料保護最佳實務的建議。您的儀表板具有向下切入選項可產生及視覺化組織、帳戶、 AWS 區域、儲存類別、儲存貯體、字首或 Storage Lens 群組層級的洞察。您也可以將 CSV 或 Parquet 格式的每日指標匯出傳送到 S3 儲存貯體。

Amazon S3 儲存鏡頭的預設儀表板為default-account-dashboard。此儀表板是由 Amazon S3 預先設定，可協助您在主控台上以視覺化方式呈現整個帳戶彙總的免費和進階指標的摘要洞見和趨勢。您無法修改預設儀表板的組態範圍，但您可以將指標選項從免費指標升級為付費進階指標和建議、設定選用指標匯出，甚至停用預設儀表板。無法刪除預設儀表板。

您可以從 Amazon S3 主控台刪除或停用 Amazon S3 Storage Lens 儀表板。停用或刪除儀表板可防止它未來產生指標。已停用的儀表板仍會保留其組態資訊，以便在重新啟用時可以輕鬆恢復。停用的儀表板會保留其歷史資料，直到資料不再可用於查詢。

免費指標選項的資料可供查詢使用 14 天，而進階指標和建議選項的資料可供查詢使用 15 個月。

正在停用 Amazon S3 Storage Lens 儀表板

停用 S3 Storage Lens 儀表板

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。

3. 在 Dashboards (儀表板) 清單中，選擇您要停用的儀表板，然後選擇清單頂端的 Disable (停用)。
4. 在確認頁面上，於文字欄位中輸入儀表板名稱，確認您要停用儀表板，然後選擇確認。

正在刪除 Amazon S3 Storage Lens 儀表板

> **Note**
>
> 您無法刪除預設儀表板。不過，您可以停用它。在刪除您已建立的儀表板之前，請考慮下列事項：
>
> - 除了刪除儀表板之外，您還可以停用儀表板，以便將來可以重新啟用儀表板。如需詳細資訊，請參閱 正在停用 Amazon S3 Storage Lens 儀表板。
> - 刪除儀表板會刪除與其相關聯的所有組態設定。
> - 刪除儀表板會使所有歷史指標資料無法使用。這些歷史資料仍會保留 15 個月。如果您想要再次存取此資料，請在與已刪除的主要區域相同的主要區域中建立具有相同名稱的儀表板。

刪除 S3 Storage Lens 儀表板

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左導覽窗格中，選擇 Storage Lens、Dashboards (儀表板)。
3. 在 Dashboards (儀表板) 清單中，選擇您要刪除的儀表板，然後選擇清單頂端的 Delete (刪除)。
4. 在刪除儀表板頁面上，於文字欄位中輸入儀表板名稱，確認您要刪除儀表板。然後選擇 Confirm (確認)。

使用建立組織層級儀 AWS Organizations 表板

S3 Storage Lens 會彙總您的指標，並在 Amazon S3 主控台 Buckets (儲存貯體) 頁面上的 Account snapshot (帳戶快照) 區段中顯示資訊。S3 Storage Lens 也會提供互動式儀表板，您可以用來以視覺化方式呈現洞見與趨勢、標記異常值，以及接收最佳化儲存成本和套用資料保護最佳實務的建議。您的儀表板具有向下切入選項可產生及視覺化組織、帳戶、 AWS 區域、儲存類別、儲存貯體、字首或 Storage Lens 群組層級的洞察。您也可以將 CSV 或 Parquet 格式的每日指標匯出傳送到 S3 儲存貯體。

Amazon S3 儲存鏡頭的預設儀表板為default-account-dashboard。此儀表板是由 Amazon S3 預先設定，可協助您在主控台上以視覺化方式呈現整個帳戶彙總的免費和進階指標的摘要洞見和趨勢。您無法

修改預設儀表板的組態範圍，但您可以將指標選項從免費指標升級為付費進階指標和建議、設定選用指標匯出，甚至停用預設儀表板。無法刪除預設儀表板。

您也可以建立專注於組織 AWS 帳戶 中特定 AWS 區域、S3 儲存貯體或其他儲存貯體的其他 S3 儲存貯體的其他 S3 儲存鏡頭儀表板。

S3 Storage Lens 儀表板針對其儲存範圍的相關資訊提供了豐富的資源。儀表板視覺化超過 30 個代表趨勢和資訊的指標，包括儲存摘要、成本效益、資料保護和活動。

Amazon S3 儲存鏡頭可用於收集屬於 AWS Organizations 階層一部分的所有帳戶的儲存指標和使用情況資料。若要這樣做，您必須使用 AWS Organizations，而且必須使用 AWS Organizations 管理帳戶啟用 S3 Storage Lens 受信任存取。

受信任的存取權啟用後，您可以將委派的管理員存取權新增至組織中的帳戶。這些帳戶接著可以為 S3 Storage Lens 建立全組織的儀表板和設定。如需有關啟用受信任存取權的詳細資訊，請參閱《 AWS Organizations使用者指南》中的 Amazon S3 Lens 和AWS Organizations。

下列主控台控制項僅適用於 AWS Organizations 管理帳戶。

在您的組織中啟用 S3 Storage Lens 的受信任存取權

啟用受信任存取可讓 Amazon S3 儲存鏡頭透過 AWS Organizations API 操作存取您的 AWS Organizations 階層、會員資格和結構。S3 Storage Lens 成為整個組織結構的受信任服務。每當建立儀表板組態時，它會在您組織的管理或委派管理員帳戶中建立服務連結角色。

服務連結的角色會授予 S3 Storage Lens 許可，以描述組織、列出帳戶、驗證組織的服務存取清單，以及取得組織的委派管理員。這可讓 S3 Storage Lens 接著收集組織中帳戶內儀表板的跨帳戶儲存使用量和活動指標。

如需詳細資訊，請參閱 讓 Amazon S3 Storage Lens 使用服務連結角色。

> **Note**
>
> - 受信任存取權只能由「管理帳戶」啟用。
> - 只有管理帳戶和委派的管理員可以為您的組織建立 S3 Storage Lens 儀表板或組態。

啟用 S3 Storage Lens 來取得受信任的存取權

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

2. 在左導覽窗格中，選擇 Storage Lens、Organization settings (組織設定)。
3. 在 Organizations access (組織存取) 中，選擇 Edit (編輯)。

   Organization access (組織存取) 頁面即可開啟。您可以在此處啟用 S3 Storage Lens 的受信任存取權。這可讓您和新增為委派管理員的任何其他帳戶擁有者，為組織中的所有帳戶和儲存體建立儀表板。

在您的組織中停用 S3 Storage Lens 的受信任存取權

停用受信任存取權會限制 S3 Storage Lens 只能在帳戶層級上運作。每個帳戶持有人只能看到 S3 Storage Lens 的好處，僅限於其帳戶範圍，而不能看到其組織範圍。任何需要受信任存取權的儀表板都不會再更新，但這些儀表板可以根據各自的資料可用於查詢時段來查詢其歷史資料。

以委派管理員身分移除帳戶，會將帳戶擁有者的 S3 Storage Lens 儀表板指標的存取權限制在只能在帳戶層級上運作。其所建立的任何組織儀表板都不會再更新，但其可以根據各自的資料可用於查詢時段來查詢其歷史資料。

> **Note**
>
> - 停用受信任存取權也會自動停用所有組織層級的儀表板，因為 S3 Storage Lens 將不再具有組織帳戶的受信任存取權，無法再收集和彙總儲存指標。
> - 管理和委派管理員帳戶仍然可以查看這些已停用儀表板的歷史資料，並可以在可用時查詢此資料。

停用 S3 Storage Lens 的受信任存取權

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左導覽窗格中，選擇 Storage Lens、Organization settings (組織設定)。
3. 在 Organizations access (組織存取) 中，選擇 Edit (編輯)。

   Organization access (組織存取) 頁面即可開啟。您可以在此停用 S3 Storage Lens 的受信任存取權。

正在註冊 S3 Storage Lens 的委派管理員

啟用受信任存取權之後，您就可以註冊委派管理員的存取權，讓管理員存取組織中的帳戶。當帳戶註冊為委派管理員時，帳戶會收到存取所有唯讀 AWS Organizations API 作業的授權。此授權會提供組織成員和結構的可見性，讓他們可以代表您建立 S3 Storage Lens 儀表板。

註冊 S3 Storage Lens 的委派管理員

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左導覽窗格中，選擇 Storage Lens、Organization settings (組織設定)。
3. 在 delegated access (委派存取權) 區段中，在 Accounts (帳戶) 下選擇 Add account (新增帳戶)。

   Delegated admin access (委派管理員存取權) 頁面即可開啟。您可以在此處新增 AWS 帳戶 ID 作為委派管理員，為組織中的所有帳戶和儲存體建立組織層級的儀表板。

正在取消註冊 S3 Storage Lens 的委派管理員

您可以取消註冊委派管理員存取權，讓管理員無法存取組織中的帳戶。當帳戶被取消註冊為委派的系統管理員時，帳戶會失去存取所有唯讀 AWS Organizations API 作業的授權，這些作業可讓您看到組織的成員和結構。

> **Note**
>
> - 取消註冊委派管理員也會自動停用委派管理員建立的所有組織層級儀表板。
> - 委派管理員帳戶仍然可以根據各自的資料可用於查詢時段，查看這些已停用儀表板的歷史資料。

取消註冊委派管理員存取權的帳號

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左導覽窗格中，選擇 Storage Lens、Organization settings (組織設定)。
3. 在 Accounts with delegated access (具有委派存取權的帳戶) 區段中，選擇您要取消註冊的帳戶 ID，然後選擇 Remove (移除)。

使用 AWS CLI 的 Amazon S3 Storage Lens 範例

S3 Storage Lens 會彙總您的指標，並在 Amazon S3 主控台 Buckets (儲存貯體) 頁面上的 Account snapshot (帳戶快照) 區段中顯示資訊。S3 Storage Lens 也會提供互動式儀表板，您可以用來以視覺化方式呈現洞見與趨勢、標記異常值，以及接收最佳化儲存成本和套用資料保護最佳實務的建議。您的儀表板具有向下切入選項可產生及視覺化組織、帳戶、AWS 區域、儲存類別、儲存貯體、字首或 Storage Lens 群組層級的洞察。您也可以將 CSV 或 Parquet 格式的每日指標匯出傳送到 S3 儲存貯體。如需詳細資訊，請參閱使用 Amazon S3 Storage Lens 評估儲存活動和使用量。

下列範例示範如何搭配 AWS Command Line Interface 使用 S3 Storage Lens。

主題

使用 Amazon S3 Storage Lens 的 Helper 檔案

為您的範例使用下列 JSON 檔案及其金鑰輸入。

使用 JSON 的 S3 Storage Lens 範例組態

Example **`config.json`**

`config.json` 檔案包含 S3 Storage Lens 組織層級進階指標和建議組態的詳細資訊。若要使用下列範例，請以您自己的資訊取代 *`user input placeholders`*。

> **Note**
>
> 進階指標和建議需支付額外費用。如需詳細資訊，請參閱進階指標和建議。

```
{
  "Id": "SampleS3StorageLensConfiguration", //Use this property to identify your S3
 Storage Lens configuration.
  "AwsOrg": { //Use this property when enabling S3 Storage Lens for AWS Organizations.
    "Arn": "arn:aws:organizations::123456789012:organization/o-abcdefgh"
  },
  "AccountLevel": {
    "ActivityMetrics": {
```

```
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "DetailedStatusCodesMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
        "IsEnabled":true
      },
      "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
      },
      "AdvancedCostOptimizationMetrics": {
        "IsEnabled":true
      },
      "DetailedStatusCodesMetrics": {
        "IsEnabled":true
      },
      "PrefixLevel":{
        "StorageMetrics":{
          "IsEnabled":true,
          "SelectionCriteria":{
            "MaxDepth":5,
            "MinStorageBytesPercentage":1.25,
            "Delimiter":"/"
          }
        }
      }
    }
  },
  "Exclude": { //Replace with "Include" if you prefer to include Regions.
    "Regions": [
      "eu-west-1"
    ],
    "Buckets": [ //This attribute is not supported for AWS Organizations-level
configurations.
      "arn:aws:s3:::source_bucket1"
    ]
```

```
  },
  "IsEnabled": true, //Whether the configuration is enabled
  "DataExport": { //Details about the metrics export
    "S3BucketDestination": {
      "OutputSchemaVersion": "V_1",
      "Format": "CSV", //You can add "Parquet" if you prefer.
      "AccountId": "111122223333",
      "Arn": "arn:aws:s3:::destination-bucket-name", // The destination bucket for your
 metrics export must be in the same Region as your S3 Storage Lens configuration.
      "Prefix": "prefix-for-your-export-destination",
      "Encryption": {
        "SSES3": {}
      }
    },
    "CloudWatchMetrics": {
      "IsEnabled": true
    }
  }
}
```

使用 JSON Storage Lens 群組的 S3 Storage Lens 範例組態

Example **`config.json`**

`config.json` 檔案包含您在使用 Storage Lens 群組時要套用至 Storage Lens 組態的詳細資訊。若要使用該範例，請將 *`user input placeholders`* 取代為您自己的資訊。

若要將所有 Storage Lens 群組連接至儀表板，請使用下列語法更新您的 Storage Lens 組態：

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
```

```
      },
    "StorageLensGroupLevel": {},
  "IsEnabled": true
}
```

若您的 Storage Lens 儀表板組態中只要包含兩個 Storage Lens 群組 (*slg-1* 和 *slg-2*)，請使用下列語法：

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
   "StorageLensGroupLevel": {
        "SelectionCriteria": {
            "Include": [
                "arn:aws:s3:us-east-1:111122223333:storage-lens-group/slg-1",
                "arn:aws:s3:us-east-1:444455556666:storage-lens-group/slg-2"
            ]
    },
  "IsEnabled": true
}
```

若只有特定 Storage Lens 群組不要連接至儀表板組態，請使用下列語法：

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
```

```
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
   "StorageLensGroupLevel": {
        "SelectionCriteria": {
            "Exclude": [
                "arn:aws:s3:us-east-1:111122223333:storage-lens-group/slg-1",
                "arn:aws:s3:us-east-1:444455556666:storage-lens-group/slg-2"
            ]
    },
  "IsEnabled": true
}
```

使用 JSON 的 S3 Storage Lens 範例標籤組態

Example **tags.json**

tags.json 檔案包含您想要套用至 S3 Storage Lens 組態的標籤。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
[
    {
        "Key": "key1",
        "Value": "value1"
    },
    {
        "Key": "key2",
        "Value": "value2"
    }
]
```

S3 Storage Lens 範例組態 IAM 許可

Example **permissions.json** - 特定儀表板名稱

此範例政策顯示已指定特定儀表板名稱的 S3 Storage Lens IAM permissions.json 檔案。將 *value1*、*us-east-1*、*your-dashboard-name* 和 *example-account-id* 取代為您自己的值。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetStorageLensConfiguration",
                "s3:DeleteStorageLensConfiguration",
                "s3:PutStorageLensConfiguration"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/key1": "value1"
                }
            },
            "Resource": "arn:aws:s3:us-east-1:example-account-id:storage-lens/your-dashboard-name"
        }
    ]
}
```

Example **permissions.json** - 沒有特定的儀表板名稱

此範例政策顯示未指定特定儀表板名稱的 S3 Storage Lens IAM permissions.json 檔案。用您的數值取代 *value1*、*us-east-1* 和 *example-account-id*。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetStorageLensConfiguration",
                "s3:DeleteStorageLensConfiguration",
                "s3:PutStorageLensConfiguration"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/key1": "value1"
                }
            },
            "Resource": "arn:aws:s3:us-east-1:example-account-id:storage-lens/*"
```

```
        }
    ]
}
```

搭配 AWS CLI 使用 Amazon S3 Storage Lens 組態

您可以使用 AWS CLI 來列出、建立、刪除、取得、標記和更新 S3 Storage Lens 組態。下面的範例使用 helper JSON 檔案進行金鑰輸入。若要使用這些範例，請以您自己的資訊取代 *user input placeholders*。

建立 S3 Storage Lens 組態

Example 建立 S3 Storage Lens 組態

```
aws s3control put-storage-lens-configuration --account-id=111122223333 --
config-id=example-dashboard-configuration-id --region=us-east-1 --storage-lens-
configuration=file://./config.json --tags=file://./tags.json
```

建立無索引標籤的 S3 Storage Lens 組態

Example 建立無索引標籤的 S3 Storage Lens 組態

```
aws s3control put-storage-lens-configuration --account-id=222222222222 --config-
id=your-configuration-id --region=us-east-1 --storage-lens-configuration=file://./
config.json
```

取得 S3 Storage Lens 組態

Example 取得 S3 Storage Lens 組態

```
aws s3control get-storage-lens-configuration --account-id=222222222222 --config-
id=your-configuration-id --region=us-east-1
```

列出沒有下一個字符的 S3 Storage Lens 組態

Example 列出沒有下一個字符的 S3 Storage Lens 組態

```
aws s3control list-storage-lens-configurations --account-id=222222222222 --region=us-
east-1
```

列出 S3 Storage Lens 組態

Example 列出 S3 Storage Lens 組態

```
aws s3control list-storage-lens-configurations --account-id=222222222222 --region=us-east-1 --next-token=abcdefghij1234
```

刪除 S3 Storage Lens 組態

Example 刪除 S3 Storage Lens 組態

```
aws s3control delete-storage-lens-configuration --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

將索引標籤新增至 S3 Storage Lens 組態

Example 將索引標籤新增至 S3 Storage Lens 組態

```
aws s3control put-storage-lens-configuration-tagging --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id --tags=file://./tags.json
```

取得 S3 Storage Lens 組態的標籤

Example 取得 S3 Storage Lens 組態的標籤

```
aws s3control get-storage-lens-configuration-tagging --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

刪除 S3 Storage Lens 組態的標籤

Example 刪除 S3 Storage Lens 組態的標籤

```
aws s3control delete-storage-lens-configuration-tagging --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

透過 AWS CLI 搭配 AWS Organizations 範例使用 Amazon S3 Storage Lens

使用 Amazon S3 Storage Lens，來收集屬於 AWS Organizations 階層之所有帳戶的儲存指標和使用量資料。如需詳細資訊，請參閱將 Amazon S3 Storage Lens 與 AWS Organizations 搭配使用。

啟用 S3 Storage Lens 的 Organizations 受信任存取權

Example 啟用 S3 Storage Lens 的 Organizations 受信任存取權

```
aws organizations enable-aws-service-access --service-principal storage-
lens.s3.amazonaws.com
```

停用 S3 Storage Lens 的 Organizations 受信任存取權

Example 停用 S3 Storage Lens 的 Organizations 受信任存取權

```
aws organizations disable-aws-service-access --service-principal storage-
lens.s3.amazonaws.com
```

註冊 S3 Storage Lens 的 Organizations 委派管理員

Example 註冊 S3 Storage Lens 的 Organizations 委派管理員

若要使用此範例，請將 *111122223333* 取代為適當的 AWS 帳戶 ID。

```
aws organizations register-delegated-administrator --service-principal storage-
lens.s3.amazonaws.com --account-id 111122223333
```

取消註冊 S3 Storage Lens 的 Organizations 委派管理員

Example 取消註冊 S3 Storage Lens 的 Organizations 委派管理員

若要使用此範例，請將 *111122223333* 取代為適當的 AWS 帳戶 ID。

```
aws organizations deregister-delegated-administrator --service-principal storage-
lens.s3.amazonaws.com --account-id 111122223333
```

使用 Java 開發套件的 Amazon S3 Storage Lens 範例

S3 Storage Lens 會彙總您的指標，並在 Amazon S3 主控台 Buckets (儲存貯體) 頁面上的 Account snapshot (帳戶快照) 區段中顯示資訊。S3 Storage Lens 也會提供互動式儀表板，您可以用來以視覺化方式呈現洞見與趨勢、標記異常值，以及接收最佳化儲存成本和套用資料保護最佳實務的建議。您的儀表板具有向下切入選項可產生及視覺化組織、帳戶、AWS 區域、儲存類別、儲存貯體、字首或 Storage Lens 群組層級的洞察。您也可以將 CSV 或 Parquet 格式的每日指標匯出傳送到 S3 儲存貯體。如需詳細資訊，請參閱使用 Amazon S3 Storage Lens 評估儲存活動和使用量。

下列範例示範如何搭配 AWS SDK for Java 使用 S3 Storage Lens。

主題

- 使用適用於 Java 的開發套件使用 Amazon S3 Storage Lens 組態

使用適用於 Java 的開發套件使用 Amazon S3 Storage Lens 組態

您可以使用 Java 開發套件範例來列出、建立、取得並更新 S3 Storage Lens 組態。下面的範例使用 helper JSON 檔案進行金鑰輸入。

主題

- 建立及更新 S3 Storage Lens 組態
- 刪除 S3 Storage Lens 組態
- 取得 S3 Storage Lens 組態
- 列出 S3 Storage Lens 組態
- 將索引標籤新增至 S3 Storage Lens 組態
- 取得 S3 Storage Lens 組態的標籤
- 刪除 S3 Storage Lens 組態的標籤
- 使用進階指標和建議來更新預設 S3 Storage Lens 組態
- 將 Storage Lens 群組連接至 S3 Storage Lens 儀表板
- 透過適用於 Java 的 SDK 搭配 AWS Organizations 範例使用 Amazon S3 Storage Lens

建立及更新 S3 Storage Lens 組態

Example 建立及更新 S3 Storage Lens 組態

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.CloudWatchMetrics;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
```

```
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateAndUpdateDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        String exportAccountId = "Destination Account ID";
        String exportBucketArn = "arn:aws:s3:::destBucketName"; // The destination
 bucket for your metrics export must be in the same Region as your S3 Storage Lens
 configuration.
        String awsOrgARN = "arn:aws:organizations::123456789012:organization/o-
abcdefgh";
        Format exportFormat = Format.CSV;

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
```

```
                    .withAdvancedDataProtectionMetrics(new
AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            Include include = new Include()
                    .withBuckets(Arrays.asList("arn:aws:s3:::bucketName"))
                    .withRegions(Arrays.asList("us-west-2"));

            StorageLensDataExportEncryption exportEncryption = new
StorageLensDataExportEncryption()
                    .withSSES3(new SSES3());
            S3BucketDestination s3BucketDestination = new S3BucketDestination()
                    .withAccountId(exportAccountId)
                    .withArn(exportBucketArn)
                    .withEncryption(exportEncryption)
                    .withFormat(exportFormat)
                    .withOutputSchemaVersion(OutputSchemaVersion.V_1)
                    .withPrefix("Prefix");
            CloudWatchMetrics cloudWatchMetrics = new CloudWatchMetrics()
                    .withIsEnabled(true);
            StorageLensDataExport dataExport = new StorageLensDataExport()
                    .withCloudWatchMetrics(cloudWatchMetrics)
                    .withS3BucketDestination(s3BucketDestination);

            StorageLensAwsOrg awsOrg = new StorageLensAwsOrg()
                    .withArn(awsOrgARN);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withInclude(include)
                    .withDataExport(dataExport)
```

```
                    .withAwsOrg(awsOrg)
                    .withIsEnabled(true);

            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

刪除 S3 Storage Lens 組態

Example 刪除 S3 Storage Lens 組態

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
```

```
import com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class DeleteDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.deleteStorageLensConfiguration(new
 DeleteStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

取得 S3 Storage Lens 組態

Example 取得 S3 Storage Lens 組態

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.GetStorageLensConfigurationRequest;
```

```
import com.amazonaws.services.s3control.model.GetStorageLensConfigurationResult;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class GetDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            final StorageLensConfiguration configuration =
                    s3ControlClient.getStorageLensConfiguration(new
 GetStorageLensConfigurationRequest()
                            .withAccountId(sourceAccountId)
                            .withConfigId(configurationId)
                    ).getStorageLensConfiguration();

            System.out.println(configuration.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

列出 S3 Storage Lens 組態

Example 列出 S3 Storage Lens 組態

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
```

```
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.ListStorageLensConfigurationEntry;
import com.amazonaws.services.s3control.model.ListStorageLensConfigurationsRequest;

import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class ListDashboard {

    public static void main(String[] args) {
        String sourceAccountId = "Source Account ID";
        String nextToken = "nextToken";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            final List<ListStorageLensConfigurationEntry> configurations =
                    s3ControlClient.listStorageLensConfigurations(new
 ListStorageLensConfigurationsRequest()
                            .withAccountId(sourceAccountId)
                            .withNextToken(nextToken)
                    ).getStorageLensConfigurationList();

            System.out.println(configurations.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

將索引標籤新增至 S3 Storage Lens 組態

Example 將索引標籤新增至 S3 Storage Lens 組態

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import
 com.amazonaws.services.s3control.model.PutStorageLensConfigurationTaggingRequest;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class PutDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";

        try {
            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfigurationTagging(new
 PutStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
```

```
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

取得 S3 Storage Lens 組態的標籤

Example 取得 S3 Storage Lens 組態的標籤

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationRequest;
import
 com.amazonaws.services.s3control.model.GetStorageLensConfigurationTaggingRequest;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class GetDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            final List<StorageLensTag> s3Tags = s3ControlClient
```

```
                    .getStorageLensConfigurationTagging(new
 GetStorageLensConfigurationTaggingRequest()
                            .withAccountId(sourceAccountId)
                            .withConfigId(configurationId)
                    ).getTags();

            System.out.println(s3Tags.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

刪除 S3 Storage Lens 組態的標籤

Example 刪除 S3 Storage Lens 組態的標籤

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import
 com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationTaggingRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class DeleteDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
```

```
                    .build();

            s3ControlClient.deleteStorageLensConfigurationTagging(new
 DeleteStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用進階指標和建議來更新預設 S3 Storage Lens 組態

Example 使用進階指標和建議來更新預設 S3 Storage Lens 組態

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
```

```
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateDefaultConfigWithPaidFeatures {

    public static void main(String[] args) {
        String configurationId = "default-account-dashboard"; // This configuration ID
 cannot be modified.
        String sourceAccountId = "Source Account ID";

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
 PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();
```

```
            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

> **Note**
>
> 進階指標和建議需支付額外費用。如需詳細資訊，請參閱進階指標和建議。

將 Storage Lens 群組連接至 S3 Storage Lens 儀表板

Example 將所有 Storage Lens 群組連接至儀表板

下列適用於 Java 的 SDK 範例會將帳戶 *111122223333* 中的所有 Storage Lens 群組連接至 *DashBoardConfigurationId* 儀表板：

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
```

```
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWithStorageLensGroups {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel();

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example 將兩個 Storage Lens 群組連接至儀表板

下列 AWS SDK for Java 範例會將兩個 Storage Lens 群組 (*StorageLensGroupName1* 和 *StorageLensGroupName2*) 連接至 *ExampleDashboardConfigurationId* 儀表板。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;
import com.amazonaws.services.s3control.model.StorageLensGroupLevelSelectionCriteria;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWith2StorageLensGroups {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String storageLensGroupName1 = "StorageLensGroupName1";
        String storageLensGroupName2 = "StorageLensGroupName2";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevelSelectionCriteria selectionCriteria = new
 StorageLensGroupLevelSelectionCriteria()
                    .withInclude(
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName1,
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName2);

            System.out.println(selectionCriteria);
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel()
                    .withSelectionCriteria(selectionCriteria);

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);
```

```
            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example 連接所有附帶排除項目的 Storage Lens 群組

下列適用於 Java 的 SDK 範例會將所有 Storage Lens 群組連接至 *ExampleDashboardConfigurationId* 儀表板，但兩個指定的項目 (*StorageLensGroupName1* 和 *StorageLensGroupName2*) 除外：

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
```

```
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;
import com.amazonaws.services.s3control.model.StorageLensGroupLevelSelectionCriteria;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWith2StorageLensGroupsExcluded {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String storageLensGroupName1 = "StorageLensGroupName1";
        String storageLensGroupName2 = "StorageLensGroupName2";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevelSelectionCriteria selectionCriteria = new
 StorageLensGroupLevelSelectionCriteria()
                    .withInclude(
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName1,
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName2);

            System.out.println(selectionCriteria);
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel()
                    .withSelectionCriteria(selectionCriteria);

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();
```

```
            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

透過適用於 Java 的 SDK 搭配 AWS Organizations 範例使用 Amazon S3 Storage Lens

使用 Amazon S3 Storage Lens，來收集屬於 AWS Organizations 階層之所有帳戶的儲存指標和使用量資料。如需詳細資訊，請參閱將 Amazon S3 Storage Lens 與 AWS Organizations 搭配使用。

主題

- 啟用 S3 Storage Lens 的 Organizations 受信任存取權
- 停用 S3 Storage Lens 的 Organizations 受信任存取權
- 註冊 S3 Storage Lens 的 Organizations 委派管理員
- 取消註冊 S3 Storage Lens 的 Organizations 委派管理員

啟用 S3 Storage Lens 的 Organizations 受信任存取權

Example 啟用 S3 Storage Lens 的 Organizations 受信任存取權

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import com.amazonaws.services.organizations.model.EnableAWSServiceAccessRequest;
```

```
public class EnableOrganizationsTrustedAccess {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.enableAWSServiceAccess(new
 EnableAWSServiceAccessRequest()
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

停用 S3 Storage Lens 的 Organizations 受信任存取權

Example 停用 S3 Storage Lens 的 Organizations 受信任存取權

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import com.amazonaws.services.organizations.model.DisableAWSServiceAccessRequest;

public class DisableOrganizationsTrustedAccess {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";
```

```
 public static void main(String[] args) {
  try {
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            // Make sure to remove any existing delegated administrator for S3 Storage
 Lens
            // before disabling access; otherwise, the request will fail.
            organizationsClient.disableAWSServiceAccess(new
 DisableAWSServiceAccessRequest()
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

註冊 S3 Storage Lens 的 Organizations 委派管理員

Example 註冊 S3 Storage Lens 的 Organizations 委派管理員

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import
 com.amazonaws.services.organizations.model.RegisterDelegatedAdministratorRequest;

public class RegisterOrganizationsDelegatedAdministrator {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
```

```
  try {
            String delegatedAdminAccountId = "111122223333"; // Account Id for the
 delegated administrator.
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.registerDelegatedAdministrator(new
 RegisterDelegatedAdministratorRequest()
                .withAccountId(delegatedAdminAccountId)
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

取消註冊 S3 Storage Lens 的 Organizations 委派管理員

Example 取消註冊 S3 Storage Lens 的 Organizations 委派管理員

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import
 com.amazonaws.services.organizations.model.DeregisterDelegatedAdministratorRequest;

public class DeregisterOrganizationsDelegatedAdministrator {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
```

```
            String delegatedAdminAccountId = "111122223333"; // Account Id for the
 delegated administrator.
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.deregisterDelegatedAdministrator(new
 DeregisterDelegatedAdministratorRequest()
                .withAccountId(delegatedAdminAccountId)
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

使用 Amazon S3 Storage Lens 群組

Amazon S3 Storage Lens 群組會根據物件中繼資料使用自訂篩選條件彙總指標。Storage Lens 群組可協助您深入了解資料的特性，例如依存留期、最常見的檔案類型等項目分類的物件分佈。例如，您可以依物件標籤篩選指標，以識別成長最快的資料集，或根據物件大小和存留期將儲存視覺化，以制定儲存封存策略。因此，Amazon S3 Storage Lens 群組可協助您深入了解 S3 儲存並將其最佳化。

使用 Storage Lens 群組時，您可以使用物件中繼資料 (例如字首、尾碼、物件標籤、物件大小或物件存留期) 來分析及篩選 S3 Storage Lens 指標。您也可以套用這些篩選條件的組合。將 Storage Lens 群組連接至 S3 Storage Lens 儀表板後，您可以直接在儀表板中檢視以 Amazon S3 Storage Lens 群組彙總的 S3 Storage Lens 指標。

例如，您也可以依物件大小或存留期範圍來篩選指標，以判斷儲存體的哪個部分包含小型物件。然後，您可以將此資訊用於 S3 Intelligent-Tiering 或 S3 生命週期，將小型物件轉換為不同的儲存類別，以進行成本和儲存最佳化。

主題

- S3 Storage Lens 群組的運作方式

- 使用 Storage Lens 群組

S3 Storage Lens 群組的運作方式

您可以使用 Storage Lens 群組，根據物件中繼資料使用自訂篩選條件彙總指標。定義自訂篩選條件時，您可以使用字首、尾碼、物件標籤、物件大小、物件存留期，或這些自訂篩選條件的組合。在 Storage Lens 群組建立期間，也可以加入單一篩選條件或多個篩選條件。若要指定多個篩選條件，請使用 `And` 或 `Or` 邏輯運算子。

當您建立並設定 Storage Lens 群組時，Storage Lens 群組本身會在連接群組的儀表板中當作自訂篩選條件。然後，您可以在儀表板中使用 Storage Lens 群組篩選條件，根據您在群組中定義的自訂篩選條件取得儲存指標。

若要在 S3 Storage Lens 儀表板中檢視 Storage Lens 群組的資料，您必須在建立群組後將該群組連接至儀表板。將 Storage Lens 群組連接至 Storage Lens 儀表板後，儀表板將在 48 小時內開始收集儲存用量指標。然後，您可以在 Storage Lens 儀表板中視覺化這些資料，或透過指標匯出將資料匯出。如果您忘記將 Storage Lens 群組連接至儀表板，您的 Storage Lens 群組資料就不會擷取或顯示在任何位置。

> **Note**
>
> - 在建立 S3 Storage Lens 群組時，您正在建立 AWS 資源。因此，每個 Storage Lens 群組都有自己的 Amazon 資源名稱 (ARN)，您可以在將其連接至 S3 Storage Lens 儀表板或從中加以排除時指定該名稱。
> - 如果您的 Storage Lens 群組未連接至儀表板，您在建立 Storage Lens 群組時將不會產生任何額外費用。
> - S3 Storage Lens 會彙總一個物件在所有相符 Storage Lens 群組下的用量指標量。因此，如果某個物件符合兩個或更多 Storage Lens 群組的篩選條件，您將會看到相同物件在儲存用量中有重複的計數。

您可以在指定主要區域中的帳戶層級建立 Storage Lens 群組 (從支援的 AWS 區域 清單)。然後，您可以將 Storage Lens 群組連接至多個 Storage Lens 儀表板，只要儀表板位於相同的 AWS 帳戶 和主要區域即可。在 AWS 帳戶 中，每個主要區域最多可以建立 50 個 Storage Lens 群組。

您可以使用 Amazon S3 主控台、AWS Command Line Interface (AWS CLI)、AWS SDK 或 Amazon S3 REST API，建立和管理 S3 Storage Lens 群組。

主題

檢視 Storage Lens 群組彙總指標

您可以將群組連接至儀表板，以檢視 Storage Lens 群組的彙總指標。您要連接的 Storage Lens 群組必須位於儀表板帳戶中指定的主要區域內。

若要將 Storage Lens 群組連接至儀表板，您必須在儀表板組態的 Storage Lens 群組彙總區段中指定群組。如果您有數個 Storage Lens 群組，您可以篩選 Storage Lens 群組彙總結果，讓其僅包含或排除您要的群組。如需關於將群組連接至儀表板的詳細資訊，請參閱 the section called “安裝或移除 Storage Lens 群組”。

連接群組後，您會在 48 小時內，在儀表板中看到其他 Storage Lens 群組彙總資料。

> **Note**
>
> 若要檢視 Storage Lens 群組的彙總指標，您必須將該群組連接至 S3 Storage Lens 儀表板。

Storage Lens 群組許可

Storage Lens 需要 AWS Identity and Access Management (IAM) 中的特定許可，才能授予對 S3 Storage Lens 群組動作的存取權。若要授予這些許可，您可以使用身分型 IAM 政策。您可以將此政策連接至 IAM 使用者、群組或角色，為他們授予許可。這類許可可能包含建立或刪除 Storage Lens 群組、檢視其組態或管理其標籤的功能。

由您授與許可的 IAM 使用者或角色必須屬於建立或擁有 Storage Lens 群組的帳戶。

若要使用 Storage Lens 群組及檢視您的 Storage Lens 群組指標，您必須先具備使用 S3 Storage Lens 的適當許可。如需詳細資訊，請參閱 the section called “S3 Storage Lens 許可”。

若要建立和管理 S3 Storage Lens 群組，您必須具有下列 IAM 許可，具體取決於您要執行的動作：

動作	IAM 許可
建立新的 Storage Lens 群組	`s3:CreateStorageLensGroup`
使用標籤建立新的 Storage Lens 群組	`s3:CreateStorageLensGroup` ， `s3:TagResource`
更新現有的 Storage Lens 群組	`s3:UpdateStorageLensGroup`
傳回 Storage Lens 群組組態的詳細資訊	`s3:GetStorageLensGroup`
列出您主要區域中所有的 Storage Lens 群組	`s3:ListStorageLensGroups`
刪除 Storage Lens 群組	`s3:DeleteStorageLensGroup`
列出新增至 Storage Lens 群組的標籤	`s3:ListTagsForResource`
新增或更新現有 Storage Lens 群組的 Storage Lens 群組標籤	`s3:TagResource`
從 Storage Lens 群組中刪除標籤	`s3:UntagResource`

以下是如何在建立 Storage Lens 群組的帳戶中設定 IAM 政策的範例。若要使用此原則，請將 *us-east-1* 取代為 Storage Lens 群組所在的主要區域。將 *111122223333* 取代為您的 AWS 帳戶 ID，並將 *example-storage-lens-group* 取代為您的 Storage Lens 群組名稱。若要將這些許可套用至所有 Storage Lens 群組，請將 *example-storage-lens-group* 取代為 *。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EXAMPLE-Statement-ID",
            "Effect": "Allow",
            "Action": [
                "s3:CreateStorageLensGroup",
                "s3:UpdateStorageLensGroup",
                "s3:GetStorageLensGroup",
                "s3:ListStorageLensGroups",
                "s3:DeleteStorageLensGroup,
                "s3:TagResource",
```

```
                "s3:UntagResource",
                "s3:ListTagsForResource"
                ],
            "Resource": "arn:aws:s3:us-east-1:111122223333:storage-lens-group/example-storage-lens-group"
        }
    ]
}
```

如需 S3 Storage Lens 許可的詳細資訊，請參閱 Amazon S3 Storage Lens 許可。如需 IAM 政策語言的詳細資訊，請參閱 Amazon S3 中的政策和許可。

Storage Lens 群組組態

S3 Storage Lens 群組名稱

建議您為 Storage Lens 群組指定描述其用途的名稱，以便您判斷應將哪些群組連接至儀表板。若要將 Storage Lens 群組連接至儀表板，您必須在儀表板組態的 Storage Lens 群組彙總區段中指定群組。

Storage Lens 群組名稱在帳戶內必須是唯一的。其長度不可超過 64 個字元，且只能包含英文字母 (a-z、A-Z)、數字 (0-9)、連字號 (-) 和底線 (_)。

主要區域

主要區域是您的 Storage Lens 群組建立和維護所在的 AWS 區域。您的 Storage Lens 群組會建立在與 Amazon S3 Storage Lens 儀表板相同的主要區域。Storage Lens 群組組態和指標也會儲存在此區域中。您最多可在主要區域中建立 50 個 Storage Lens 群組。

建立 Storage Lens 群組之後，您就無法編輯主要區域。

範圍

若要將物件包含在 Storage Lens 群組中，物件必須在 Amazon S3 Storage Lens 儀表板的範圍內。Storage Lens 儀表板的範圍取決於您在 S3 Storage Lens 儀表板組態的儀表板範圍中包含的儲存貯體。

您可以對物件使用不同的篩選條件，以定義 Storage Lens 群組的範圍。若要在 S3 Storage Lens 儀表板中檢視這些 Storage Lens 群組指標，物件必須與您包含在 Storage Lens 群組中的篩選條件相符。例如，假設您的 Storage Lens 群組包含具有字首 `marketing` 和尾碼 `.png` 的物件，但沒有符合這些條件的物件。在此情況下，此 Storage Lens 群組的指標將不會產生在您的每日指標匯出中，且儀表板中不會顯示此群組的指標。

篩選條件

您可以在 S3 Storage Lens 群組中使用下列篩選條件：

- 字首 – 指定所含物件的字首，即物件索引鍵名稱開頭處的字元字串。例如，字首篩選條件的值 `images` 包含具有下列任一字首的物件：`images/`、`images-marketing` 和 `images/production`。字首的最大長度為 1,024 位元組。
- 尾碼 – 指定所含物件的尾碼 (例如 `.png`、`.jpeg` 或 `.csv`)。尾碼的最大長度為 1,024 位元組。
- 物件標籤 – 指定要作為篩選依據的物件標籤清單。標籤索引鍵不可超過 128 個 Unicode 字元，且標籤值不可超過 256 個 Unicode 字元。請注意，如果物件標籤值欄位保留空白，S3 Storage Lens 群組只會將該物件與也具有空白標籤值的其他物件進行比對。
- 存留期 – 指定所含物件的物件存留期範圍 (以天為單位)。僅支援整數。
- 大小 – 指定所含物件的物件大小範圍 (以位元組為單位)。僅支援整數。允許的值上限為 5 TB。

Storage Lens 群組物件標籤

您可以建立最多包含 10 個物件標籤篩選條件的 Storage Lens 群組。下列範例包含兩個物件標籤鍵/值對，作為名為 *Marketing-Department* 的 Storage Lens 群組的篩選條件。若要使用此範例，請將 *Marketing-Department* 取代為您的群組名稱，並將 *object-tag-key-1*、*object-tag-value-1* 等項目取代為要作為篩選依據的物件標籤鍵/值對。

```
{
    "Name": "Marketing-Department",
    "Filter": {
     "MatchAnyTag":[
                {
                    "Key": "object-tag-key-1",
                    "Value": "object-tag-value-1"
                },
                {
                    "Key": "object-tag-key-2",
                    "Value": "object-tag-value-2"
                }
            ]
    }
}
```

邏輯運算子 (And 或 Or)

若要在 Storage Lens 群組中包含多個篩選條件，您可以使用邏輯運算子 (And 或 Or)。在下列範例中，名為 *Marketing-Department* 的 Storage Lens 群組具有包含 And、Prefix 和 ObjectAge 篩選條件的運算子。由於使用了 And 運算子，因此只有符合前述所有篩選條件的物件，才會包含在 Storage Lens 群組的範圍內。

若要使用此範例，請將 *user input placeholders* 取代為要作為篩選依據的值。

```
{
    "Name": "Marketing-Department",
    "Filter": {
        "And": {
            "MatchAnyPrefix": [
                "prefix-1",
                "prefix-2",
                "prefix-3/sub-prefix-1"
            ],
             "MatchObjectAge": {
                "DaysGreaterThan": 10,
                "DaysLessThan": 60
            },
            "MatchObjectSize": {
                "BytesGreaterThan": 10,
                "BytesLessThan": 60
            }
        }
    }
}
```

> **Note**
>
> 如果您要在篩選條件中包含符合任何條件的物件，請將此範例中的 And 邏輯運算子取代為 Or 邏輯運算子。

AWS 資源標籤

每個 S3 Storage Lens 群組都會計為一個具有其本身 Amazon Resource Name (ARN) 的 AWS 資源。因此，當您設定 Storage Lens 群組時，您可以選擇性地將 AWS 資源標籤新增至群組。每個

Storage Lens 群組最多可新增 50 個標籤。若要建立具有標籤的 Storage Lens 群組，您必須擁有 `s3:CreateStorageLensGroup` 和 `s3:TagResource` 許可。

您可以使用 AWS 資源標籤，根據部門、業務單位或專案來分類資源。此做法在您擁有許多相同類型的資源時很有用。藉由套用標籤，您可以根據先前指派的標籤，快速識別特定的 Storage Lens 群組。您也可以使用標籤來追蹤和配置成本。

此外，當您將 AWS 資源標籤新增至 Storage Lens 群組時，將會啟用屬性型存取控制 (ABAC)。ABAC 是一種根據屬性 (在此案例中為標籤) 定義許可的授權策略。您也可以在 IAM 政策中使用指定資源標籤的條件，來控制對 AWS 資源的存取。

您可以編輯標籤金鑰和值，並且可以隨時從資源移除標籤。此外，請留意下列限制：

- 標籤索引鍵與標籤值皆區分大小寫。
- 若您將與現有標籤具有相同鍵的標籤新增到該資源，則新值會覆寫舊值。
- 如果您刪除資源，也會刪除任何該資源的標籤。
- 請勿在 AWS 資源標籤中包含私人或敏感資料。
- 不支援系統標籤 (或具有以 `aws:` 開頭的標籤索引鍵的標籤)。
- 每個標籤索引鍵的長度不可超過 128 個字元。每個標籤值的長度不可超過 256 個字元。

Storage Lens 群組指標匯出

S3 Storage Lens 群組指標會包含在 Storage Lens 群組連接到的儀表板的 Amazon S3 Storage Lens 指標匯出中。如需關於 Storage Lens 指標匯出功能的一般資訊，請參閱 使用資料匯出檢視 Amazon S3 Storage Lens 指標。

Storage Lens 群組的指標匯出包含您的 Storage Lens 群組連接到的儀表板範圍內的任何 S3 Storage Lens 指標。匯出也包含 Storage Lens 群組的其他指標資料。

建立 Storage Lens 群組後，您的指標匯出會每天傳送至您為群組連接到的儀表板設定指標匯出時選取的儲存貯體。最多可能需要 48 小時，才會收到第一次指標匯出。

若要產生每日匯出的指標，物件必須符合您包含在 Storage Lens 群組中的篩選條件。如果沒有任何物件符合您在 Storage Lens 群組中包含的篩選條件，則不會產生任何指標。不過，如果有物件符合兩個或更多 Storage Lens 群組，當該物件出現在指標匯出中時，將會個別就每個群組列出該物件。

您可以在儀表板的指標匯出中，尋找 `record_type` 欄中的下列其中一個值，以識別 Storage Lens 群組的指標：

- STORAGE_LENS_GROUP_BUCKET
- STORAGE_LENS_GROUP_ACCOUNT

record_value 欄會顯示 Storage Lens 群組的資源 ARN (例如 arn:aws:s3:*us-east-1*:*111122223333*:storage-lens-group/*Marketing-Department*)。

使用 Storage Lens 群組

Amazon S3 Storage Lens 群組會根據物件中繼資料使用自訂篩選條件彙總指標。您可以使用字首、尾碼、物件標籤、物件大小或物件存留期來分析及篩選 S3 Storage Lens 指標。透過 Amazon S3 Storage Lens 群組，您也可以分類個別和不同 Amazon S3 儲存貯體的用量。因此，您將能夠進一步了解和最佳化 S3 儲存。

若要開始視覺化 Storage Lens 群組的資料，您必須先將 Storage Lens 群組連接至 S3 Storage Lens 儀表板。如果您需要在儀表板中管理 Storage Lens 群組，可以編輯儀表板組態。若要確認帳戶下有哪些 Storage Lens 群組，您可以將其列出。若要確認有哪些 Storage Lens 群組連接至儀表板，您可以隨時查看儀表板中的 Storage Lens 群組索引標籤。若要檢閱或更新現有 Storage Lens 群組的範圍，您可以檢視其詳細資訊。您也可以永久刪除 Storage Lens 群組。

若要管理許可，您可以建立使用者定義的 AWS 資源標籤，並將其新增至 Storage Lens 群組。您可以使用 AWS 資源標籤，根據部門、業務單位或專案來分類資源。此做法在您擁有許多相同類型的資源時很有用。藉由套用標籤，您可以根據先前指派的標籤，快速識別特定的 Storage Lens 群組。

此外，當您將 AWS 資源標籤新增至 Storage Lens 群組時，將會啟用屬性型存取控制 (ABAC)。ABAC 是一種根據屬性 (在此案例中為標籤) 定義許可的授權策略。您也可以在 IAM 政策中使用指定資源標籤的條件，來控制對 AWS 資源的存取。

主題

- 建立 Storage Lens 群組
- 在您的儀表板上連接或移除 S3 Storage Lens 群組
- 將您的 Storage Lens 群組資料視覺化
- 更新 Storage Lens 群組
- 使用 Storage Lens 群組管理 AWS 資源標籤
- 列出所有 Storage Lens 群組
- 檢視 Storage Lens 群組詳細資訊
- 刪除 Storage Lens 群組

建立 Storage Lens 群組

下列範例示範如何使用 Amazon S3 主控台 AWS Command Line Interface (AWS CLI) 和建立 Amazon S3 儲存鏡頭群組 AWS SDK for Java。

使用 S3 主控台

建立 Storage Lens 群組

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在頁面頂端的導覽列中，選擇目前顯示的 AWS 區域名稱。接下來，選擇您要切換到的區域。
3. 在左側導覽窗格中，選擇 Storage Lens 群組。
4. 選擇建立 Storage Lens 群組。
5. 在「一般」下方，檢視您的「居住地區」並輸入儲存鏡頭群組名稱。
6. 在範圍底下，選擇要套用至 Storage Lens 群組的篩選條件。若要套用多個篩選條件，請選擇篩選條件，然後選擇 AND 或 OR 邏輯運算子。
   - 針對字首篩選條件，選擇字首，然後輸入字首字串。若要新增多個字首，請選擇新增字首。若要移除字首，請選擇要移除的字首旁的移除。
   - 在物件標籤篩選條件中，選擇物件標籤，然後輸入物件的鍵/值對。然後，選擇新增標籤。若要移除標籤，請選擇要移除的標籤旁的移除。
   - 針對尾碼篩選條件，選擇尾碼，然後輸入尾碼字串。若要新增多個尾碼，請選擇新增尾碼。若要移除尾碼，請選擇要移除的尾碼旁的移除。
   - 針對存留期篩選條件，指定物件存留期範圍 (天數)。選擇指定最短物件存留期，然後輸入最短的物件存留期。然後，選擇指定最長物件存留期，並輸入最長的物件存留期。
   - 針對大小篩選條件，指定物件大小範圍和測量單位。選擇指定最小物件大小，然後輸入最小的物件大小。選擇指定最大物件大小，並輸入最大的物件大小。
7. (選擇性) 對於 AWS 資源標籤，請新增鍵值配對，然後選擇 [新增標籤]。
8. 選擇建立 Storage Lens 群組。

使用 AWS CLI

下列範例 AWS CLI 指令會建立儲存鏡頭群組。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control create-storage-lens-group --account-id 111122223333 \
```

```
--region us-east-1 --storage-lens-group=file://./marketing-department.json
```

下列範例 AWS CLI 指令會建立具有兩個 AWS 資源標籤的儲存鏡頭群組。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control create-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json \
--tags Key=k1,Value=v1 Key=k2,Value=v2
```

如需範例 JSON 組態，請參閱 Storage Lens 群組組態。

使用適用於 Java 的 AWS 開發套件

下列 AWS SDK for Java 範例會建立儲存鏡頭群組。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

Example – 使用單一篩選條件建立 Storage Lens 群組

下列範例會建立名為 *Marketing-Department* 的 Storage Lens 群組。此群組具有會將存留期範圍指定為 *30* 到 *90* 天的物件存留期篩選條件。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectAge;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;

public class CreateStorageLensGroupWithObjectAge {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            StorageLensGroupFilter objectAgeFilter = StorageLensGroupFilter.builder()
                    .matchObjectAge(MatchObjectAge.builder()
                            .daysGreaterThan(30)
```

```
                            .daysLessThan(90)
                            .build())
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(objectAgeFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example – 使用包含多個篩選條件的 **AND** 運算子建立 Storage Lens 群組

下列範例會建立名為 *Marketing-Department* 的 Storage Lens 群組。此群組會使用 AND 運算子指出物件必須符合所有篩選條件。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectAge;
import software.amazon.awssdk.services.s3control.model.MatchObjectSize;
import software.amazon.awssdk.services.s3control.model.S3Tag;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupAndOperator;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;


public class CreateStorageLensGroupWithAndFilter {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create object tags.
            S3Tag tag1 = S3Tag.builder()
                    .key("object-tag-key-1")
                    .value("object-tag-value-1")
                    .build();
            S3Tag tag2 = S3Tag.builder()
                    .key("object-tag-key-2")
                    .value("object-tag-value-2")
                    .build();

            StorageLensGroupAndOperator andOperator =
 StorageLensGroupAndOperator.builder()
                    .matchAnyPrefix("prefix-1", "prefix-2", "prefix-3/sub-prefix-1")
                    .matchAnySuffix(".png", ".gif", ".jpg")
                    .matchAnyTag(tag1, tag2)
                    .matchObjectAge(MatchObjectAge.builder()
                            .daysGreaterThan(30)
                            .daysLessThan(90).build())
                    .matchObjectSize(MatchObjectSize.builder()
                            .bytesGreaterThan(1000L)
                            .bytesLessThan(6000L).build())
                    .build();

            StorageLensGroupFilter andFilter = StorageLensGroupFilter.builder()
                    .and(andOperator)
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
```

```
                    .name(storageLensGroupName)
                    .filter(andFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example – 使用包含多個篩選條件的 **OR** 運算子建立 Storage Lens 群組

下列範例會建立名為 *Marketing-Department* 的 Storage Lens 群組。此群組會使用 OR 運算子來套用字首篩選條件 (*prefix-1*、*prefix-2*、*prefix3/sub-prefix-1*) 或物件大小篩選條件，其大小範圍介於 *1000* 位元組與 *6000* 位元組之間。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectSize;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
```

```
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupOrOperator;

public class CreateStorageLensGroupWithOrFilter {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            StorageLensGroupOrOperator orOperator =
 StorageLensGroupOrOperator.builder()
                    .matchAnyPrefix("prefix-1", "prefix-2", "prefix-3/sub-prefix-1")
                    .matchObjectSize(MatchObjectSize.builder()
                            .bytesGreaterThan(1000L)
                            .bytesLessThan(6000L)
                            .build())
                    .build();

            StorageLensGroupFilter orFilter = StorageLensGroupFilter.builder()
                    .or(orOperator)
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(orFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
```

```
            e.printStackTrace();
        }
    }
}
```

Example — 使用單個濾鏡和兩個 AWS 資源標籤創建存儲鏡頭組

下列範例會建立名為 *Marketing-Department*、具有尾碼篩選條件的 Storage Lens 群組。此範例也會將兩個 AWS 資源標籤新增至儲存鏡頭群組。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.Tag;

public class CreateStorageLensGroupWithResourceTags {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create AWS resource tags.
            Tag resourceTag1 = Tag.builder()
                    .key("resource-tag-key-1")
                    .value("resource-tag-value-1")
                    .build();
            Tag resourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-value-2")
                    .build();

            StorageLensGroupFilter suffixFilter = StorageLensGroupFilter.builder()
                    .matchAnySuffix(".png", ".gif", ".jpg")
                    .build();
```

```
            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(suffixFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .tags(resourceTag1, resourceTag2)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

如需範例 JSON 組態，請參閱 Storage Lens 群組組態。

在您的儀表板上連接或移除 S3 Storage Lens 群組

在 Amazon S3 Storage Lens 中升級至進階方案後，您可以將 Storage Lens 群組連接至儀表板。如果您有數個 Storage Lens 群組，您可以包含或排除所需的群組。

您的 Storage Lens 群組必須位於儀表板帳戶中指定的主要區域內。將 Storage Lens 群組連接至儀表板後，您會在 48 小時內，在指標匯出中收到其他 Storage Lens 群組彙總資料。

> **Note**
>
> 如果您想要檢視 Storage Lens 群組的彙總指標，您必須將其連接至 Storage Lens 儀表板。如需 Storage Lens 群組 JSON 組態檔案的範例，請參閱 使用 JSON Storage Lens 群組的 S3 Storage Lens 範例組態。

將 Storage Lens 群組連接至 S3 Storage Lens 儀表板

將 Storage Lens 群組連接至 Storage Lens 儀表板

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格的 Storage Lens 下，選擇儀表板。
3. 針對 Storage Lens 群組要連接到的 Storage Lens 儀表板，選擇其選項按鈕。
4. 選擇 編輯 。
5. 在 Metrics selection (指標選擇) 下，選擇 Advanced metrics and recommendations (進階指標和建議)。
6. 選取 Storage Lens 群組彙總。

   > **Note**
   >
   > 依預設，也會選取進階指標。不過，您也可以取消選取此設定，因為不一定要彙總 Storage Lens 群組資料。

7. 向下捲動至 Storage Lens 群組彙總，並指定要在資料彙總中包含或排除的一或多個 Storage Lens 群組。您可以使用下列篩選選項：

   - 如果您要包含某些 Storage Lens 群組，請選擇包含 Storage Lens 群組。在要包含的 Storage Lens 群組底下，選取您的 Storage Lens 群組。
   - 如果您想要包含所有的 Storage Lens 群組，請選取包含此帳戶的主要區域中所有的 Storage Lens 群組。
   - 如果您要排除某些 Storage Lens 群組，請選擇排除 Storage Lens 群組。在要排除的 Storage Lens 群組底下，選取您要排除的 Storage Lens 群組。

8. 選擇 Save changes (儲存變更)。如果您已正確設定 Storage Lens 群組，您將在 48 小時內，在儀表板中看到其他 Storage Lens 群組彙總資料。

從 S3 Storage Lens 儀表板中移除 Storage Lens 群組

從 S3 Storage Lens 儀表板中移除 Storage Lens 群組

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左導覽窗格的 Storage Lens 下，選擇儀表板。
3. 針對要從 Storage Lens 群組中移除的 Storage Lens 儀表板，選擇其選項按鈕。
4. 選擇檢視儀表板組態。
5. 選擇 編輯 。
6. 向下捲動至指標選取區段。
7. 在 Storage Lens 群組彙總底下，選擇您要移除的 Storage Lens 群組旁的 X。這會移除您的 Storage Lens 群組。

   如果您將所有的 Storage Lens 群組包含在儀表板中，請清除包含此帳戶的主要區域中所有的 Storage Lens 群組旁的核取方塊。
8. 選擇 Save changes (儲存變更)。

> **Note**
>
> 儀表板最多可能需要 48 小時才會反映組態更新。

將您的 Storage Lens 群組資料視覺化

您可以將群組連接至 Amazon S3 Storage Lens 儀表板，以將您的 Storage Lens 群組資料視覺化。將 Storage Lens 群組彙總中的 Storage Lens 群組納入您的儀表板組態之後，Storage Lens 群組資料最多可能需要 48 小時才會顯示在儀表板中。

儀表板組態更新後，任何新連接的 Storage Lens 群組都會出現在 Storage Lens 群組索引標籤底下的可用資源清單中。您也可以用另一個維度分割資料，以進一步分析概觀索引標籤中的儲存用量。例如，您可以選擇前 3 個類別底下列出的項目之一，然後選擇分析依據，依其他維度進行資料分割。您無法套用與篩選條件本身相同的維度。

> **Note**
>
> 您無法將 Storage Lens 群組篩選條件與字首篩選條件一起套用，或反向操作。您也無法使用字首篩選條件進一步分析 Storage Lens 群組。

您可以使用 Amazon S3 Storage Lens 儀表板中的 Storage Lens 群組索引標籤，為連接至儀表板的 Storage Lens 群組自訂資料視覺效果。您可以視覺化連接至儀表板的部分或所有 Storage Lens 群組的資料。

視覺化 S3 Storage Lens 儀表板中的 Storage Lens 群組資料時，請注意下列事項：

- S3 Storage Lens 會彙總一個物件在所有相符 Storage Lens 群組下的用量指標量。因此，如果某個物件符合兩個或更多 Storage Lens 群組的篩選條件，您將會看到相同物件在儲存用量中有重複的計數。
- 物件必須與您包含在 Storage Lens 群組中的篩選條件相符。如果沒有任何物件符合您在 Storage Lens 群組中包含的篩選條件，則不會產生任何指標。若要判斷是否有任何未指派的物件，請在帳戶層級與儲存貯體層級檢查儀表板中的物件總數。

更新 Storage Lens 群組

下列範例示範如何更新 Amazon S3 Storage Lens 群組。您可以使用 Amazon S3 主控台、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 來更新 Storage Lens 群組。

使用 S3 主控台

更新 Storage Lens 群組

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Storage Lens 群組。
3. 在 Storage Lens 群組底下，選擇您要更新的 Storage Lens 群組。
4. 在範圍底下，選擇編輯。
5. 在範圍頁面上，選取要套用至 Storage Lens 群組的篩選條件。若要套用多個篩選條件，請選取篩選條件，然後選擇 AND 或 OR 邏輯運算子。
   - 針對字首篩選條件，選取字首，然後輸入字首字串。若要新增多個字首，請選擇新增字首。若要移除字首，請選擇要移除的字首旁的移除。

- 針對物件標籤篩選條件，輸入物件的鍵/值對。然後，選擇新增標籤。若要移除標籤，請選擇要移除的標籤旁的移除。
- 針對尾碼篩選條件，選取尾碼，然後輸入尾碼字串。若要新增多個尾碼，請選擇新增尾碼。若要移除尾碼，請選擇要移除的尾碼旁的移除。
- 針對存留期篩選條件，指定物件存留期範圍 (天數)。選擇指定最短物件存留期，然後輸入最短的物件存留期。針對指定最長物件存留期，輸入最長的物件存留期。
- 針對大小篩選條件，指定物件大小範圍和測量單位。選擇指定最小物件大小，然後輸入最小的物件大小。針對指定最大物件大小，輸入最大的物件大小。

6. 選擇 Save changes (儲存變更)。Storage Lens 群組的詳細資訊頁面隨即出現。
7. (選用) 如果您要新增 AWS 資源標籤，請捲動至 AWS 資源標籤區段，然後選擇新增標籤。Add tags (新增標籤) 頁面隨即出現。

   新增新的鍵/值對，然後選擇儲存變更。Storage Lens 群組的詳細資訊頁面隨即出現。
8. (選用) 如果要移除現有的 AWS 資源標籤，請捲動至 AWS 資源標籤區段，並選取資源標籤。再選擇 Delete (刪除)。刪除 AWS 標籤對話方塊隨即出現。

   再次選擇刪除，可永久刪除 AWS 資源標籤。

   > **Note**
   >
   > AWS 資源標籤永久刪除後，即無法還原。

使用 AWS CLI

下列 AWS CLI 範例命令會傳回名為 *marketing-department* 的 Storage Lens 群組的組態詳細資訊。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control get-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

下列 AWS CLI 範例會更新 Storage Lens 群組。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control update-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json
```

如需範例 JSON 組態，請參閱 Storage Lens 群組組態。

使用適用於 Java 的 AWS SDK

下列 AWS SDK for Java 範例會傳回帳戶 *111122223333* 中 *Marketing-Department* Storage Lens 群組的組態詳細資訊。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupResponse;

public class GetStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            GetStorageLensGroupRequest getRequest =
 GetStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            GetStorageLensGroupResponse response =
 s3ControlClient.getStorageLensGroup(getRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
```

```
        }
    }
}
```

下列範例會更新帳戶 *111122223333* 中名為 *Marketing-Department* 的 Storage Lens 群組。此範例會更新儀表板範圍，以包含符合下列任何尾碼的物件：*.png*、*.gif*、*.jpg* 或 *.jpeg*。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.UpdateStorageLensGroupRequest;

public class UpdateStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create updated filter.
            StorageLensGroupFilter suffixFilter = StorageLensGroupFilter.builder()
                    .matchAnySuffix(".png", ".gif", ".jpg", ".jpeg")
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(suffixFilter)
                    .build();

            UpdateStorageLensGroupRequest updateStorageLensGroupRequest =
 UpdateStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId)
                    .build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
```

```
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.updateStorageLensGroup(updateStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

如需範例 JSON 組態，請參閱 Storage Lens 群組組態。

使用 Storage Lens 群組管理 AWS 資源標籤

每個 Amazon S3 Storage Lens 群組都會計為一個具有其本身 Amazon Resource Name (ARN) 的 AWS 資源。因此，當您設定 Storage Lens 群組時，您可以選擇性地將 AWS 資源標籤新增至群組。每個 Storage Lens 群組最多可新增 50 個標籤。若要建立具有標籤的 Storage Lens 群組，您必須擁有 `s3:CreateStorageLensGroup` 和 `s3:TagResource` 許可。

您可以使用 AWS 資源標籤，根據部門、業務單位或專案來分類資源。此做法在您擁有許多相同類型的資源時很有用。藉由套用標籤，您可以根據先前指派的標籤，快速識別特定的 Storage Lens 群組。您也可以使用標籤來追蹤和配置成本。

此外，當您將 AWS 資源標籤新增至 Storage Lens 群組時，將會啟用屬性型存取控制 (ABAC)。ABAC 是一種根據屬性 (在此案例中為標籤) 定義許可的授權策略。您也可以在 IAM 政策中使用指定資源標籤的條件，來控制對 AWS 資源的存取。

您可以編輯標籤金鑰和值，並且可以隨時從資源移除標籤。此外，請留意下列限制：

- 標籤索引鍵與標籤值皆區分大小寫。
- 若您將與現有標籤具有相同鍵的標籤新增到該資源，則新值會覆寫舊值。
- 如果您刪除資源，也會刪除任何該資源的標籤。
- 請勿在 AWS 資源標籤中包含私人或敏感資料。
- 不支援系統標籤 (具有以 `aws:` 開頭的標籤索引鍵)。
- 每個標籤索引鍵的長度不可超過 128 個字元。每個標籤值的長度不可超過 256 個字元。

下列範例示範如何將 AWS 資源標籤與 Storage Lens 群組搭配使用。

主題

將 AWS 資源標籤新增至 Storage Lens 群組

下列範例示範如何將 AWS 資源標籤新增至 Amazon S3 Storage Lens 群組。您可以使用 Amazon S3 主控台、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 新增資源標籤。

使用 S3 主控台

將 AWS 資源標籤新增至 Storage Lens 群組

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Storage Lens 群組。
3. 在 Storage Lens 群組底下，選擇您要更新的 Storage Lens 群組。
4. 在 AWS 資源標籤底下，選擇新增標籤。
5. 在新增標籤頁面上，新增新的鍵/值對。

   > **Note**
   >
   > 新增與現有標籤具有相同的索引鍵的標籤，將會覆寫先前的標籤值。

6. (選用) 若要新增多個標籤，請再次選擇新增標籤以繼續新增項目。您最多可以為 Storage Lens 群組新增最多 50 個 AWS 資源標籤。
7. (選用) 如果您要移除新增的項目，請在移除的標籤旁選擇移除。
8. 選擇 Save changes (儲存變更)。

使用 AWS CLI

下列範例 AWS CLI 命令會將兩個資源標籤新增至名為 *marketing-department* 的現有 Storage Lens 群組。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control tag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1 --tags Key=k1,Value=v1 Key=k2,Value=v2
```

使用適用於 Java 的 AWS SDK

下列 AWS SDK for Java 範例會將兩個 AWS 資源標籤新增至現有的 Storage Lens 群組。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;

public class TagResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            Tag resourceTag1 = Tag.builder()
                .key("resource-tag-key-1")
                .value("resource-tag-value-1")
                .build();
            Tag resourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-value-2")
                    .build();
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tags(resourceTag1, resourceTag2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
```

```
            s3ControlClient.tagResource(tagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

更新 Storage Lens 群組標籤值

下列範例示範如何使用 Amazon S3 主控台、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 更新 Storage Lens 群組標籤值。

使用 S3 主控台

更新 Storage Lens 群組的 AWS 資源標籤

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Storage Lens 群組。
3. 在 Storage Lens 群組底下，選擇您要更新的 Storage Lens 群組。
4. 在 AWS 資源標籤底下，選取您要更新的標籤。
5. 使用與您要更新的鍵/值對相同的索引鍵，新增新的標籤值。選擇更新標籤值的核取記號圖示。

   > **Note**
   >
   > 新增與現有標籤具有相同的索引鍵的標籤，將會覆寫先前的標籤值。

6. (選用) 如果您要新增標籤，請選擇新增標籤以新增項目。Add tags (新增標籤) 頁面隨即出現。

   您最多可以為 Storage Lens 群組新增最多 50 個 AWS 資源標籤。當您完成新增標籤的作業時，請選擇儲存變更。

7. (選用) 如果您要移除新增的項目，請在移除的標籤旁選擇移除。當您完成移除標籤的作業時，請選擇儲存變更。

使用 AWS CLI

下列範例 AWS CLI 指令會更新名為 *marketing-department* 的 Storage Lens 群組的兩個標籤值。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control tag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1 --tags Key=k1,Value=v3 Key=k2,Value=v4
```

使用適用於 Java 的 AWS SDK

下列 AWS SDK for Java 範例會更新兩個 Storage Lens 群組標籤值。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;

public class UpdateTagsForResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            Tag updatedResourceTag1 = Tag.builder()
                .key("resource-tag-key-1")
                .value("resource-tag-updated-value-1")
                .build();
            Tag updatedResourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-updated-value-2")
                    .build();
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tags(updatedResourceTag1, updatedResourceTag2)
                    .accountId(accountId)
```

```
                .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                .region(Region.US_WEST_2)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();
            s3ControlClient.tagResource(tagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

從 Storage Lens 群組中刪除 AWS 資源標籤

下列範例示範如何從 Storage Lens 群組中刪除 AWS 資源標籤。您可以使用 Amazon S3 主控台、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 來刪除標籤。

使用 S3 主控台

從 Storage Lens 群組中刪除 AWS 資源標籤

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Storage Lens 群組。
3. 在 Storage Lens 群組底下，選擇您要更新的 Storage Lens 群組。
4. 在 AWS 資源標籤底下，選取您要刪除的鍵/值對。
5. 選擇 Delete (刪除)。刪除 AWS 資源標籤對話方塊隨即出現。

> **Note**
>
> 如果標籤用來控制存取，繼續執行此動作可能會影響到相關資源。標籤永久刪除後，即無法還原。

6. 選擇刪除，永久刪除鍵/值對。

使用 AWS CLI

下列 AWS CLI 命令會從現有的 Storage Lens 群組中刪除兩個 AWS 資源標籤：若要使用此範例命令，請將 *user input placeholders* 取代為您自己的資訊。

```
aws s3control untag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/Marketing-
Department \
--region us-east-1 --tag-keys k1 k2
```

使用適用於 Java 的 AWS SDK

下列 AWS SDK for Java 範例會從您在帳戶 *111122223333* 中指定的 Storage Lens 群組 Amazon 資源名稱 (ARN) 刪除兩個 AWS 資源標籤。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.UntagResourceRequest;

public class UntagResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            String tagKey1 = "resource-tag-key-1";
            String tagKey2 = "resource-tag-key-2";
            UntagResourceRequest untagResourceRequest = UntagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tagKeys(tagKey1, tagKey2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
```

```
            s3ControlClient.untagResource(untagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

列出 Storage Lens 群組標籤

下列範例示範如何列出與 Storage Lens 群組相關聯的 AWS 資源標籤。您可以使用 Amazon S3 主控台、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 來列出標籤。

使用 S3 主控台

檢閱 Storage Lens 群組的標籤與標籤值清單

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Storage Lens 群組。
3. 在 Storage Lens 群組底下，選擇您感興趣的 Storage Lens 群組。
4. 向下捲動至 AWS 資源標籤區段。所有新增至 Storage Lens 群組的使用者定義 AWS 資源標籤，都會與其標籤值一起列出。

使用 AWS CLI

下列 AWS CLI 範例命令會列出名為 *marketing-department* 的 Storage Lens 群組的所有 Storage Lens 群組標籤值。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control list-tags-for-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1
```

使用適用於 Java 的 AWS SDK

下列 AWS SDK for Java 範例會列出您指定的 Storage Lens 群組 Amazon 資源名稱 (ARN) 的 Storage Lens 群組標籤值。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceRequest;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceResponse;

public class ListTagsForResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            ListTagsForResourceResponse response =
 s3ControlClient.listTagsForResource(listTagsForResourceRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
```

```
}
```

列出所有 Storage Lens 群組

下列範例示範如何列出 AWS 帳戶 和主要區域中的所有 Amazon S3 Storage Lens 群組。這些範例說明如何使用 Amazon S3 主控台、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 列出所有 Storage Lens 群組。

使用 S3 主控台

列出帳戶和主要區域中的所有 Storage Lens 群組

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Storage Lens 群組。
3. 在 Storage Lens 群組底下，會顯示您的帳戶中的 Storage Lens 群組清單。

使用 AWS CLI

下列 AWS CLI 範例會列出您的帳戶的所有 Storage Lens 群組。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control list-storage-lens-groups --account-id 111122223333 \
--region us-east-1
```

使用適用於 Java 的 AWS SDK

下列 AWS SDK for Java 範例會列出帳戶 *111122223333* 的 Storage Lens 群組。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListStorageLensGroupsRequest;
import software.amazon.awssdk.services.s3control.model.ListStorageLensGroupsResponse;

public class ListStorageLensGroups {
```

```
    public static void main(String[] args) {
        String accountId = "111122223333";

        try {
            ListStorageLensGroupsRequest listStorageLensGroupsRequest =
 ListStorageLensGroupsRequest.builder()
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            ListStorageLensGroupsResponse response =
 s3ControlClient.listStorageLensGroups(listStorageLensGroupsRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

檢視 Storage Lens 群組詳細資訊

下列範例示範如何檢視 Amazon S3 Storage Lens 群組組態詳細資訊。您可以使用 Amazon S3 主控台、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 來檢視這些詳細資訊。

使用 S3 主控台

檢視 Storage Lens 群組組態詳細資訊

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Storage Lens 群組。
3. 在 Storage Lens 群組底下，選擇您感興趣的 Storage Lens 群組旁的選項按鈕。
4. 請選擇 View Details (查看詳細資訊)。您現在可以檢閱 Storage Lens 群組的詳細資訊。

使用 AWS CLI

下列 AWS CLI 範例會傳回 Storage Lens 群組的組態詳細資訊。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control get-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

使用適用於 Java 的 AWS SDK

下列 AWS SDK for Java 範例會傳回帳戶 *111122223333* 中名為 *Marketing-Department* 的 Storage Lens 群組的組態詳細資訊。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupResponse;

public class GetStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            GetStorageLensGroupRequest getRequest =
 GetStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            GetStorageLensGroupResponse response =
 s3ControlClient.getStorageLensGroup(getRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
```

```
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

刪除 Storage Lens 群組

下列範例示範如何使用 Amazon S3 主控台、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 刪除 Amazon S3 Storage Lens 群組。

使用 S3 主控台

刪除 Storage Lens 群組

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Storage Lens 群組。
3. 在 Storage Lens 群組底下，選擇您要刪除的 Storage Lens 群組旁的按鈕選項。
4. 選擇 Delete (刪除)。刪除 Storage Lens 群組對話方塊隨即顯示。
5. 再次選擇刪除，可永久刪除 Storage Lens 群組。

> **Note**
>
> Storage Lens 群組一旦刪除，就無法還原。

使用 AWS CLI

下列 AWS CLI 範例會刪除名為 *marketing-department* 的 Storage Lens 群組。若要使用此範例命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control delete-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

使用適用於 Java 的 AWS SDK

下列 AWS SDK for Java 範例會刪除帳戶 *111122223333* 中名為 *Marketing-Department* 的 Storage Lens 群組。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.DeleteStorageLensGroupRequest;

public class DeleteStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            DeleteStorageLensGroupRequest deleteStorageLensGroupRequest =
 DeleteStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.deleteStorageLensGroup(deleteStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用 AWS X-Ray 追蹤 Amazon S3 請求

AWS X-Ray 會收集應用程式提供的請求相關資料。然後，您可以檢視和篩選資料，以識別並疑難排解分散式應用程式和微型服務架構中的效能問題和錯誤。對於對任何應用程式的追蹤請求，X-Ray 會顯示有關應用程式對下游 AWS 資源、微型服務、資料庫和 HTTP Web API 所進行的請求、回應和呼叫等詳細資訊。

如需詳細資訊，請參閱《AWS X-Ray 開發人員指南》中的什麼是 AWS X-Ray？章節。

主題

X-Ray 如何與 Amazon S3 搭配使用

AWS X-Ray 支援 Amazon S3 的追蹤內容傳播，因此您可以查看端對端請求在整個應用程式中的傳播。X-Ray 會彙總由個別服務 (例如 Amazon S3、AWS Lambda 和 Amazon EC2) 以及組成應用程式的許多資源所產生的資料。X-Ray 為您提供了應用程式如何執行的整體視圖。

Amazon S3 與 X-Ray 整合以傳播追蹤內容，並為您提供一個具有上游和下游節點的請求鏈。如果上游服務包含有效格式化的追蹤標頭及其 S3 請求，Amazon S3 會在將事件通知傳遞給下游服務 (例如 Lambda、Amazon SQS 和 Amazon SNS) 時傳遞追蹤標頭。如果您已將所有這些服務與 X-Ray 主動整合，則其會連結在一個請求鏈中，以便為您提供 Amazon S3 請求的完整詳細資訊。

若要透過 Amazon S3 傳送 X-Ray 追蹤標頭，您必須在請求中包含格式化的 X-Amzn-Trace-Id。您也可以使用 AWS X-Ray 開發套件來檢測 Amazon S3 用戶端。如需支援的開發套件清單，請參閱 AWS X-Ray 文件。

服務地圖

X-Ray 服務地圖以近乎即時的方式顯示 Amazon S3 與應用程式中其他 AWS 服務和資源之間的關係。若要使用 X-Ray 服務地圖查看端對端請求，您可以使用 X-Ray 主控台來檢視 Amazon S3 與應用程式使用的其他服務之間連線的地圖。您可以輕鬆地偵測出現高延遲的位置、視覺化這些服務的節點分佈，然後深入分析影響應用程式效能的特定服務和路徑。

X-Ray Analytics

您也可以使用 X-Ray Analytics 主控台來分析追蹤、檢視延遲和失敗率等指標，以及產生深入解析以協助您識別問題並疑難排解問題。此主控台也會顯示平均延遲和失敗率等指標。如需詳細資訊，請參閱《AWS X-Ray 開發人員指南》中的 AWS X-Ray 主控台。

可用的區域

所有 AWS X-Ray 區域均提供適用於 Amazon S3 的 AWS X-Ray 支援。如需詳細資訊，請參閱《AWS X-Ray 開發人員指南》中的 Amazon S3 和 AWS X-Ray。

使用 Amazon S3 託管靜態網站

您可以使用Amazon S3 託管靜態網站。在靜態網站中，每個網頁都包含靜態內容。這些內容也可能包含用戶端指令碼。

相較之下，動態網站則倚賴伺服器端的處理，會包含伺服器端指令碼，例如 PHP、JSP 或 ASP.NET。Amazon S3 不支援伺服器端指令碼，但 AWS 具有用於託管動態網站的其他資源。要了解有關網站託管的更多信息 AWS，請參閱虛擬主機。

> **Note**
>
> 您可以使用主 AWS Amplify 控台來託管單頁 Web 應用程式。 AWS Amplify 主控台支援單一頁面應用程式框架 (例如，React JS、Vue JS、Angular JS 和 Nuxt) 和靜態網站產生器 (例如，Gatsby JS、React-static、Jekyll 和 Hugo) 建置的單一頁面應用程式。如需詳細資訊，請參閱《AWS Amplify 主控台使用者指南》中的入門。
>
> Amazon S3 網站端點不支援 HTTPS。如果您想使用 HTTPS，則可以使用 Amazon CloudFront 為 Amazon S3 上託管的靜態網站提供服務。如需詳細資訊，請參閱如 CloudFront 何使用為 Amazon S3 儲存貯體提供 HTTPS 請求？ 若要共同使用 HTTPS 和自訂網域，請參閱使用透過 Route 53 登記的自訂網域配置靜態網站。

如需在 Amazon S3 上託管靜態網站的詳細資訊，包括指示和step-by-step 逐步解說，請參閱下列主題。

主題

- 網站端點
- 啟用網站託管
- 設定索引文件
- 設定自訂錯誤文件
- 設定網站存取許可
- (選y用) 記錄 Web 流量
- (選用) 配置網頁重新導向

網站端點

將儲存貯體設定為網站時，便可透過 AWS 區域專用的網站端點來提供該網站。網站端點不同於您傳送 REST API 要求的端點。如需端點差異的詳細資訊，請參閱「網站端點與 REST API 端點之間的主要差異」。

視您的區域而定，您的 Amazon S3 網站端點會是以下兩種格式之一。

- s3-website dash (-) Region - `http://bucket-name.s3-website-Region.amazonaws.com`
- s3-website 句點 (.) Region - `http://bucket-name.s3-website.Region.amazonaws.com`

這些 URL 會傳回您為網站設定的預設索引文件。如需 Amazon S3 網站端點的完整清單，請參閱 Amazon S3 網站端點。

> **Note**
>
> 為了增強您的 Amazon S3 靜態網站的安全性，Amazon S3 網站端點域（例如，S3-阿馬遜網站或 S3 website-us-east-網站南 1.amazonaws.com）註冊在公共尾碼列表（PSL）中。為了加強安全性，如果您需要在 Amazon S3 靜態網站的網域名稱中設定敏感性 Cookie，建議您使用具有 `__Host-` 字首的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

如果希望網站成為公開狀態，您必須將所有內容開放給大眾讀取，以便客戶能夠在網站端點上存取內容。如需詳細資訊，請參閱 設定網站存取許可。

> **Important**
>
> Amazon S3 網站端點不支援 HTTPS 或存取點。如果您想使用 HTTPS，則可以使用 Amazon CloudFront 為 Amazon S3 上託管的靜態網站提供服務。如需詳細資訊，請參閱如CloudFront 何使用為 Amazon S3 儲存貯體提供 HTTPS 請求？ 若要共同使用 HTTPS 和自訂網域，請參閱使用透過 Route 53 登記的自訂網域配置靜態網站。
>
> 申請者付款儲存貯體不允許透過網站端點存取。所有對這類儲存貯體的請求都會收到 403 存取遭拒回應。如需詳細資訊，請參閱「使用儲存體傳輸和用量的申請者付款儲存貯體」。

主題

網站端點範例

下列範例說明如何存取設定為靜態網站的 Amazon S3 儲存貯體。

Example — 在根層級請求物件

若要請求儲存在儲存貯體根層級的特定物件，請使用下列 URL 結構。

```
http://bucket-name.s3-website.Region.amazonaws.com/object-name
```

例如，以下 URL 會請求儲存在儲存貯體中根層級的 photo.jpg 物件：

```
http://example-bucket.s3-website.us-west-2.amazonaws.com/photo.jpg
```

Example — 請求字首中的物件

若要請求儲存在儲存貯體的資料夾中的物件，請使用此 URL 結構。

```
http://bucket-name.s3-website.Region.amazonaws.com/folder-name/object-name
```

以下 URL 會請求儲存貯體中的 docs/doc1.html 物件。

```
http://example-bucket.s3-website.us-west-2.amazonaws.com/docs/doc1.html
```

新增 DNS CNAME

若您已登記網域，就能新增 DNS CNAME 項目，將其指向 Amazon S3 網站端點。例如，若您已登記 www.example-bucket.com 網域，就能建立儲存貯體 www.example-bucket.com，並新增指向 www.example-bucket.com.s3-website.*Region*.amazonaws.com 的 DNS CNAME 記錄。所有對 http://www.example-bucket.com 要求都會路由到 www.example-bucket.com.s3-website.*Region*.amazonaws.com。

如需詳細資訊，請參閱「使用 CNAME 記錄自訂 Amazon S3 URL」。

搭配 Route 53 使用自訂網域

您可以使用自己向 Amazon Route 53 註冊的網域來提供內容，而不是使用 Amazon S3 網站端點存取網站，例如 `example.com`。您可以使用 Amazon S3 與 Route 53 來託管根域的網站。例如，若您有根網域 `example.com`，並在 Amazon S3 上託管網站，則網站訪客只需要在他們的瀏覽器上輸入 `http://www.example.com` 或 `http://example.com`，就能存取該網站。

如需範例演練，請參閱 教學課程：使用向 Route 53 註冊的自訂網域設定靜態網站。

網站端點與 REST API 端點之間的主要差異

Amazon S3 網站端點已經過最佳化，可以從 Web 瀏覽器存取。下表摘要說明 REST API 端點與網站端點之間的主要差異。

主要差異	REST API 端點	網站端點
存取控制	支援公有與私有的內容	僅支援可供大眾讀取的內容
錯誤訊息處理	傳回 XML 格式的錯誤回應	傳回 HTML 文件
重新導向支援	不適用	支援物件層級與儲存貯體層級的重新導向
支援的要求	支援所有儲存貯體與物件操作	僅支援物件的 GET 與 HEAD 請求
回應儲存貯體根的 GET 與 HEAD 要求	傳回儲存貯體中的物件金鑰清單	傳回網站組態中指定的索引文件
Secure Sockets Layer (SSL) 支援	支援 SSL 連線	不支援 SSL 連線

如需 Amazon S3 端點的完整清單，請參閱《AWS 一般參考》中的 Amazon S3 端點和配額。

啟用網站託管

將儲存貯體設定為靜態網站時，您必須啟用靜態網站託管、設定索引文件，以及設定許可。

您可以使用 Amazon S3 主控台、REST API、 AWS 開發套件、或 AWS CloudFormation啟用靜態網站託管。 AWS CLI

若要使用自訂網域來設定您的網站，請參閱 教學課程：使用向 Route 53 註冊的自訂網域設定靜態網站。

使用 S3 主控台

啟用靜態網站託管

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在 Buckets (儲存貯體名稱) 清單中，選擇希望為其啟用靜態網站託管的儲存貯體名稱。
3. 選擇 Properties (屬性)。
4. 在 Static website hosting (靜態網站託管) 下，選擇 Edit (編輯)。
5. 選擇 Use this bucket to host a website (使用此儲存貯體來託管網站)。
6. 在 Static website hosting (靜態網站託管) 下，選擇Enable (啟用)。
7. 在索引文件中，輸入索引文件的名稱，通常是 `index.html`。

   索引文件名稱區分大小寫，而且必須完全符合您計畫上傳至 S3 儲存貯體的 HTML 索引文件檔案名稱。當您為網站託管設定儲存貯體時，必須指定索引文件。在對根網域或任何子資料夾提出請求時，Amazon S3 會傳回此索引文件。如需詳細資訊，請參閱「設定索引文件」。
8. 若要為 4XX 類別錯誤提供自己的自訂錯誤文件，請在 Error document (錯誤文件) 中輸入自訂錯誤文件檔案名稱。

   錯誤文件名稱區分大小寫，而且必須完全符合您計畫上傳至 S3 儲存貯體的 HTML 錯誤文件檔案名稱。如果您未指定自訂錯誤文件且發生錯誤，則 Amazon S3 會傳回預設的 HTML 錯誤文件。如需詳細資訊，請參閱 設定自訂錯誤文件。
9. (選用) 如果您要指定進階重新導向規則，請在 Redirection rules (重新導向規則) 中輸入 JSON 來描述規則。

   例如，您可依據要求中特定的物件金鑰名稱或字首，依條件路由要求。如需詳細資訊，請參閱「配置重新引導規則以使用進階條件重新引導」。
10. 選擇 Save changes (儲存變更)。

    Amazon S3 會為您的儲存貯體啟用靜態網站託管。在頁面底部的靜態網站託管下，您會看到儲存貯體的網站端點。

11. 在 Static website hosting 下，請記下 Endpoint (端點)。

    端點是儲存貯體的 Amazon S3 網站端點。將儲存貯體設為靜態網站之後，您可以使用此端點來測試您的網站。

使用 REST API

如需有關直接傳送 REST 請求以啟用靜態網站託管的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的下列章節：

- PUT 儲存貯體網站
- GET 儲存貯體網站
- DELETE 儲存貯體網站

使用 AWS 軟體開發套件

若要在 Amazon S3 託管靜態網站，可以設定網站託管用的 Amazon S3 儲存貯體，然後將網站內容上傳至儲存貯體。您也可以使用 AWS SDK，以程式設計方式建立、更新及刪除網站組態。SDK 提供 Amazon S3 REST API 的包裝函式類別。您也可以視應用程式之需要，直接從應用程式傳送 REST API 要求。

.NET

下列範例顯示如何使用 AWS SDK for .NET 來管理值區的網站設定。若要為儲存貯體新增網站組態，請您提供儲存貯體名稱和網站組態。網站組態資訊必須含有索引文件，且包含選擇性錯誤文件。這些文件必須存放在儲存貯體中。如需詳細資訊，請參閱 PUT 儲存貯體網站。如需 Amazon S3 網站功能的詳細資訊，請參閱 使用 Amazon S3 託管靜態網站。

下列 C# 程式碼範例會將網站組態新增至指定的儲存貯體。組態會同時指定索引文件及錯誤文件名稱。如需有關設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
```

```
{
    class WebsiteConfigTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string indexDocumentSuffix = "*** index object key ***"; //
 For example, index.html.
        private const string errorDocument = "*** error object key ***"; // For
 example, error.html.
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddWebsiteConfigurationAsync(bucketName, indexDocumentSuffix,
 errorDocument).Wait();
        }

        static async Task AddWebsiteConfigurationAsync(string bucketName,
                                            string indexDocumentSuffix,
                                            string errorDocument)
        {
            try
            {
                // 1. Put the website configuration.
                PutBucketWebsiteRequest putRequest = new PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument
                    }
                };
                PutBucketWebsiteResponse response = await
 client.PutBucketWebsiteAsync(putRequest);

                // 2. Get the website configuration.
                GetBucketWebsiteRequest getRequest = new GetBucketWebsiteRequest()
                {
                    BucketName = bucketName
                };
```

```
                GetBucketWebsiteResponse getResponse = await
 client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine("Index document: {0}",
 getResponse.WebsiteConfiguration.IndexDocumentSuffix);
                Console.WriteLine("Error document: {0}",
 getResponse.WebsiteConfiguration.ErrorDocument);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

下列 PHP 範例會將網站組態新增至指定的儲存貯體。`create_website_config` 方法明確地提供了索引文件及錯誤文件名稱。此範例同時也擷取網站組態，並會印出回應。如需 Amazon S3 網站功能的詳細資訊，請參閱 使用 Amazon S3 託管靜態網站。

有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);


// Add the website configuration.
$s3->putBucketWebsite([
```

```
    'Bucket'                => $bucket,
    'WebsiteConfiguration'  => [
        'IndexDocument' => ['Suffix' => 'index.html'],
        'ErrorDocument' => ['Key' => 'error.html']
    ]
]);

// Retrieve the website configuration.
$result = $s3->getBucketWebsite([
    'Bucket' => $bucket
]);
echo $result->getPath('IndexDocument/Suffix');

// Delete the website configuration.
$s3->deleteBucketWebsite([
    'Bucket' => $bucket
]);
```

使用 AWS CLI

如需使用將 S3 儲存貯體設定 AWS CLI 為靜態網站的詳細資訊，請參閱AWS CLI 命令參考中的網站。

接下來，您必須設定索引文件並設定許可。如需詳細資訊，請參閱 設定索引文件 及 設定網站存取許可。

您也可以選擇性地設定錯誤文件、Web 流量記錄或重新導向。

設定索引文件

啟用網站託管時，您還必須設定和上傳索引文件。索引文件是對網站的根或任何子資料夾提出請求時，Amazon S3 傳回的網頁。例如，若使用者在瀏覽器中輸入 `http://www.example.com`，表示使用者未要求任何特定頁面。在此情況下，Amazon S3 會提供索引文件，有時也稱為預設頁面。

當您為儲存貯體啟用靜態網站託管時，請輸入索引文件的名稱 (例如，`index.html`)。為儲存貯體啟用靜態網站託管後，您可以將含有索引文件名稱的 HTML 檔案上傳到儲存貯體。

根層級 URL 結尾的斜線並非必要。例如，若將設有 `index.html` 的網站設定為索引文件，下列兩個 URL 的其中之一將會傳回 `index.html`。

```
http://example-bucket.s3-website.Region.amazonaws.com/
```

```
http://example-bucket.s3-website.Region.amazonaws.com
```

如需 Amazon S3 網站端點的詳細資訊，請參閱 網站端點。

索引文件和資料夾

在 Amazon S3 中，儲存貯體是平坦的物件容器。與電腦檔案系統不同，不會提供任何階層式組織。不過，您可以使用指向資料夾結構的物件金鑰名稱建立邏輯階層。

例如，試想有一個儲存貯體具有三個使用下列金鑰名稱的物件。雖然這些物件都不是以實體階層組織方式存放，但您可以從金鑰名稱推斷出下列邏輯資料夾結構：

- `sample1.jpg` — 物件位於儲存貯體的根目錄。
- `photos/2006/Jan/sample2.jpg` — 物件位於 `photos/2006/Jan` 子資料夾。
- `photos/2006/Feb/sample3.jpg` — 物件位於 `photos/2006/Feb` 子資料夾。

在 Amazon S3 主控台中，您也可以建立儲存貯體裡的資料夾。例如，您可以建立名為 `photos` 的資料夾。您可以將物件上傳到儲存貯體，或是上傳到儲存貯體內的 `photos` 資料夾。若將物件 `sample.jpg` 新增到儲存貯體，其金鑰名稱為 `sample.jpg`。若將物件上傳到 `photos` 資料夾，其金鑰名稱為 `photos/sample.jpg`。

若要在儲存貯體中建立資料夾結構，則各階層都必須具備一份索引文件。在每個資料夾中，索引文件必須具有相同的名稱，例如 `index.html`。當使用者指定類似於資料夾查詢的 URL 時，結尾加或不加斜線將會決定網站的行為。例如，下列 URL 在結尾加了斜線，將會傳回 `photos/index.html` 索引文件。

```
http://bucket-name.s3-website.Region.amazonaws.com/photos/
```

若將上述的 URL 結尾不含斜線，Amazon S3 將會優先在儲存貯體中尋找物件 `photos`。若找不到 `photos` 物件，將會搜尋索引文件 `photos/index.html`。如有找到該文件，Amazon S3 將會傳回 `302 Found` 訊息並指向 `photos/` 金鑰。對於後續的 `photos/` 請求，Amazon S3 會傳回 `photos/index.html`。若找不到索引文件，Amazon S3 會傳回錯誤。

配置索引文件

若要使用 S3 主控台設定索引文件，請使用下列程序。您也可以使用 REST API、 AWS 軟體開發套件、或 AWS CloudFormation來設定索引文件。 AWS CLI

> **Note**
>
> 在啟用版本控制的儲存貯體中，您可以上傳多個 `index.html` 複本，但只會解析為最新版本。如需使用 S3 版本控制的詳細資訊，請參閱 在 S3 儲存貯體中使用版本控制。

當您為儲存貯體啟用靜態網站託管時，請輸入索引文件的名稱 (例如，**`index.html`**)。為儲存貯體啟用靜態網站託管後，您可以將含有索引文件名稱的 HTML 檔案上傳到儲存貯體。

設定索引文件

1. 建立 `index.html` 檔案。

   如果您還沒有 `index.html` 檔案，您可以使用下列HTML 建立一個檔案。

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

2. 在本機儲存索引檔案。

   索引文件檔案名稱必須完全符合您在 Static website hosting (靜態網站託管) 對話方塊中輸入的索引文件名稱。索引文件名稱有區分大小寫。例如，如果您在 Static website hosting (靜態網站託管) 對話方塊的 Index document (索引文件) 名稱中輸入 `index.html`，您的索引文件檔案名稱也必須是 `index.html` 而非 `Index.html`。

3. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

4. 在 Buckets (儲存貯體) 清單中，選擇您要用於託管靜態網站的儲存貯體名稱。

5. 為您的儲存貯體啟用靜態網站，然後輸入索引文件的確切名稱 (例如，`index.html`)。如需詳細資訊，請參閱「啟用網站託管」。

   啟用靜態網路託管之後，請繼續執行步驟 6。

6. 若要將索引文件上傳至您的儲存貯體，請執行下列其中一項：

- 將索引檔拖放到主控台儲存貯體清單中。
- 選擇 Upload (上傳)，然後依照提示選擇並上傳索引檔案。

如需 step-by-step 指示，請參閱上傳物件。

7. (選用) 將其他網站內容上傳到您的儲存貯體。

接下來，您必須設定存取網站的許可。如需相關資訊，請參閱「設定網站存取許可」。

您也可以選擇性地設定錯誤文件、Web 流量記錄或重新導向。

設定自訂錯誤文件

將儲存貯體配置為靜態網站後，若發生錯誤，則 Amazon S3 會傳回 HTML 錯誤文件。您可以選擇使用自訂錯誤文件來配置您的儲存貯體，使 Amazon S3 在發生錯誤時傳回該文件。

> **Note**
>
> 有一些瀏覽器會在錯誤發生時顯示自有的錯誤訊息，並忽略 Amazon S3 傳回的錯誤文件。例如當發生 HTTP 404 Not Found (HTTP 404 找不到) 錯誤時，Google Chrome 可能會忽略 Amazon S3 傳回的錯誤文件，而只顯示自己的錯誤。

主題

Amazon S3 HTTP 回應代碼

下表列出發生錯誤時，Amazon S3 傳回之 HTTP 回應碼中的一部分。

HTTP 錯誤代碼	描述
301 Moved Permanently (301 永久移除)	當使用者直接向 Amazon S3 網站端點 (`http://s3-website.` *`Region`*`.amazonaws.com/`) 傳送請求時，Amazon S3 會傳回 301 永久移除回應，並將這些請求重新導向至 `https://aws.amazon.com/s3/` 。

HTTP 錯誤代碼	描述
302 Found (302 已找到)	當 Amazon S3 收到金鑰 x、http://*bucket-name* .s3-website. *Region*.amazonaws.com/x (結尾不含斜線) 的請求時，會先尋找有金鑰名稱為 x 的物件。若找不到該物件，Amazon S3 會判定此請求乃針對子資料夾 x，並在結尾處新增斜線重新導向請求，然後傳回 302 已找到。
304 Not Modified (304 未修改)	Amazon S3 使用請求標頭 If-Modified-Since 、If-Unmodified-Since 、If-Match 及 (或) If-None-Match 來確定請求的物件與用戶端持有的快取複本是否相同。若物件相同，網站端點會傳回 304 Not Modified (304 未修改) 回應。
400 Malformed Request (400 請求格式不正確)	當使用者嘗試透過不正確的區域端點存取儲存貯體時，網站端點會以 400 Malformed Request (請求格式不正確) 回應。
403 Forbidden (403 禁止)	當使用者請求轉譯成不可公開讀取的物件時，網站端點會以 403 Forbidden (403 禁止) 回應。物件擁有者必須使用儲存貯體政策或物件 ACL，將物件設為可供大眾讀取。

HTTP 錯誤代碼	描述
404 Not Found (404 找不到)	網站端點在下列幾種情況會以 404 Not Found (404 找不到) 回應： • Amazon S3 判斷網站 URL 所參考的物件金鑰不存在。 • Amazon S3 推斷所請求的索引文件不存在。 • URL 中指定的儲存貯體不存在。 • URL 中指定的儲存貯體存在，但未設定成網站。 您可以建立自訂文件，供傳回 404 Not Found (404 找不到) 時使用。請務必將文件上傳到已設定為網站的儲存貯體，並將網站託管組態設為使用該文件。 如需 Amazon S3 如何將 URL 解譯成物件或索引文件請求的資訊，請參閱 設定索引文件。
500 Service Error (500 服務錯誤)	當發生內部伺服器錯誤時，網站端點會以 500 Service Error (500 服務錯誤) 回應。
503 Service Unavailable (503 服務無法使用)	當 Amazon S3 認為您需要降低請求率時，網站端點會回應 503 服務無法使用。

對於這些錯誤，Amazon S3 一律會傳回預先定義的 HTML 訊息。以下範例是針對 403 Forbidden (403 禁止) 回應所傳回的範例 HTML 訊息。

設定自訂錯誤文件

將儲存貯體設定為靜態網站時，您可以提供自訂錯誤文件，其中包含易於使用的錯誤訊息和其他說明。只有 HTTP 4XX 類別的錯誤訊息，Amazon S3 才會傳回您自訂的錯誤文件。

若要使用 S3 主控台配置自訂錯誤文件，請依照下列步驟操作。您也可以使用 REST API、 AWS 軟體開發套件、或 AWS CloudFormation來設定錯誤文件。 AWS CLI如需詳細資訊，請參閱下列內容：

- PutBucketWebsiteAmazon 簡單存儲服務 API 參考
- AWS::S3::Bucket WebsiteConfiguration《AWS CloudFormation 使用者指南》中的
- 《AWS CLI 命令參考》中的 put-bucket-website 一節

當您為儲存貯體啟用靜態網站託管時，請輸入錯誤文件的名稱 (例如 **`404.html`**)。為儲存貯體啟用靜態網站託管後，您可以將含有錯誤引文件名稱的 HTML 檔案上傳到儲存貯體。

設定錯誤文件

1. 建立錯誤文件，例如 `404.html`。
2. 將錯誤文件檔案儲存在本機。

   錯誤文件名稱區分大小寫，且須完全符合您在啟用靜態網站託管時所輸入的名稱。例如，如果您在 Static website hosting (靜態網站託管) 對話方塊的 Error document (錯誤文件) 名稱中輸入 `404.html`，您的錯誤文件檔案名稱也必須是 `404.html`。

3. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
4. 在 Buckets (儲存貯體) 清單中，選擇您要用於託管靜態網站的儲存貯體名稱。
5. 為您的儲存貯體啟用靜態網站託管，並輸入錯誤文件的確切名稱 (例如 `404.html`)。如需詳細資訊，請參閱 啟用網站託管 及 設定自訂錯誤文件。

   啟用靜態網路託管之後，請繼續執行步驟 6。
6. 若要將錯誤文件上傳至您的儲存貯體，請執行下列其中一項：

   - 將錯誤文件檔案拖放至主控台儲存貯體清單中。
   - 選擇 Upload (上傳)，然後依照提示選擇並上傳索引檔案。

   如需 step-by-step 指示，請參閱上傳物件。

設定網站存取許可

當您將儲存貯體作為靜態網站時，如果您希望您的網站是公開的，則可以授予公開讀取權限。為使得儲存貯體能為公眾讀取，您必須停用儲存貯體的封鎖公開存取設定，並編寫授予公開存取權限的儲存貯體政策。如果您的儲存貯體包含不屬於儲存貯體擁有者的物件，可能也需要新增物件存取控制清單 (ACL)，以授予所有人這些物件的讀取存取權。

如果您不想停用儲存貯體的封鎖公開存取設定，但仍希望網站公開，您可以建立 Amazon CloudFront 分發來為您的靜態網站提供服務。如需詳細資訊，請參閱 Amazon R oute 53 開發人員指南中的加快您的網站與 Amazon CloudFront或使用 Amazon CloudFront 分發服務靜態網站。

> **Note**
>
> 在網站端點上，若使用者請求的物件不存在，Amazon S3 會傳回 HTTP 回應碼 `404 (Not Found)`。若物件存在，但您未授予其讀取許可，網站端點會傳回 HTTP 回應碼 `403 (Access Denied)`。使用者可以使用回應碼推斷特定物件是否存。若不希望有此行為，請勿啟用儲存貯體的網站支援。

主題

- 步驟 1：編輯 S3 封鎖公有存取設定
- 步驟 2：新增儲存貯體政策

- 物件存取控制清單

步驟 1：編輯 S3 封鎖公有存取設定

若要將現有的儲存貯體配置為具有公有存取權的靜態網站，您必須編輯該儲存貯體的封鎖公有存取配置。您可能還必須編輯帳戶層級的「封鎖公開存取」設定。Amazon S3 會套用儲存貯體層級和帳戶層級「封鎖公開存取」設定中限制性最高的組合。

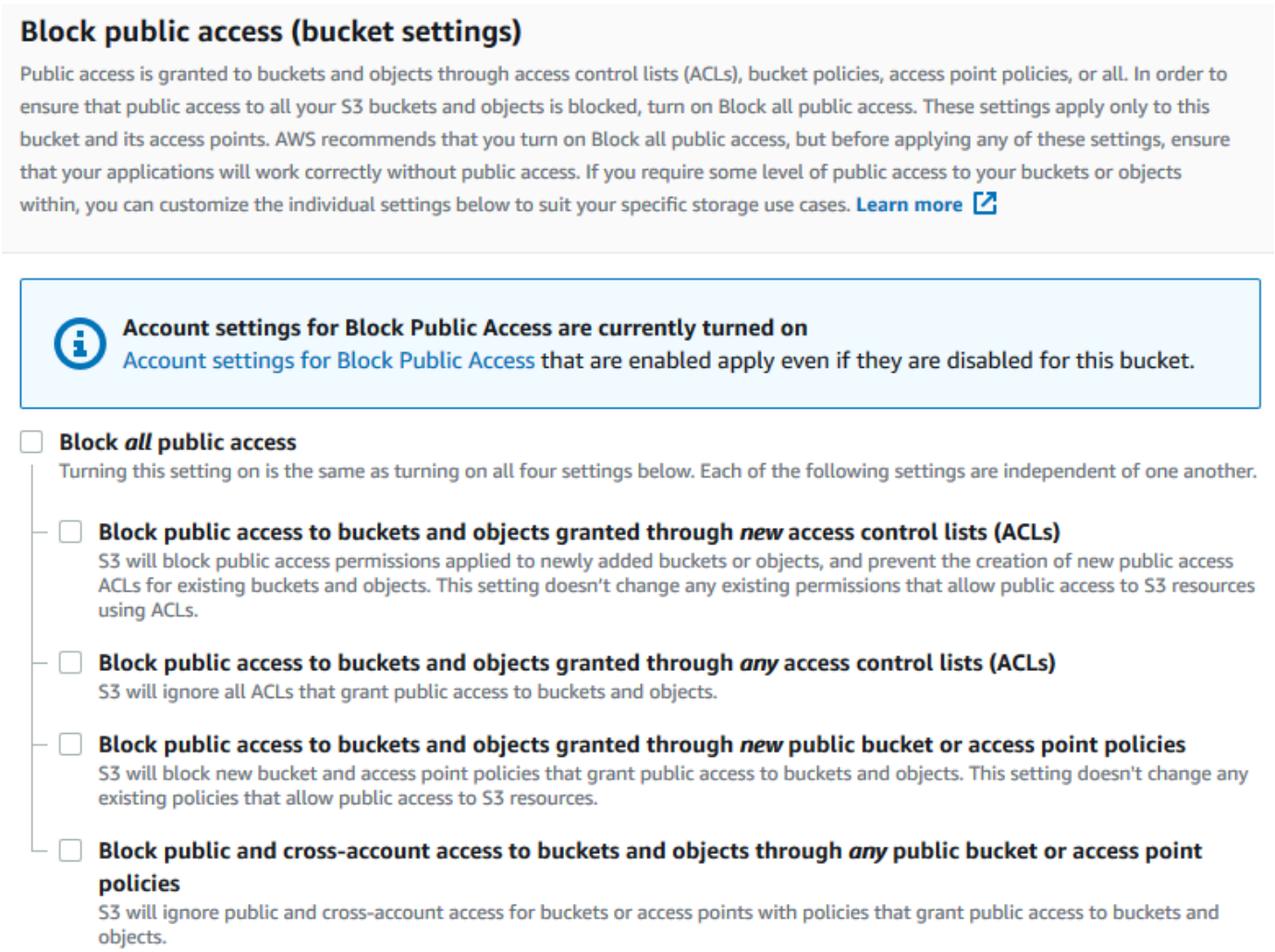
例如，如果允許儲存貯體的公有存取，但封鎖帳戶層級的所有公有存取，則 Amazon S3 會繼續封鎖對該儲存貯體的公有存取。如果是這種情況，您就必須編輯自己的儲存貯體層級和帳戶層級封鎖公有存取設定。如需詳細資訊，請參閱「封鎖對 Amazon S3 儲存體的公開存取權」。

根據預設，Amazon S3 會封鎖對帳戶和儲存貯體的公開存取。如想要使用儲存貯體託管靜態網站，您可使用這些步驟編輯封鎖公有存取設定：

> **Warning**
>
> 在完成此步驟之前，請檢閱 封鎖對 Amazon S3 儲存體的公開存取權 以確保您了解並接受允許公開存取所涉及的風險。當您關閉封鎖公開存取設定以公開儲存貯體時，網際網路上的任何人都可以存取您的儲存貯體。我們建議您封鎖對儲存貯體的所有公用存取權。

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 選擇已設定為靜態網站的儲存貯體名稱。
3. 選擇 Permissions (許可)。
4. 在 Block public access (bucket settings) (封鎖公開存取 (儲存貯體設定)) (封鎖公開存取 (儲存貯體設定)) 下，選擇 Edit (編輯)。
5. 清除 Block all public access (封鎖所有公開存取)，然後選擇 Save changes (儲存變更)。

> **Warning**
>
> 在完成此步驟之前，請檢閱 封鎖對 Amazon S3 儲存體的公開存取權 以確保您了解並接受允許公開存取所涉及的風險。當您關閉封鎖公開存取設定以公開儲存貯體時，網際網路上的任何人都可以存取您的儲存貯體。我們建議您封鎖對儲存貯體的所有公用存取權。



Amazon S3 會關閉儲存貯體的封鎖公開存取設定。若要建立公開的靜態網站，在新增儲存貯體原則之前，可能還需要針對您的帳戶編輯封鎖公開存取設定。如果帳戶的封鎖公開存取設定目前已開啟，您在 封鎖公開存取 (儲存貯體設定) 下會看到附註。

步驟 2：新增儲存貯體政策

若要讓儲存貯體中的物體能為公眾讀取，您必須編寫儲存貯體政策，授予所有人 `s3:GetObject` 許可。

編輯 S3 封鎖公用存取設定之後，您可以新增儲存貯體政策，以授予儲存貯體的公用讀取權限。當您授予公有讀取權限時，網際網路上的任何人都可以存取您的儲存貯體。

> **Important**
>
> 以下政策僅為範例，允許完整存取您儲存貯體的內容。繼續執行此步驟之前，請檢閱如何保護 Amazon S3 儲存貯體中的檔案？，以確保您瞭解 S3 儲存貯體中檔案保護的最佳實務，以及授予公開存取權所涉及的風險。

1. 在 Buckets(儲存貯體) 下方，選擇儲存貯體的名稱。
2. 選擇 Permissions (許可)。
3. 在 Bucket Policy (儲存貯體政策) 下方，選擇 Edit (編輯)。
4. 若要授予您網站的公開讀取存取權，請複製以下儲存貯體政策，並將它貼上至 Bucket policy editor (儲存貯體政策編輯器)。

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

5. 將 `Resource` 更新為您的儲存貯體名稱。

   在上述範例儲存貯體政策中，*`Bucket-Name`* 是儲存貯體名稱的預留位置。若要使用此儲存貯體策略與您自己的儲存貯體搭配，您必須更新此名稱以符合您的儲存貯體名稱。

6. 選擇 Save changes (儲存變更)。

   顯示的訊息指出已成功新增儲存貯體原則。

如果您看到指出 `Policy has invalid resource` 的錯誤，請確認儲存貯體政策中的儲存貯體名稱與您的儲存貯體名稱相符。如需有關新增儲存貯體原則的資訊，請參閱如何新增 S3 儲存貯體原則？

如果您收到錯誤訊息且無法儲存貯體原則，請檢查您的帳戶和儲存貯體的封鎖公開存取設定，以確認您允許公開存取儲存貯體。

物件存取控制清單

您可以使用儲存貯體政策，授予您物件的公有讀取許可。但是，儲存貯體政策僅適用於儲存貯體擁有者所擁有的物件。若您的儲存貯體包含不屬於儲存貯體擁有者的物件，建議儲存貯體擁有者使用物件存取控制清單 (ACL) 授予這些物件的公有 READ 許可。

S3 物件擁有權是一項 Amazon S3 儲存貯體層級設定，您可以用來同時控制上傳至儲存貯體之物件的擁有權，以及停用或啟用 ACL。根據預設，物件擁有權設定為「儲存貯體擁有者強制執行」設定，而且所有 ACL 都會停用。停用 ACL 時，儲存貯體擁有者會擁有儲存貯體中的所有物件，並使用存取管理政策專門管理對這些物件的存取。

Amazon S3 中的大多數新式使用案例不再需要使用 ACL。建議您將 ACL 保持停用狀態，除非在異常情況下必須個別控制每個物件的存取。停用 ACL 後，您可以使用政策來控制對儲存貯體中所有物件的存取，無論是誰將物件上傳到您的儲存貯體。如需詳細資訊，請參閱 控制物件的擁有權並停用儲存貯體的 ACL。

> **Important**
>
> 如果儲存貯體使用 S3 物件擁有權的儲存貯體擁有者強制執行設定，則您必須使用政策將存取權授予儲存貯體及其中的物件。在啟用儲存貯體擁有者強制執行設定的情況下，請求設定存取控制清單 (ACL) 或更新 ACL 失敗，並傳回 `AccessControlListNotSupported` 錯誤碼。仍支援讀取 ACL 的請求。

若使用 ACL 將物件設為可供大眾讀取，請將 READ 許可授予 `AllUsers` 群組，如下列授予元素所示。將此授予元素新增到物件 ACL。如需如何管理 ACL 的資訊，請參閱「存取控制清單 (ACL) 概觀」。

```
<Grant>
  <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
          xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
```

(選y用) 記錄 Web 流量

您可以選擇性地針對配置為靜態網站的儲存貯體啟用 Amazon S3 伺服器存取日誌日誌。伺服器存取記錄日誌，應您的儲存貯體要求，提出的詳細記錄。如需詳細資訊，請參閱 使用伺服器存取記錄記錄要求。如果您打算使用 Amazon CloudFront 加快網站速度，也可以使用 CloudFront 日誌記錄。如需詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的設定和使用存取日誌。

為您的靜態網站儲存貯體啟用伺服器存取記錄

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在建立設為靜態網站之儲存貯體的相同區域中，建立用於記錄的儲存貯體，例如 `logs.example.com`。
3. 為伺服器存取記錄日誌檔建立資料夾 (例如，`logs`)。
4. (選擇性) 如果您想 CloudFront 要用來改善網站效能，請為 CloudFront 記錄檔建立資料夾 (例如`cdn`)。

   如需詳細資訊，請參閱 加快您的網站與 Amazon CloudFront。
5. 在 Buckets (儲存貯體) 清單中，選擇您的儲存貯體。
6. 選擇 Properties (屬性)。
7. 在 Server access logging (伺服器存取記錄) 下，選擇 Edit (編輯)。
8. 選擇 Enable (啟用)。
9. 在目的地儲存貯體下，選擇伺服器存取日誌的儲存貯體和資料夾目的地：
   - 瀏覽至資料夾和儲存貯體位置：
     1. 選擇 Browse S3 (瀏覽 S3)。
     2. 選擇儲存貯體名稱，然後選擇日誌資料夾。
     3. 選擇 Choose path (選擇路徑)。
   - 輸入 S3 儲存貯體路徑，例如 **`s3://logs.example.com/logs/`**。
10. 選擇 Save changes (儲存變更)。

在您的日誌儲存貯體中，您現在可以存取您的日誌。Amazon S3 會每隔 2 小時，將網站存取日誌寫入您的日誌儲存貯體。

(選用) 配置網頁重新導向

如果您的 Amazon S3 儲存貯體配置為靜態網站託管，則您可配置儲存貯體或其中物件的重新引導。下列選項皆可供您設定重新引導。

主題

將儲存貯體網站端點的請求重新引導至其他儲存貯體或網域

您可以將對儲存貯體之網站端點的所有請求重新引導至其他儲存貯體或網域。如果您重新導向所有請求，對網站端點提出的任何請求都會重新引導至指定的儲存貯體或網域。

例如，若根網域為 `example.com`，而您想要處理對於 `http://example.com` 及 `http://www.example.com` 的要求，則必須建立兩個儲存貯體，並各自命名為 `example.com` 及 `www.example.com`。接著，維護 `example.com` 儲存貯體中的內容，再設定其他 `www.example.com` 儲存貯體，將所有請求重新導向至 `example.com` 儲存貯體。如需詳細資訊，請參閱使用自訂網域名稱設定靜態網站。

重新導向儲存貯體網站端點的請求

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在 Buckets (儲存貯體) 下，選擇要重新導向所有請求的來源儲存貯體名稱 (如 `www.example.com`)。
3. 選擇 Properties (屬性)。
4. 在 Static website hosting (靜態網站託管) 下，選擇 Edit (編輯)。
5. 選擇 Redirect requests for an object (重新導向物件請求)。
6. 在主機名稱方塊中，輸入儲存貯體或自訂網域的網站端點。

   例如，如果您要重新導向至根網域位址，您會輸入 **`example.com`**。

7. 在 Protocol (通訊協定) 中，選擇重新導向請求的通訊協定 (無、http 或 https)。

   如果您未指定通訊協定，則預設選項為無。

8. 選擇 Save changes (儲存變更)。

配置重新引導規則以使用進階條件重新引導

您可以使用進階重新導向規則，依據特定的物件金鑰名稱、要求的字首或回應碼，有條件地路由要求。例如，假設您要刪除或重新命名儲存貯體中的物件。您可以新增路由規則，將要求重新導向至另一個物件。若您想要將資料夾設為無法使用，可以新增路由規則，將要求重新導向至另一個網頁。您也可以新增路由規則處理錯誤條件，方法是在處理錯誤時，將傳回錯誤的要求路由到其他網域。

在啟用儲存貯體的靜態網站託管時，您可以選擇性地指定進階重新引導規則。Amazon S3 具有每個網站組態 50 個路由規則的限制。如果需要 50 個以上的路由規則，您可以使用物件重新導向。如需詳細資訊，請參閱 使用 S3 主控台。

如需有關使用 REST API 設定路由規則的詳細資訊，請參閱 Amazon 簡單儲存服務 API 參考PutBucketWebsite中的。

> **Important**
>
> 若要在新的 Amazon S3 主控台中建立重新導向規則，您必須使用 JSON。如需 JSON 範例，請參閱 重新導向規則範例。

設定靜態網站的重新導向規則

若要為已啟用靜態網站託管的儲存貯體新增重新導向規則，請依照下列步驟進行。

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在 Buckets (儲存貯體) 清單中，選擇已設定為靜態網站的儲存貯體名稱。
3. 選擇 Properties (屬性)。
4. 在 Static website hosting (靜態網站託管) 下，選擇 Edit (編輯)。
5. 在 Redirection rules (重新導向規則) 方塊中，輸入 JSON 中的重新導向規則。

   在 S3 主控台中，您可以使用 JSON 描述規則。如需 JSON 範例，請參閱 重新導向規則範例。Amazon S3 具有每個網站組態 50 個路由規則的限制。

6. 選擇 Save changes (儲存變更)。

路由規則元素

以下是在網站組態中定義路由規則的一般語法 (JSON 和 XML 格式)。若要在新的 S3 主控台中設定重新導向規則，您必須使用 JSON。如需 JSON 範例，請參閱 重新導向規則範例。

JSON

```
[
    {
      "Condition": {
        "HttpErrorCodeReturnedEquals": "string",
        "KeyPrefixEquals": "string"
      },
      "Redirect": {
        "HostName": "string",
        "HttpRedirectCode": "string",
        "Protocol": "http"|"https",
        "ReplaceKeyPrefixWith": "string",
        "ReplaceKeyWith": "string"
      }
    }
  ]

Note: Redirect must each have at least one child element. You can have either
 ReplaceKeyPrefix with or ReplaceKeyWith but not both.
```

XML

```
<RoutingRules> =
    <RoutingRules>
         <RoutingRule>...</RoutingRule>
         [<RoutingRule>...</RoutingRule>
         ...]
    </RoutingRules>

<RoutingRule> =
   <RoutingRule>
      [ <Condition>...</Condition> ]
      <Redirect>...</Redirect>
   </RoutingRule>

<Condition> =
   <Condition>
```

```
      [ <KeyPrefixEquals>...</KeyPrefixEquals> ]
      [ <HttpErrorCodeReturnedEquals>...</HttpErrorCodeReturnedEquals> ]
   </Condition>
    Note: <Condition> must have at least one child element.

<Redirect> =
   <Redirect>
      [ <HostName>...</HostName> ]
      [ <Protocol>...</Protocol> ]
      [ <ReplaceKeyPrefixWith>...</ReplaceKeyPrefixWith>  ]
      [ <ReplaceKeyWith>...</ReplaceKeyWith> ]
      [ <HttpRedirectCode>...</HttpRedirectCode> ]
   </Redirect>

Note: <Redirect> must have at least one child element. You can have either
 ReplaceKeyPrefix with or ReplaceKeyWith but not both.
```

下表說明路由規則中的元素。

名稱	描述
RoutingRules	容器，包含 RoutingRule 元素集合。
RoutingRule	規則，可識別條件及符合條件時所套用的重新導向。 條件： • RoutingRules 容器至少須包含一項路由規則。
Condition	容器，描述要套用指定的重新導向時所須符合的條件。若路由規則不含條件，便會將該規套用到所有要求。
KeyPrefixEquals	物件金鑰的字首，而該物件金鑰是重新導向要求的起源。 若未指定 KeyPrefixEquals ，即須指定 HttpErrorCodeReturnedEquals 。若同時指定了 KeyPrefix

名稱	描述
	Equals 及 HttpErrorCodeReturnedEquals ，則兩者都必須為 true 才符合條件。
HttpErrorCodeReturnedEquals	HTTP 錯誤代碼，套用重新導向時必須符合此代碼。發生錯誤時，若錯誤代碼符合此值，便會套用指定的重新導向。 若未指定 HttpErrorCodeReturnedEquals ，即須指定 KeyPrefixEquals 。若同時指定了 KeyPrefixEquals 及 HttpErrorCodeReturnedEquals ，則兩者都必須為 true 才符合條件。
Redirect	容器元素，提供重新導向要求的指示。您可以將要求重新導向至其他主機或頁面，或是指定使用其他協定。RoutingRule 必須具有 Redirect 元素。Redirect 元素至少須包含下列一個同級元素：Protocol、HostName、ReplaceKeyPrefixWith 、ReplaceKeyWith 或 HttpRedirectCode 。
Protocol	回應傳回的 http 標頭中要使用的 https 或 Location 協定。 若已提供其同級項目，便無須 Protocol。
HostName	回應傳回的 Location 標頭中要使用的主機名稱。 若已提供其同級項目，便無須 HostName。
ReplaceKeyPrefixWith	物件金鑰名稱的字首，會取代重新導向要求中的 KeyPrefixEquals 值。 若已提供其同級項目，便無須 ReplaceKeyPrefixWith 。只在未提供 ReplaceKeyWith 時才須提供。

名稱	描述
ReplaceKeyWith	回應傳回的 Location 標頭中要使用的物件金鑰。 若已提供其同級項目，便無須 ReplaceKeyWith 。只在未提供 ReplaceKeyPrefixWith 時才須提供。
HttpRedirectCode	回應傳回的 Location 標頭中要使用的 HTTP 重新導向代碼。 若已提供其同級項目，便無須 HttpRedirectCode 。

重新導向規則範例

下列範例說明常見的重新導向工作：

> **Important**
> 若要在新的 Amazon S3 主控台中建立重新導向規則，您必須使用 JSON。

Example 1：重新命名金鑰字首後重新導向

假設您的儲存貯體包含下列物件：

- index.html
- docs/article1.html
- docs/article2.html

您決定將資料夾從 docs/ 重新命名為 documents/。當完成此變更後，您必須將字首為 docs/ 的要求重新導向至 documents/。例如，docs/article1.html 的要求會重新導向至 documents/article1.html。

在本案例中，您需要在網站組態中新增下列路由規則。

JSON

```
[
```

```
    {
        "Condition": {
            "KeyPrefixEquals": "docs/"
        },
        "Redirect": {
            "ReplaceKeyPrefixWith": "documents/"
        }
    }
]
```

XML

```
  <RoutingRules>
    <RoutingRule>
    <Condition>
      <KeyPrefixEquals>docs/</KeyPrefixEquals>
    </Condition>
    <Redirect>
      <ReplaceKeyPrefixWith>documents/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

Example 2：將對已刪除之資料夾的要求重新導向至頁面

假設您要刪除 images/ 資料夾 (亦即刪除金鑰字首為 images/ 的所有物件)。您可以新增路由規則，將對於所有金鑰字首為 images/ 之物件的要求重新導向至名為 folderdeleted.html 的頁面。

JSON

```
[
    {
        "Condition": {
            "KeyPrefixEquals": "images/"
        },
        "Redirect": {
            "ReplaceKeyWith": "folderdeleted.html"
        }
    }
]
```

XML

```
  <RoutingRules>
    <RoutingRule>
    <Condition>
       <KeyPrefixEquals>images/</KeyPrefixEquals>
    </Condition>
    <Redirect>
      <ReplaceKeyWith>folderdeleted.html</ReplaceKeyWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

Example 3：重新導向 HTTP 錯誤

假設找不到請求的物件時，您想要將請求重新導向至 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。新增重新導向規則，以在傳回 HTTP 狀態代碼 404 (Not Found) (找不到) 時，將網站訪客重新導向至處理請求的 Amazon EC2 執行個體。

下列範例也會在重新導向中，插入物件金鑰字首 `report-404/`。例如，若您請求頁面 `ExamplePage.html`，卻收到了 HTTP 404 錯誤，便會將該請求重新導向至指定之 Amazon EC2 執行個體的頁面 `report-404/ExamplePage.html`。若無任何路由規則，但發生了 HTTP 錯誤 404，將會傳回組態中指定的錯誤文件。

JSON

```
[
    {
        "Condition": {
            "HttpErrorCodeReturnedEquals": "404"
        },
        "Redirect": {
            "HostName": "ec2-11-22-333-44.compute-1.amazonaws.com",
            "ReplaceKeyPrefixWith": "report-404/"
        }
    }
]
```

XML

```
  <RoutingRules>
```

```
    <RoutingRule>
    <Condition>
      <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals >
    </Condition>
    <Redirect>
      <HostName>ec2-11-22-333-44.compute-1.amazonaws.com</HostName>
      <ReplaceKeyPrefixWith>report-404/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

物件的重新引導請求

您可以在物件的中繼資料內設定網站重新導向位置，將物件要求重新導向至其他物件或 URL。只要將 `x-amz-website-redirect-location` 屬性新增到物件中繼資料，就能設定重新導向。在 Amazon S3 主控台中，您可以在物件的中繼資料內設定 網站重新導向位置。如果您使用 Amazon S3 API，則可以設置 `x-amz-website-redirect-location`。網站會將該物件解譯為 301 重新導向。

若要將要求重新導向至其他物件，必須將重新導向位置設為目標物件的金鑰。若要將要求重新導向至外部 URL，必須將重新導向位置設為所需的 URL。如需物件中繼資料的詳細資訊，請參閱「系統定義的物件中繼資料」。

設定頁面重新導向時，可以保留或刪除來源物件內容。例如，若儲存貯體中有 `page1.html` 物件，您可以將任何對此頁面的要求重新導向至另一個物件 `page2.html`。您有兩種選擇：

- 保留 `page1.html` 物件的內容並重新導向頁面要求。
- 刪除 `page1.html` 的內容並上傳名為 `page1.html` 的零位元組物件，以取代現有物件及重新導向頁面要求。

使用 S3 主控台

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在 Buckets (儲存貯體) 清單中，選擇已配置為靜態網站的儲存貯體名稱 (例如 `example.com`)。
3. 在 Objects (物件) 下，選擇您的物件。
4. 選擇 Actions (動作)，然後選擇 Edit metadata (編輯中繼資料)。
5. 選擇 Metadata (中繼資料)。
6. 選擇 Add Metadata (新增中繼資料)。

7. 在 Type (類型) 下，選擇 System Defined (系統定義)。
8. 在 [索引鍵] 中，選擇 [x-amz-website-redirect位置]。
9. 在 Value (值) 中，輸入您要重新導向至的物件金鑰名稱，例如 `/page2.html`。

   如果是同一儲存貯體中的其他物件，則此值的字首必須是 `/`。您也可以將值設為外部 URL，例如 `http://www.example.com`。
10. 選擇 Edit metadata (編輯中繼資料)。

使用 REST API

下列 Amazon S3 API 動作支援請求中的 `x-amz-website-redirect-location` 標頭。Amazon S3 會在物件中繼資料中將標頭值儲存為 `x-amz-website-redirect-location`。

- PUT 物件
- 啟動分段上傳
- POST 物件
- PUT 物件 - 複製

設定為網站託管用的儲存貯體同時具有網站端點與 REST 端點。已設為 301 重新導向的頁面要求可能會有下列結果，視要求的端點而定：

- 區域專用的網站端點 – Amazon S3 會依據 `x-amz-website-redirect-location` 屬性的值重新導向頁面請求。
- REST 端點 – Amazon S3 不會重新導向頁面請求，而會傳回所要求的物件。

如需端點的詳細資訊，請參閱「網站端點與 REST API 端點之間的主要差異」。

您可以在設定頁面重新導向時，決定要保留或刪除物件內容。例如，假設您的儲存貯體中有 `page1.html` 物件。

- 若要保留 `page1.html` 的內容，並只重新導向頁面請求，您可以提交 PUT 物件 - 複製請求，以現有的 `page1.html` 物件作為來源建立新的 `page1.html` 物件。在您的要求中設定 `x-amz-website-redirect-location` 標頭。當請求完成後，原始頁面及其內容保持不變，但 Amazon S3 會將所有對此頁面的請求重新導向至您指定的位置。
- 若要刪除 `page1.html` 物件的內容，並重新導向頁面的請求，可以傳送 PUT 物件請求，上傳具有相同物件金鑰 `page1.html` 的零位元組物件。在 PUT 要求中，將 `x-amz-website-redirect-`

location 的 page1.html 設為新物件。當要求完成後，page1.html 中不具任何內容，而要求會重新導向至 x-amz-website-redirect-location 指定的位置。

當您使用 GET 物件動作擷取物件及其他物件中繼資料時，Amazon S3 會在回應中傳回 x-amz-website-redirect-location 標頭。

使用 Amazon S3 進行開發

本節涵蓋使用 Amazon S3 的開發人員相關主題。如需詳細資訊，請檢閱下列主題。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱什麼是 S3 Express One Zone？ 和 目錄值區。

主題

提出要求

Amazon S3 是一個 REST 服務。您可以使用 REST API 或 AWS 開發套件 (請參閱範本程式碼與程式庫) 包裝函式程式庫 (其會包裝基礎 Amazon S3 REST API)，傳送請求至 Amazon S3，從而簡化程式設計工作。

與 Amazon S3 的每次互動，可以經過驗證身分或是匿名進行。身分驗證是確認嘗試存取 Amazon Web Services (AWS) 產品之申請者身分的過程。經過身分驗證的要求，必須包含能驗證要求傳送者身分的簽章值。就某部分而言，簽章值產生自申請者的 AWS 存取金鑰 (存取金鑰 ID 與私密存取金鑰)。如需取得存取金鑰的詳細資訊，請參閱《AWS 一般參考》中的如何取得安全憑證？。

若目前使用 AWS 開發套件，程式庫會運算您提供之金鑰中的簽章。但如果直接在應用程式中呼叫 REST API，您必須撰寫程式碼來運算簽章，並將其新增至要求。

主題

- 透過 IPv6 向 Amazon S3 提出請求
- 使用 AWS 開發套件提出請求
- 使用 REST API 提出要求

關於存取金鑰

以下各節會檢閱您可用於驗證要求的存取金鑰類型。

AWS 帳戶 存取金鑰

帳戶存取金鑰提供對帳戶擁有之 AWS 資源的完整存取權。以下為存取金鑰範例：

- 存取金鑰 ID (20 個字元的英數字串)。例如：AKIAIOSFODNN7EXAMPLE
- 私密存取金鑰 (40 個字元的字串)。例如：wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

存取金鑰 ID 可找出獨一的 AWS 帳戶。您可以使用這些存取金鑰，將驗證請求傳送至 Amazon S3 。

IAM 使用者存取金鑰

您可以為公司建立一個 AWS 帳戶，但該組織內可能有數名員工需要存取組織的 AWS 資源。分享您的 AWS 帳戶 存取金鑰會降低安全性，而為每名員工建立個別的 AWS 帳戶 又不實際。而且您也無法輕易地分享像是儲存貯體與物件等資源，因為它們皆屬不同帳戶所擁有。若要分享資源，您必須授予許可，這是額外進行的工作。

在這類案例中，您可以使用 AWS Identity and Access Management (IAM) 在您的 AWS 帳戶 下，以其本身的存取金鑰建立使用者，並為使用者連接授予適當資源存取許可的 IAM 使用者政策。為加強管理這些使用者，您可利用 IAM 建立使用者群組，並授予會套用到該群組內所有使用者的群組層級許可。

這些使用者稱為 IAM 使用者，而您可於 內建立及管理AWS 父帳戶能控制使用者存取 的能力AWS IAM 使用者建立的任何資源，都由父 AWS 帳戶 控制，且由其付費。這些 IAM 使用者會使用自己的安全登入資料，將經過身分驗證的請求傳送至 Amazon S3。如需建立及管理 AWS 帳戶 下使用者的詳細資訊，請前往 AWS Identity and Access Management 產品詳細資訊頁面。

暫時安全登入資料

除了以使用者自己的存取金鑰建立 IAM 使用者之外，您也可利用 IAM 將暫時性安全登入資料 (暫時性存取金鑰與安全權杖) 授予任何 IAM 使用者，讓他們可以存取您的 AWS 服務與資。您也可以在 AWS 之外，於自己的系統中管理使用者。這些稱為聯合身分使用者。此外，使用者可以是您建立用來存取 AWS 資源的申請人。

IAM 提供 AWS Security Token Service API，供您請求暫時安全登入資料之用。您可以使用 AWS STS API 或 AWS 開發套件，申請這些登入資料。API 會傳回暫時性安全登入資料 (存取金鑰 ID 與私密存取金鑰) 及安全權杖。這些登入資料只有在您申請時所指定的期間內才有效。使用存取金鑰 ID 和私密金鑰的方式，與您利用 AWS 帳戶 或 IAM 使用者存取金鑰來傳送請求時的使用方式一樣。此外，傳送至 Amazon S3 的每個請求中都必要有權杖。

IAM 使用者可以申請這些暫時性安全登入資料供其本身使用，或將它們提供給聯合身分使用者或應用程式。為聯合身分使用者申請暫時性安全登入資料時，您必須提供使用者名稱且定義您希望與這些暫時性安全登入資料相關聯性之許可的 IAM 原則。聯合身分使用者取得的許可數目，不得超過已申請暫時性登入資料的父 IAM 使用者。

您可使用這些暫時性安全登入資料，對 Amazon S3 提出請求。API 程式庫會使用這些登入資料來運算出必要的簽章值，以驗證您的要求。若使用過期的登入資料傳送請求，Amazon S3 會拒絕該請求。

如需使用 REST API 要求中暫時性安全登入資料來簽署要求的資訊，請參閱「簽署與驗證 REST 要求」。如需使用 AWS 開發套件傳送請求的資訊，請參閱「使用 AWS 開發套件提出請求」。

如需 IAM 的臨時安全登入資料支援的詳細資訊，請參閱《IAM 使用者指南》中的臨時安全登入資料。

對於新增的安全來說，您可以在存取 Amazon S3 資源時配置儲存貯體原則，請求進行多重驗證 (MFA)。如需相關資訊，請參閱需要 MFA。在您需要 MFA 才可存取 Amazon S3 資源後，可以存取這些資源唯一的方法是提供以 MFA 金鑰建立的暫時性登入資料。如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 多重要素驗證詳細資訊頁面與設定受 MFA 保護的 API 存取權限。

要求端點

您要將 REST 要求傳送到服務的預先定義端點。如需所有I AWS 服務及其對應端點的清單，請參閱《AWS 一般參考》中的區域與端點。

透過 IPv6 向 Amazon S3 提出請求

Amazon Simple Storage Service (Amazon S3) 除了 IPv4 通訊協定之外，還支援使用網際網路通訊協定第 6 版 (IPv6) 存取 S3 儲存貯體的能力。Amazon S3 雙堆疊端點支援透過 IPv6 與 IPv4 的 S3 儲存貯體要求。透過 IPv6 存取 Amazon S3 不另行收費。如需定價的詳細資訊，請參閱 Amazon S3 定價。

主題

- 透過 IPv6 提出請求的入門
- 在 IAM 原則中使用 IPv6 地址

- 測試 IP 地址相容性
- 使用 Amazon S3 雙堆疊端點

透過 IPv6 提出請求的入門

若要透過 IPv6 向 S3 儲存貯體提出請求，您需要使用雙堆疊端點。下節說明如何使用雙堆疊端點透過 IPv6 提出請求。

以下是透過 IPv6 嘗試存取儲存貯體前的須知事項：

- 存取儲存貯體的用戶端與網路必須啟用才能使用 IPv6。
- 虛擬託管式與路徑式請求都支援 IPv6 存取。如需詳細資訊，請參閱 Amazon S3 雙堆疊端點。
- 如果您在 AWS Identity and Access Management (IAM) 使用者或儲存貯體政策中使用來源 IP 位址篩選，則需要更新政策以包含 IPv6 位址範圍。如需詳細資訊，請參閱 在 IAM 原則中使用 IPv6 地址。
- 使用 IPv6 時，伺服器會存取 IPv6 格式的日誌檔案輸出 IP 地址。您需要更新用來剖析 Amazon S3 日誌檔案的現有工具、指令碼與軟體，以便它們可以剖析 IPv6 格式的 `Remote IP` 地址。如需詳細資訊，請參閱「Amazon S3 伺服器存取日誌格式」和「使用伺服器存取記錄記錄要求」。

> **Note**
>
> 如果碰到與日誌檔案中 IPv6 地址相關的問題，請聯絡 AWS Support。

使用雙重堆疊端點透過 IPv6 提出請求

您使用雙重堆疊端點透過 IPv6 提出具癎 Amazon S3 API 呼叫的請求。無論您是透過 IPv6 還是透過 IPv4 存取Amazon S3，Amazon S3 API 操作都會以相同的方式運作。效能也應該是相同。

使用 REST API 時，您會直接存取雙堆疊端點。如需詳細資訊，請參閱 雙堆疊端點。

使用 AWS Command Line Interface (AWS CLI) 和 AWS SDK 時，您可以使用參數或旗標來變更為雙堆疊端點。您也可以直接指定雙堆疊端點來覆寫設定檔中的 Amazon S3 端點。

您可以使用雙堆疊端點，從下列任一位置透過 IPv6 存取儲存貯體：

- 」， AWS CLI請參閱使用雙堆疊端點 AWS CLI。

- AWS 軟體開發套件，請參閱從 AWS 開發套件使用雙堆疊端點。
- REST API，請參閱「使用 REST API 提出雙重堆疊端點要求」。

無法透過 IPv6 提供的功能

目前，透過 IPv6 存取 S3 儲存貯體時，不支援下列功能：從 S3 儲存貯體託管靜態網站。

在 IAM 原則中使用 IPv6 地址

嘗試使用 IPv6 存取儲存貯體之前，您必須確保用於 IP 地址篩選條件的所有 IAM 使用者或 S3 儲存貯體原則，皆已更新包含 IPv6 地址範圍。未針對處理 IPv6 地址更新的IP 地址篩選條件原則，可能會導致用戶端在開始使用 IPv6 時，錯誤遺失或取得儲存貯體的存取許可。如需 IAM 管理存取許可的詳細資訊，請參閱 適用於 Amazon S3 的 Identity and Access Management。

篩選 IP 地址的 IAM 原則使用IP 地址條件運算子。以下儲存貯體原則會使用 IP 地址條件運算子找出允許之 IPv4 地址的 54.240.143.* 範圍。拒絕此範圍外的任何 IP 地址存取儲存貯體 (`examplebucket`)。因為所有的 IPv6 地址都在允許的範圍外，所以此原則可避免 IPv6 地址得以存取 `examplebucket`。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

您可以修改儲存貯體政策的 `Condition` 元素，允許 IPv4 (`54.240.143.0/24`) 與 IPv6 (`2001:DB8:1234:5678::/64`) 地址範圍，如下例所示。您可使用本例所示的同類型 `Condition` 區塊，更新您的 IAM 使用者與儲存貯體原則。

```
       "Condition": {
```

```
         "IpAddress": {
            "aws:SourceIp": [
              "54.240.143.0/24",
               "2001:DB8:1234:5678::/64"
             ]
          }
        }
```

使用 IPv6 之前，您必須更新所有相關的 IAM 使用者與儲存貯體原則，它們會使用 IP 地址篩選條件允許 IPv6 地址範圍。除現有的 IPv4 地址範圍外，建議您也更新 IAM 原則的貴組織 IPv6 地址範圍。如需允許透過 IPv6 與 IPv4 存取的儲存貯體原則範例，請參閱限制特定 IP 地址的存取。

您可以使用 https://console.aws.amazon.com/iam/的 IAM 主控台來檢閱 IAM 使用者原則。如需 IAM 的詳細資訊，請參閱《IAM 使用者指南》。如需有關編輯 S3 儲存貯體政策的資訊，請參閱「使用 Amazon S3 主控台新增儲存貯體政策」。

測試 IP 地址相容性

如果使用 Linux/Unix 或 Mac OS X，您可以使用 `curl` 命令測試能否透過 IPv6 存取雙堆疊端點，如下例所示：

Example

```
curl -v  http://s3.dualstack.us-west-2.amazonaws.com/
```

您會收到類似下例的資訊。如果您是透過 IPv6 連線，則連線的 IP 地址就會是 IPv6 地址。

```
* About to connect() to s3-us-west-2.amazonaws.com port 80 (#0)
*   Trying IPv6 address... connected
* Connected to s3.dualstack.us-west-2.amazonaws.com (IPv6 address) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1 OpenSSL/1.0.1t
 zlib/1.2.3
> Host: s3.dualstack.us-west-2.amazonaws.com
```

如果使用 Microsoft Windows 7 或 Windows 10，您可以使用 `ping` 命令測試能否透過 IPv6 或 IPv4 存取雙堆疊端點，如下例所示。

```
ping ipv6.s3.dualstack.us-west-2.amazonaws.com
```

使用 Amazon S3 雙堆疊端點

Amazon S3 雙堆疊端點支援透過 IPv6 與 IPv4 的 S3 儲存貯體要求。本節說明如何使用雙堆疊端點。

主題

Amazon S3 雙堆疊端點

當您請求雙堆疊端點時，儲存貯體 URL 會解析為 IPv6 或 IPv4 地址。如需如何存取 IPv6 儲存貯體的詳細資訊，請參閱 透過 IPv6 向 Amazon S3 提出請求。

使用 REST API 等同於使用端點名稱 (URI) 直接存取 Amazon S3 端點。您可以使用虛擬代管型或路徑型的端點名稱，透過雙堆疊端點存取 S3 儲存貯體。Amazon S3 僅支援區域雙堆疊端點名稱，亦即，指定的名稱必須包含區域。

請使用下列命名慣例，為雙堆疊虛擬代管型及路徑型的端點名稱命名：

- 虛擬代管型雙堆疊端點：

  bucketname.s3.dualstack.*aws-region*.amazonaws.com

- 路徑型雙堆疊端點：

  s3.dualstack.*aws-region*.amazonaws.com/*bucketname*

如需端點名稱樣式的詳細資訊，請參閱存取及列出 Amazon S3 儲存貯體。如需 Amazon S3 端點的清單，請參閱《AWS 一般參考》中的區域與端點。

> **Important**
>
> 您可以搭配使用 Transfer Acceleration 與雙堆疊端點。如需詳細資訊，請參閱 Amazon S3 Transfer Acceleration 入門。

> **Note**
>
> 用於存取 Amazon S3 (介面 VPC 端點和閘道 VPC 端點) 的兩種 VPC 端點類型不支援雙堆疊。如需 Amazon S3 的 VPC 端點的詳細資訊，請參閱 AWS PrivateLink 適用於 Amazon S3。

使用 AWS Command Line Interface (AWS CLI) 和 AWS SDK 時，您可以使用參數或旗標來變更為雙堆疊端點。您也可以直接指定雙堆疊端點來覆寫設定檔中的 Amazon S3 端點。以下各節說明如何使用來自 AWS CLI 和 AWS SDK 的雙堆疊端點。

使用雙堆疊端點 AWS CLI

本節提供用於向雙堆疊端點發出要求的 AWS CLI 命令範例。如需有關設定的指示 AWS CLI，請參閱使用 AWS CLI 來透過 Amazon S3 進行開發。

您可以在檔案中的設定 AWS Config 檔`true`中`use_dualstack_endpoint`將組態值設定為，以將`s3`和`s3api` AWS CLI 命令發出的所有 Amazon S3 請求導向指定區域的雙堆疊端點。您可以在設定檔或命令中使用 `--region` 選項指定區域。

將雙堆疊端點與配合使用時 AWS CLI，`path`和`virtual`定址樣式均受支援。設定檔中設定的定址樣式控制儲存貯體名稱應包含主機名稱中，或是包含在 URL 中。根據預設，CLI 會盡可能地嘗試使用虛擬樣式，但會在必要時切換回路徑樣式。如需詳細資訊，請參閱 AWS CLI Amazon S3 組態。

您也可以使用命令變更組態，將預設設定檔中的 `use_dualstack_endpoint` 設為 `true`，以及將 `addressing_style` 設為 `virtual`，如下所示。

```
$ aws configure set default.s3.use_dualstack_endpoint true
$ aws configure set default.s3.addressing_style virtual
```

如果您只想將雙堆疊端點用於指定的 AWS CLI 命令 (並非所有命令)，您可以使用下列其中一種方法：

- 您可以將任何 `--endpoint-url` 或 `https://s3.dualstack.`*`aws-region`*`.amazonaws.com` 命令的 `http://s3.dualstack.`*`aws-region`*`.amazonaws.com` 參數設成 `s3` 或 `s3api`，以在個別的命令中使用雙堆疊端點。

  ```
  $ aws s3api list-objects --bucket bucketname --endpoint-url https://s3.dualstack.aws-region.amazonaws.com
  ```

- 您可以在檔案中設定個別的設定 AWS Config 檔。例如，您可以建立一個設定檔，將 `use_dualstack_endpoint` 設為 `true`，再建立另一個設定檔不設定

use_dualstack_endpoint。當您執行命令時，必須依據是否要使用雙堆疊端點來指定所要使用的設定檔。

> **Note**
>
> 使用時， AWS CLI 您目前無法對雙堆疊端點使用傳輸加速。但是，對於的支持 AWS CLI 即將推出。如需詳細資訊，請參閱 使用 AWS CLI。

從 AWS 開發套件使用雙堆疊端點

本節提供如何使用 AWS SDK 存取雙堆疊端點的範例。

AWS SDK for Java 雙堆疊端點範例

下列範例示範如何在使用 AWS SDK for Java建立的 Amazon S3 用戶端狀態下，啟用雙堆疊端點。

如需建立和測試可運作之 Java 範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

public class DualStackEndpoints {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            // Create an Amazon S3 client with dual-stack endpoints enabled.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .withDualstackEnabled(true)
                    .build();

            s3Client.listObjects(bucketName);
```

```
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

如果您在視窗 AWS SDK for Java 上使用，您可能必須設定下列 Java 虛擬機器 (JVM) 屬性：

```
java.net.preferIPv6Addresses=true
```

AWS .NET SDK 雙堆疊端點範例

使用 AWS SDK for .NET 時，您可以使用`AmazonS3Config`類別來啟用雙堆疊端點的使用，如下列範例所示。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DualStackEndpointTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            var config = new AmazonS3Config
            {
                UseDualstackEndpoint = true,
                RegionEndpoint = bucketRegion
```

```
            };
            client = new AmazonS3Client(config);
            Console.WriteLine("Listing objects stored in a bucket");
            ListingObjectsAsync().Wait();
        }

        private static async Task ListingObjectsAsync()
        {
            try
            {
                var request = new ListObjectsV2Request
                {
                    BucketName = bucketName,
                    MaxKeys = 10
                };
                ListObjectsV2Response response;
                do
                {
                    response = await client.ListObjectsV2Async(request);

                    // Process the response.
                    foreach (S3Object entry in response.S3Objects)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }
                    Console.WriteLine("Next Continuation Token: {0}",
 response.NextContinuationToken);
                    request.ContinuationToken = response.NextContinuationToken;
                } while (response.IsTruncated == true);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

如需列出物件的完整 .NET 範例，請參閱「以程式設計方式列出物件索引鍵」。

如需有關設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

從 REST API 使用雙堆疊端點

如需如何使用 REST API 要雙堆疊端點的資訊，請參閱「使用 REST API 提出雙重堆疊端點要求」。

使用 AWS 開發套件提出請求

主題

您可使用 AWS 開發套件或直接在應用程式中呼叫 REST API，將驗證請求傳送至 Amazon S3。AWS 開發套件 API 使用您提供的登入資料，透過運算簽章來驗證身分。若直接在應用程式中使用 REST API，您必須撰寫必要的程式碼，來運算簽章以驗證您的要求。如需可用的 AWS 開發套件清單，請前往範本程式碼與程式庫。

使用 AWS 帳戶 或 IAM 使用者登入資料提出請求

您可以使用 AWS 帳戶 或 IAM 使用者安全登入資料將經過驗證的請求傳送到 Amazon S3。本節提供如何使用 AWS SDK for Java、 AWS SDK for .NET和傳送已驗證要求的範例 AWS SDK for PHP。如需可用 AWS SDK 的清單，請前往範例程式碼和程式庫。

這些 AWS SDK 中的每一個都使用 SDK 特定的認證提供者鏈結來尋找和使用認證，並代表認證擁有者執行動作。所有這些憑據提供程序鏈的共同點是它們都會查找您的本地 AWS 憑據文件。

如需詳細資訊，請參閱下列主題。

主題

建立本機 AWS 認證檔

為 AWS SDK 配置憑據的最簡單方法是使用 AWS 憑據文件。如果您使用 AWS Command Line Interface (AWS CLI)，您可能已經設定了本機 AWS 認證檔案。若非如此，請使用下列程序來設定登入資料檔案：

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。

2. 建立新的使用者，使其許可受限在希望程式碼可以存取的服務與動作。如需建立新使用者的詳細資訊，請參閱建立 IAM 使用者 (主控台)，並遵循直到步驟 8 的指示執行作業。
3. 選擇 Download .csv (下載 .csv) 並在本機儲存 AWS 登入資料複本。
4. 在您的電腦上，瀏覽至主目錄並建立 `.aws` 目錄。在 Unix 系統上 (例如 Linux 或 OS X)，其會是下列位置：

   ```
   ~/.aws
   ```

   在 Windows 上，其會是下列位置：

   ```
   %HOMEPATH%\.aws
   ```

5. 在 `.aws` 目錄中，建立稱為 `credentials` 的新檔案。
6. 開啟您從 IAM 主控台下載的憑證 `.csv` 檔案，並使用下列格式將其內容複製到 `credentials` 檔案：

   ```
   [default]
   aws_access_key_id = your_access_key_id
   aws_secret_access_key = your_secret_access_key
   ```

7. 儲存 `credentials` 檔案，然後刪除在步驟 3 中所下載的 `.csv` 檔案。

您的共用認證檔案現在已設定在您的本機電腦上，並可與 AWS SDK 搭配使用。

使用 AWS SDK 傳送驗證的要求

使用 AWS SDK 傳送驗證的要求。如需傳送已驗證請求的詳細資訊，請參閱 AWS 安全憑證或 IAM Identity Center 驗證。

Java

若要使用您的 AWS 帳戶 或 IAM 使用者登入資料將經過驗證的請求傳送至 Amazon S3，請執行下列動作：

- 使用 `AmazonS3ClientBuilder` 類別建立 `AmazonS3Client` 執行個體。
- 執行其中一個 `AmazonS3Client` 方法，將請求傳送至 Amazon S3。用戶端會從您所提供的登入資料產生必要的簽章，並在要求中包含此簽章。

下列範例會執行前述作業：如需有關建立和測試工作範例的資訊，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

Example

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsRequest;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.s3.model.S3ObjectSummary;

import java.io.IOException;
import java.util.List;

public class MakingRequests {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Get a list of objects in the bucket, two at a time, and
            // print the name and size of each object.
            ListObjectsRequest listRequest = new
 ListObjectsRequest().withBucketName(bucketName).withMaxKeys(2);
            ObjectListing objects = s3Client.listObjects(listRequest);
            while (true) {
                List<S3ObjectSummary> summaries = objects.getObjectSummaries();
                for (S3ObjectSummary summary : summaries) {
                    System.out.printf("Object \"%s\" retrieved with size %d\n",
 summary.getKey(), summary.getSize());
                }
                if (objects.isTruncated()) {
                    objects = s3Client.listNextBatchOfObjects(objects);
```

```
                } else {
                    break;
                }
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

若要使用您的 AWS 帳戶 或 IAM 使用者登入資料傳送驗證的請求：

- 建立 `AmazonS3Client` 類別的執行個體。
- 執行其中一個 `AmazonS3Client` 方法，將請求傳送至 Amazon S3。用戶端會從您所提供的登入資料產生必要的簽章，並在傳送給 Amazon S3 的請求中包含此簽章。

如需詳細資訊，請參閱「使用 AWS 帳戶 或 IAM 使用者登入資料提出請求」。

> **Note**
>
> - 您可以建立 `AmazonS3Client` 用戶端，而無須提供您的安全登入資料。使用此用戶端傳送的要求，是沒有簽章的匿名要求。若針對非公有存取的資源傳送匿名要求，Amazon S3 會傳回錯誤。
> - 您可以建立 AWS 帳戶 並建立必要的使用者。您也可以管理這些使用者的登入資料。您需要這些登入資料才能執行下列範例中的任務。如需詳細資訊，請參閱《AWS SDK for .NET 開發人員指南》中的設定 AWS 憑證。
>
>   然後，您也可以將應用程式設定為主動擷取設定檔和認證，然後在建立 AWS 服務用戶端時明確使用這些認證。如需詳細資訊，請參閱《AWS SDK for .NET 開發人員指南》中的在應用程式中存取登入資料和設定檔。

下列 C# 範例會說明如何執行前述作業。如需設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class MakeS3RequestTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            using (client = new AmazonS3Client(bucketRegion))
            {
                Console.WriteLine("Listing objects stored in a bucket");
                ListingObjectsAsync().Wait();
            }
        }

        static async Task ListingObjectsAsync()
        {
            try
            {
                ListObjectsRequest request = new ListObjectsRequest
                {
                    BucketName = bucketName,
                    MaxKeys = 2
                };
                do
                {
                    ListObjectsResponse response = await
 client.ListObjectsAsync(request);
                    // Process the response.
```

```
                    foreach (S3Object entry in response.S3Objects)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }

                    // If the response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.Marker = response.NextMarker;
                    }
                    else
                    {
                        request = null;
                    }
                } while (request != null);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

如需工作範例，請參閱「Amazon S3 物件概觀」與「儲存貯體概觀」。您可以使用您 AWS 帳戶或 IAM 使用者登入資料來測試這些範例。

例如，若要列出儲存貯體中的所有物件金鑰，請參閱「以程式設計方式列出物件索引鍵」。

PHP

本節說明如何使用版本 3 中的類別，使用您的 AWS SDK for PHP AWS 帳戶 或 IAM 使用者登入資料傳送已驗證的請求。有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

下列 PHP 範例說明用戶端如何使用您的安全登入資料提出要求，以列出您帳戶的所有儲存貯體。

Example

```
require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
]);

// Retrieve the list of buckets.
$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // Print the list of objects to the page.
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Note

您可以建立 S3Client 用戶端，而無須提供您的安全登入資料。使用此用戶端傳送的要求，是沒有簽章的匿名要求。若針對非公有存取的資源傳送匿名要求，Amazon S3 會傳回錯誤。如需詳細資訊，請參閱 AWS SDK for PHP 文件中的建立匿名用戶端。

如需運作範例，請參閱「Amazon S3 物件概觀」。您可以使用 AWS 帳戶 或 IAM 使用者登入資料來測試這些範例。

如需列出儲存貯體中物件金鑰的範例，請參閱「以程式設計方式列出物件索引鍵」。

Ruby

您必須先設定 SDK 用 AWS SDK for Ruby 來驗證儲存貯體和物件存取權限的 AWS 存取登入資料，才能使用的第 3 版進行呼叫 Amazon S3。如果您在本機系統的認證設定檔中設定了共用 AWS 認證，Ruby 適用的 SDK 第 3 版就可以使用這些認證，而您不需要在程式碼中宣告這些認證。如需設定分享之登入資料的詳細資訊，請參閱「使用 AWS 帳戶 或 IAM 使用者登入資料提出請求」。

下列 Ruby 程式碼片段會使用本機電腦上共用 AWS 認證檔案中的認證來驗證要求，以取得特定值區中所有物件金鑰名稱。會執行以下項目：

1. 建立 `Aws::S3::Client` 類別的執行個體。
2. 使用 `list_objects_v2` 的 `Aws::S3::Client` 方法列舉儲存貯體中的物件，對 Amazon S3 提出要求。用戶端會從電腦上登入資料檔案中的登 AWS 入資料產生必要的簽章值，並將其包含在傳送至 Amazon S3 的請求中。
3. 將物件金鑰名稱陣列，列印至終端機。

Example

```
# Prerequisites:
#   - An existing Amazon S3 bucket.

require "aws-sdk-s3"

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'doc-example-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )
```

```
  if objects.count.positive?
    puts "The object keys in this bucket are (first 50 objects):"
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts "No objects found in this bucket."
  end

  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  return false
end

# Example usage:
def run_me
  region = "us-west-2"
  bucket_name = "BUCKET_NAME"
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

如果您沒有本機 AWS 登入資料檔案，您仍然可以建立`Aws::S3::Client`資源並針對 Amazon S3 儲存貯體和物件執行程式碼。使用適用於 Ruby 的開發套件第 3 版傳送請求，預設是沒有簽章的匿名請求。若針對非公有存取的資源傳送匿名請求，Amazon S3 會傳回錯誤。

您可以針對適用於 Ruby 的開發套件應用程式使用並展開上述程式碼片段，如下列更強大的範例所示。

```
# Prerequisites:
#  - An existing Amazon S3 bucket.

require "aws-sdk-s3"

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
```

```
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'doc-example-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts "The object keys in this bucket are (first 50 objects):"
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts "No objects found in this bucket."
  end

  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  return false
end

# Example usage:
def run_me
  region = "us-west-2"
  bucket_name = "BUCKET_NAME"
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

Go

Example

下列範例會使用 SDK for Go 從共用 AWS 認證檔案自動載入的認證。

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 s3Client := s3.NewFromConfig(sdkConfig)
 count := 10
 fmt.Printf("Let's list up to %v buckets for your account.\n", count)
 result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
 if err != nil {
  fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Buckets) == 0 {
  fmt.Println("You don't have any buckets!")
 } else {
  if count > len(result.Buckets) {
   count = len(result.Buckets)
  }
  for _, bucket := range result.Buckets[:count] {
   fmt.Printf("\t%v\n", *bucket.Name)
  }
 }
}
```

相關資源

- 使用開發 AWS 套件使用 Amazon S3 進行開發
- AWS SDK for PHP 適用於 Amazon S3 Aws\S3\S3Client 類別
- AWS SDK for PHP 文件

使用 IAM 使用者暫時性登入資料提出請求

AWS 帳戶 或 IAM 使用者可以請求臨時安全登入資料，並使用它們將經過驗證的請求傳送到 Amazon S3。本節提供的範例是如何使用 AWS SDK for Java、.NET 與 PHP 來取得暫時性安全登入資料，然後使用這些登入資料來為對 Amazon S3 的請求進行驗證。

Java

IAM 使用者或使用者 AWS 帳戶 可以請求臨時安全登入資料 (請參閱提出要求)， AWS SDK for Java 並使用它們來存取 Amazon S3。這些登入資料會在指定的工作階段使用期限之後到期。

工作階段使用期限預設是一小時。如果您使用 IAM 使用者登入資料，可以在請求暫時性安全登入資料時，為角色指定持續時間，從 15 分鐘到最大的工作階段持續時間。如需暫時安全登入資料的詳細資訊，請參閱《IAM 使用者指南》中的暫時安全登入資料。如需提出請求的詳細資訊，請參閱提出要求。

取得暫時性安全登入資料並存取 Amazon S3

1. 建立 `AWSSecurityTokenService` 類別的執行個體。如需提供登入資料的資訊，請參閱「使用開發 AWS 套件使用 Amazon S3 進行開發」。
2. 呼叫安全性字符服務 (STS) 用戶端的 `assumeRole()` 方法，擷取所需角色的臨時安全登入資料。
3. 將暫時安全登入資料封裝至 `BasicSessionCredentials` 物件。您可以使用此物件，將暫時性安全登入資料提供給 Amazon S3 用戶端。
4. 使用暫時性安全登入資料建立 `AmazonS3Client` 類別的執行個體。您可以使用此用戶端，將請求傳送給 Amazon S3。如果使用過期的登入資料傳送請求，Amazon S3 將會傳回錯誤。

> **Note**
>
> 如果您使用 AWS 帳戶 安全登入資料來取得暫時性安全登入資料，則暫時性登入資料只會作用一個小時。只有在您使用 IAM 使用者登入資料來請求工作階段時，才能指定工作階段使用期限。

下列範例會列出所指定儲存貯體中的一組物件金鑰。此範例會取得工作階段的臨時安全登入資料，然後使用這些登入資料將透過身分驗證的請求傳送給 Amazon S3。

如果您想要使用 IAM 使用者憑證來測試範例，則必須透過 AWS 帳戶建立 IAM 使用者。如需如何建立 IAM 使用者的詳細資訊，請參閱《IAM 使用者指南》中的建立第一個 IAM 使用者與管理員群組。

如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.AssumeRoleRequest;
import com.amazonaws.services.securitytoken.model.AssumeRoleResult;
import com.amazonaws.services.securitytoken.model.Credentials;

public class MakingRequestsWithIAMTempCredentials {
    public static void main(String[] args) {
        String clientRegion = "*** Client region ***";
        String roleARN = "*** ARN for role to be assumed ***";
        String roleSessionName = "*** Role session name ***";
        String bucketName = "*** Bucket name ***";

        try {
            // Creating the STS client is part of your trusted code. It has
            // the security credentials you use to obtain temporary security
 credentials.
            AWSSecurityTokenService stsClient =
 AWSSecurityTokenServiceClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Obtain credentials for the IAM role. Note that you cannot assume the
 role of
            // an AWS root account;
            // Amazon S3 will deny access. You must use credentials for an IAM user
 or an
```

```
            // IAM role.
            AssumeRoleRequest roleRequest = new AssumeRoleRequest()
                    .withRoleArn(roleARN)
                    .withRoleSessionName(roleSessionName);
            AssumeRoleResult roleResponse = stsClient.assumeRole(roleRequest);
            Credentials sessionCredentials = roleResponse.getCredentials();

            // Create a BasicSessionCredentials object that contains the credentials
 you
            // just retrieved.
            BasicSessionCredentials awsCredentials = new BasicSessionCredentials(
                    sessionCredentials.getAccessKeyId(),
                    sessionCredentials.getSecretAccessKey(),
                    sessionCredentials.getSessionToken());

            // Provide temporary security credentials so that the Amazon S3 client
            // can send authenticated requests to Amazon S3. You create the client
            // using the sessionCredentials object.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new
 AWSStaticCredentialsProvider(awsCredentials))
                    .withRegion(clientRegion)
                    .build();

            // Verify that assuming the role worked and the permissions are set
 correctly
            // by getting a set of object keys from the bucket.
            ObjectListing objects = s3Client.listObjects(bucketName);
            System.out.println("No. of Objects: " +
 objects.getObjectSummaries().size());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

IAM 使用者或 AWS 帳戶 可以使用要求臨時安全登入資料， AWS SDK for .NET 並使用它們來存取 Amazon S3。這些登入資料會在工作階段使用期限之後到期。

工作階段使用期限預設是一小時。如果您使用 IAM 使用者登入資料，可以在請求暫時性安全登入資料時，為角色指定持續時間，從 15 分鐘到最大的工作階段持續時間。如需暫時安全登入資料的詳細資訊，請參閱《IAM 使用者指南》中的暫時安全登入資料。如需提出請求的詳細資訊，請參閱提出要求。

取得暫時性安全登入資料並存取 Amazon S3

1. 建立 AWS Security Token Service 用戶端的執行個體`AmazonSecurityTokenServiceClient`。如需提供登入資料的資訊，請參閱「使用開發 AWS 套件使用 Amazon S3 進行開發」。
2. 呼叫您在上一個步驟中所建立之 STS 用戶端的 `GetSessionToken` 方法，以啟動工作階段。您可以使用 `GetSessionTokenRequest` 物件，將工作階段資訊提供給此方法。

   此方法會傳回暫時性安全登入資料。
3. 將暫時安全登入資料封裝至 `SessionAWSCredentials` 物件執行個體。您可以使用此物件，將暫時性安全登入資料提供給 Amazon S3 用戶端。
4. 傳入暫時性安全登入資料，以建立 `AmazonS3Client` 類別執行個體。您可以使用此用戶端，將請求傳送給 Amazon S3。如果使用過期的登入資料傳送請求，Amazon S3 會傳回錯誤。

> **Note**
>
> 如果您使用 AWS 帳戶 安全登入資料來取得暫時性安全登入資料，則該些登入資料只會作用一個小時。您只能在使用 IAM 使用者登入資料請求工作階段時，指定工作階段使用期限。

下列 C# 範例會列出指定儲存貯體中的物件金鑰。為了方便說明，該範例會取得預設一小時工作階段的暫時性安全登入資料，然後使用這些登入資料將透過身分驗證的請求傳送給 Amazon S3。

如果您想要使用 IAM 使用者憑證來測試範例，則必須透過 AWS 帳戶建立 IAM 使用者。如需如何建立 IAM 使用者的詳細資訊，請參閱《IAM 使用者指南》中的建立第一個 IAM 使用者與管理員群組。如需提出請求的詳細資訊，請參閱提出要求。

如需有關設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempCredExplicitSessionStartTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {
            try
            {
                // Credentials use the default AWS SDK for .NET credential search
 chain.
                // On local development machines, this is your default profile.
                Console.WriteLine("Listing objects stored in a bucket");
                SessionAWSCredentials tempCredentials = await
 GetTemporaryCredentialsAsync();

                // Create a client by providing temporary security credentials.
                using (s3Client = new AmazonS3Client(tempCredentials, bucketRegion))
                {
                    var listObjectRequest = new ListObjectsRequest
                    {
```

```
                        BucketName = bucketName
                    };
                    // Send request to Amazon S3.
                    ListObjectsResponse response = await
s3Client.ListObjectsAsync(listObjectRequest);
                    List<S3Object> objects = response.S3Objects;
                    Console.WriteLine("Object count = {0}", objects.Count);
                }
            }
            catch (AmazonS3Exception s3Exception)
            {
                Console.WriteLine(s3Exception.Message, s3Exception.InnerException);
            }
            catch (AmazonSecurityTokenServiceException stsException)
            {
                Console.WriteLine(stsException.Message,
stsException.InnerException);
            }
        }

        private static async Task<SessionAWSCredentials>
GetTemporaryCredentialsAsync()
        {
            using (var stsClient = new AmazonSecurityTokenServiceClient())
            {
                var getSessionTokenRequest = new GetSessionTokenRequest
                {
                    DurationSeconds = 7200 // seconds
                };

                GetSessionTokenResponse sessionTokenResponse =
                            await
stsClient.GetSessionTokenAsync(getSessionTokenRequest);

                Credentials credentials = sessionTokenResponse.Credentials;

                var sessionCredentials =
                    new SessionAWSCredentials(credentials.AccessKeyId,
                                              credentials.SecretAccessKey,
                                              credentials.SessionToken);
                return sessionCredentials;
            }
        }
    }
```

```
}
```

PHP

有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

IAM 使用者或 AWS 帳戶 可以使用的第 3 版要求臨時安全登入資料 AWS SDK for PHP。然後，可以使用該暫時性登入資料來存取 Amazon S3。該登入資料會在工作階段使用期限到期時到期。

工作階段使用期限預設是一小時。如果您使用 IAM 使用者登入資料，可以在請求暫時性安全登入資料時，為角色指定持續時間，從 15 分鐘到最大的工作階段持續時間。如需暫時安全登入資料的詳細資訊，請參閱《IAM 使用者指南》中的暫時安全登入資料。如需提出請求的詳細資訊，請參閱提出要求。

> **Note**
>
> 如果您使用 AWS 帳戶 安全登入資料來取得暫時性安全登入資料，則暫時性安全登入資料只會作用一個小時。只有在您使用 IAM 使用者登入資料來請求工作階段時，才能指定工作階段使用期限。

Example

下列 PHP 範例會使用暫時性安全登入資料來列出指定儲存貯體中的物件金鑰。此範例會取得預設一小時工作階段的暫時性安全登入資料，然後使用這些登入資料將透過身分驗證的請求傳送給 Amazon S3。有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

如果您想要使用 IAM 使用者憑證來測試範例，則需要透過 AWS 帳戶建立 IAM 使用者。如需如何建立 IAM 使用者的資訊，請參閱《IAM 使用者指南》中的建立第一個 IAM 使用者與管理員群組。如需使用 IAM 使用者登入資料要求工作階段時設定工作階段使用期限的範例，請參閱「使用 IAM 使用者暫時性登入資料提出請求」。

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';
```

```
$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

$sessionToken = $sts->getSessionToken();

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key'    => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token'  => $sessionToken['Credentials']['SessionToken']
    ]
]);

$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // List objects
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

IAM 使用者或使用者 AWS 帳戶 可以請求臨時安全登入資料， AWS SDK for Ruby 並使用它們來存取 Amazon S3。這些登入資料會在工作階段使用期限之後到期。

工作階段使用期限預設是一小時。如果您使用 IAM 使用者登入資料，可以在請求暫時性安全登入資料時，為角色指定持續時間，從 15 分鐘到最大的工作階段持續時間。如需暫時安全登入資料的詳

細資訊，請參閱《IAM 使用者指南》中的暫時安全登入資料。如需提出請求的詳細資訊，請參閱提出要求。

> **Note**
>
> 如果您使用 AWS 帳戶 安全登入資料來取得暫時性安全登入資料，則暫時性安全登入資料只會作用一個小時。只有在您使用 IAM 使用者登入資料請求工作階段時，才能指定工作階段使用期限。

下列 Ruby 範例會建立暫時使用者有一小時的時間可列出指定儲存貯體中的項目。若要使用此範例，您必須擁有具有建立新 AWS Security Token Service (AWS STS) 用戶端所需許可的 AWS 登入資料，並列出 Amazon S3 儲存貯體。

```
# Prerequisites:
# - A user in AWS Identity and Access Management (IAM). This user must
#   be able to assume the following IAM role. You must run this code example
#   within the context of this user.
# - An existing role in IAM that allows all of the Amazon S3 actions for all of the
#   resources in this code example. This role must also trust the preceding IAM
 user.
# - An existing S3 bucket.

require "aws-sdk-core"
require "aws-sdk-s3"
require "aws-sdk-iam"

# Checks whether a user exists in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Boolean] true if the user exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless user_exists?(iam_client, 'my-user')
def user_exists?(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  return true if response.user.user_name
rescue Aws::IAM::Errors::NoSuchEntity
  # User doesn't exist.
rescue StandardError => e
```

```
  puts "Error while determining whether the user " \
    "'#{user_name}' exists: #{e.message}"
end

# Creates a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS:IAM::Types::User] The new user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = create_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def create_user(iam_client, user_name)
  response = iam_client.create_user(user_name: user_name)
  return response.user
rescue StandardError => e
  puts "Error while creating the user '#{user_name}': #{e.message}"
end

# Gets a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS:IAM::Types::User] The existing user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def get_user(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  return response.user
rescue StandardError => e
  puts "Error while getting the user '#{user_name}': #{e.message}"
end

# Checks whether a role exists in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The role's name.
# @return [Boolean] true if the role exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless role_exists?(iam_client, 'my-role')
```

```
def role_exists?(iam_client, role_name)
  response = iam_client.get_role(role_name: role_name)
  return true if response.role.role_name
rescue StandardError => e
  puts "Error while determining whether the role " \
    "'#{role_name}' exists: #{e.message}"
end

# Gets credentials for a role in IAM.
#
# @param sts_client [Aws::STS::Client] An initialized AWS STS client.
# @param role_arn [String] The role's Amazon Resource Name (ARN).
# @param role_session_name [String] A name for this role's session.
# @param duration_seconds [Integer] The number of seconds this session is valid.
# @return [AWS::AssumeRoleCredentials] The credentials.
# @example
#   sts_client = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_credentials(
#     sts_client,
#     'arn:aws:iam::123456789012:role/AmazonS3ReadOnly',
#     'ReadAmazonS3Bucket',
#     3600
#   )
#   exit 1 if credentials.nil?
def get_credentials(sts_client, role_arn, role_session_name, duration_seconds)
  Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: role_session_name,
    duration_seconds: duration_seconds
  )
rescue StandardError => e
  puts "Error while getting credentials: #{e.message}"
end

# Checks whether a bucket exists in Amazon S3.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The name of the bucket.
# @return [Boolean] true if the bucket exists; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless bucket_exists?(s3_client, 'doc-example-bucket')
def bucket_exists?(s3_client, bucket_name)
```

```
  response = s3_client.list_buckets
  response.buckets.each do |bucket|
    return true if bucket.name == bucket_name
  end
rescue StandardError => e
  puts "Error while checking whether the bucket '#{bucket_name}' " \
    "exists: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'doc-example-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts "Name => ETag"
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end
```

相關資源

- 使用開發 AWS 套件使用 Amazon S3 進行開發
- AWS SDK for PHP 對於 Amazon S3 AWS\ S3\ S3 客戶端類

- AWS SDK for PHP 文件

使用聯合身分使用者暫時登入資料提出要求

您可以要求臨時安全登入資料，並將其提供給需要存取 AWS 資源的同盟使用者或應用程式。本節提供如何使用 AWS SDK 取得聯合身分使用者或應用程式的臨時安全登入資料，以及如何使用這些登入資料將經過驗證的請求傳送至 Amazon S3 的範例。如需可用 AWS SDK 的清單，請參閱範例程式碼和程式庫。

> **Note**
>
> IAM 使用者 AWS 帳戶 和 IAM 使用者都可以要求聯合身分使用者的臨時安全登入資料。不過，為了提高安全，只有具有必要許可的 IAM 使用者才應該請求這些暫時登入資料，確保聯合身分使用者善加利用請求 IAM 使用者的許可。在一些應用程式中，您可能會發現一些適合建立 IAM 的使用者，其具有唯一目的為將暫時安全登入資料授予聯合身分使用者與應用程式的特定許可。

Java

您可以為同盟使用者和應用程式提供臨時安全登入資料，以便他們傳送經過驗證的要求以存取您的 AWS 資源。請求這些暫時登入資料時，您必須提供使用者名稱，以及說明您想要授予之資源許可的 IAM 原則。工作階段使用期限預設是一小時。要求聯合身分使用者與應用程式的暫時安全登入資料時，您可以明確地設定不同的持續時間值。

> **Note**
>
> 在請求用於聯合身分使用者與應用程式得暫時安全登入資料時，為了提高安全，我們建議您使用專用 IAM 使用者，只進行有必要的存取許可。您所建立暫時使用者的許可絕不可高於已請求暫時安全登入資料的 IAM 使用者。如需詳細資訊，請參閱 AWS Identity and Access Management 常見問答集。

為提供安全登入資料和傳送經過驗證的要求以存取資源，請執行以下動作：

- 建立 `AWSSecurityTokenServiceClient` 類別的執行個體。
- 透過呼叫安全字符服務 (STS) 用戶端的 `getFederationToken()` 方法，開始工作階段。請您提供工作階段資訊，包括您要連線至暫時性登入資料的使用者名稱與 IAM 原則。您可以提供選用的工作階段持續時間。此方法會傳回暫時安全登入資料。

- 將暫時安全登入資料封裝至 BasicSessionCredentials 物件執行個體。您可以使用此物件，將暫時性安全登入資料提供給 Amazon S3 用戶端。
- 使用暫時性安全登入資料建立 AmazonS3Client 類別的執行個體。您可以使用此用戶端，將請求傳送給 Amazon S3。如果使用過期的登入資料傳送請求，Amazon S3 會傳回錯誤。

Example

此範例會列出指定 S3 儲存貯體中的金鑰。在範例中，您會先為聯合身分使用者，取得兩個小時的暫時安全登入資料工作階段，並使用它們將經過身分驗證的請求傳送給 Amazon S3。若要執行此範例，您需要建立具有附加政策的 IAM 使用者，以允許使用者要求臨時安全登入資料並列出您的 AWS 資源。以下政策可以完成這個情況：

```
{
  "Statement":[{
      "Action":["s3:ListBucket",
        "sts:GetFederationToken*"
      ],
      "Effect":"Allow",
      "Resource":"*"
    }
  ]
}
```

如需如何建立 IAM 使用者的詳細資訊，請參閱《IAM 使用者指南》中的建立第一個 IAM 使用者與管理員群組。

在建立 IAM 使用者和附加先前的原則，您可以執行下列的範例。如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.S3Actions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;

import java.io.IOException;

public class MakingRequestsWithFederatedTempCredentials {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Specify bucket name ***";
        String federatedUser = "*** Federated user name ***";
        String resourceARN = "arn:aws:s3:::" + bucketName;

        try {
            AWSSecurityTokenService stsClient = AWSSecurityTokenServiceClientBuilder
                    .standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            GetFederationTokenRequest getFederationTokenRequest = new
 GetFederationTokenRequest();
            getFederationTokenRequest.setDurationSeconds(7200);
            getFederationTokenRequest.setName(federatedUser);

            // Define the policy and add it to the request.
            Policy policy = new Policy();
            policy.withStatements(new Statement(Effect.Allow)
                    .withActions(S3Actions.ListObjects)
                    .withResources(new Resource(resourceARN)));
            getFederationTokenRequest.setPolicy(policy.toJson());

            // Get the temporary security credentials.
            GetFederationTokenResult federationTokenResult =
 stsClient.getFederationToken(getFederationTokenRequest);
            Credentials sessionCredentials = federationTokenResult.getCredentials();

            // Package the session credentials as a BasicSessionCredentials
```

```
            // object for an Amazon S3 client object to use.
            BasicSessionCredentials basicSessionCredentials = new
 BasicSessionCredentials(
                    sessionCredentials.getAccessKeyId(),
                    sessionCredentials.getSecretAccessKey(),
                    sessionCredentials.getSessionToken());
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new
 AWSStaticCredentialsProvider(basicSessionCredentials))
                    .withRegion(clientRegion)
                    .build();

            // To verify that the client works, send a listObjects request using
            // the temporary security credentials.
            ObjectListing objects = s3Client.listObjects(bucketName);
            System.out.println("No. of Objects = " +
 objects.getObjectSummaries().size());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

您可以為同盟使用者和應用程式提供臨時安全登入資料，以便他們傳送經過驗證的要求以存取您的 AWS 資源。請求這些暫時登入資料時，您必須提供使用者名稱，以及說明您想要授予之資源許可的 IAM 原則。預設的工作階段使用期限為一小時。要求聯合身分使用者與應用程式的暫時安全登入資料時，您可以明確地設定不同的持續時間值。如需有關傳送已驗證要求的詳細資訊，請參閱提出要求。

> **Note**
>
> 當請求用於聯合身分使用者與應用程式的暫時安全登入資料時，為了提高安全，我們建議您使用專用 IAM 使用者，僅進行有必要的存取許可。您所建立暫時使用者的許可絕不可高於

已請求暫時安全登入資料的 IAM 使用者。如需詳細資訊，請參閱 AWS Identity and Access Management 常見問答集。

您可執行下列項目：

- 創建 AWS Security Token Service 客戶端，`AmazonSecurityTokenServiceClient`類的一個實例。
- 呼叫 STS 用戶端的 `GetFederationToken` 方法，以啟動工作階段。您需要提供工作階段資訊，包括您要連線至暫時性登入資料的使用者名稱與 IAM 原則。您可以選擇性提供工作階段持續的時間。此方法會傳回暫時安全登入資料。
- 將暫時安全登入資料封裝至 `SessionAWSCredentials` 物件執行個體。您可以使用此物件，將暫時性安全登入資料提供給 Amazon S3 用戶端。
- 請傳遞暫時性安全登入資料，以建立 `AmazonS3Client` 類別的執行個體。您可以使用此用戶端，傳送請求給 Amazon S3。如果使用過期的登入資料傳送請求，Amazon S3 會傳回錯誤。

Example

下列 C# 範例會列出所指定儲存貯體中的金鑰。在範例中，您會先以聯合身分使用者(User1)，取得兩個小時的暫時安全登入資料工作階段，並使用它們將經過身分驗證的請求傳送給 Amazon S3。

- 在此範例中，請您建立 IAM 使用者，並使其擁有最少許可。若使用 IAM 使用者的登入資料，請您為其他請求暫時性登入資料。此範例僅會列出特定儲存貯體中的此物件。請使用下列附加原則來建立 IAM 使用者：

```
{
  "Statement":[{
      "Action":["s3:ListBucket",
        "sts:GetFederationToken*"
      ],
      "Effect":"Allow",
      "Resource":"*"
    }
  ]
}
```

該政策允許 IAM 使用者請求臨時安全登入資料和存取權限，以列出您的 AWS 資源。如需如何建立 IAM 使用者的詳細資訊，請參閱《IAM 使用者指南》中的建立 IAM 使用者與管理員群組。

- 使用 IAM 使用者安全登入資料來測試下列範例。此範例會使用暫時安全登入資料將經過身分驗證的請求傳送給 Amazon S3。此範例會在要求聯合身分使用者 (使用者 1) 的暫時性安全登入資料時指定下列政策，以限制存取指定儲存貯體 (YourBucketName) 中列出的物件。您必須更新政策，並提供自己的現有儲存貯體名稱。

```
{
  "Statement":[
    {
      "Sid":"1",
      "Action":["s3:ListBucket"],
      "Effect":"Allow",
      "Resource":"arn:aws:s3:::YourBucketName"
    }
  ]
}
```

- Example

  更新下列範例，並提供在上一個聯合身分使用者存取原則中所指定的儲存貯體名稱。如需有關設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。AWS

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempFederatedCredentialsTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
```

```
        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {
            try
            {
                Console.WriteLine("Listing objects stored in a bucket");
                // Credentials use the default AWS SDK for .NET credential search
chain.
                // On local development machines, this is your default profile.
                SessionAWSCredentials tempCredentials =
                    await GetTemporaryFederatedCredentialsAsync();

                // Create a client by providing temporary security credentials.
                using (client = new AmazonS3Client(bucketRegion))
                {
                    ListObjectsRequest listObjectRequest = new
ListObjectsRequest();
                    listObjectRequest.BucketName = bucketName;

                    ListObjectsResponse response = await
client.ListObjectsAsync(listObjectRequest);
                    List<S3Object> objects = response.S3Objects;
                    Console.WriteLine("Object count = {0}", objects.Count);

                    Console.WriteLine("Press any key to continue...");
                    Console.ReadKey();
                }
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}'
when writing an object", e.Message);
            }
        }
```

```
        private static async Task<SessionAWSCredentials>
 GetTemporaryFederatedCredentialsAsync()
        {
            AmazonSecurityTokenServiceConfig config = new
 AmazonSecurityTokenServiceConfig();
            AmazonSecurityTokenServiceClient stsClient =
                new AmazonSecurityTokenServiceClient(
                                            config);

            GetFederationTokenRequest federationTokenRequest =
                                      new GetFederationTokenRequest();
            federationTokenRequest.DurationSeconds = 7200;
            federationTokenRequest.Name = "User1";
            federationTokenRequest.Policy = @"{
               ""Statement"":
               [
                 {
                   ""Sid"":""Stmt1311212314284"",
                   ""Action"":[""s3:ListBucket""],
                   ""Effect"":""Allow"",
                   ""Resource"":""arn:aws:s3:::" + bucketName + @"""
                  }
               ]
             }
            ";

            GetFederationTokenResponse federationTokenResponse =
                        await
 stsClient.GetFederationTokenAsync(federationTokenRequest);
            Credentials credentials = federationTokenResponse.Credentials;

            SessionAWSCredentials sessionCredentials =
                new SessionAWSCredentials(credentials.AccessKeyId,
                                          credentials.SecretAccessKey,
                                          credentials.SessionToken);
            return sessionCredentials;
        }
    }
}
```

PHP

本主題說明如何使用第 3 版的類別為聯合身分使用者和應用程式請求臨時安全登入資料，並使用這些登入資源存取 Amazon S3 中存放的資源。 AWS SDK for PHP 有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

您可以為同盟使用者和應用程式提供臨時安全登入資料，以便他們傳送經過驗證的要求以存取您的 AWS 資源。請求這些暫時登入資料時，您必須提供使用者名稱，以及說明您想要授予之資源許可的 IAM 原則。這些登入資料會在工作階段使用期限到期時到期。工作階段使用期限預設是一小時。在要求用於聯合身分使用者與應用程式的暫時安全登入資料時，您可以明確地為持續時間設定不同的值。如需暫時安全登入資料的詳細資訊，請參閱《IAM 使用者指南》中的暫時安全登入資料。如需有關提供暫時安全登入資料用於聯合身分使用者與應用程式之詳細資料，請參閱 提出要求。

在請求用於聯合身分使用者與應用程式得暫時安全登入資料時，為了提高安全，我們建議請使用專用 IAM 使用者，只進行有必要的存取許可。您所建立暫時使用者的許可絕不可高於已請求暫時安全登入資料的 IAM 使用者。如需關於聯合身分的資訊，請參閱 AWS Identity and Access Management 常見問答集。

有關 Ruby API 的 AWS SDK 的更多信息，請訪問 AWS SDK for Ruby-版本 2。

Example

下列 PHP 範例會列出所指定儲存貯體中的金鑰。在範例中，您會為聯合身分使用者 (User1) 的身分，取得 1 小時的暫時安全登入資料工作階段。然後，請您使用暫時安全登入資料傳送驗證請求給 Amazon S3。

為了提高安全，當請求其他人的暫時登入資料時，使用安全登入資料的 IAM 使用者，有權請求暫時性安全登入資料。為確保 IAM 使用者只授予最有應用程式特定許可給聯合身分使用者，您也可以限制此 IAM 使用者的存取許可。此範例僅會列出特定儲存貯體中的物件。請使用下列附加原則來建立 IAM 使用者：

```
{
  "Statement":[{
      "Action":["s3:ListBucket",
        "sts:GetFederationToken*"
      ],
      "Effect":"Allow",
      "Resource":"*"
    }
  ]
}
```

該政策允許 IAM 使用者請求臨時安全登入資料和存取權限，以列出您的 AWS 資源。如需如何建立 IAM 使用者的詳細資訊，請參閱《IAM 使用者指南》中的建立第一個 IAM 使用者與管理員群組。

您現在可以使用 IAM 使用者安全登入資料來測試下列範例。此範例會使用暫時安全登入資料將經過身分驗證的請求傳送給 Amazon S3。在要求用於聯合身分使用者 (User1) 的暫時安全登入資料時，此範例指定下列政策，以限制列出特定儲存貯體中物件的存取。使用您儲存貯體名稱更新政策。

```
{
  "Statement":[
    {
      "Sid":"1",
      "Action":["s3:ListBucket"],
      "Effect":"Allow",
      "Resource":"arn:aws:s3:::YourBucketName"
    }
  ]
}
```

在下列範例中，您必須在指定政策資源時，將 YourBucketName 取代為您的儲存貯體名稱。

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

// In real applications, the following code is part of your trusted code. It has
// the security credentials that you use to obtain temporary security credentials.
$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

// Fetch the federated credentials.
$sessionToken = $sts->getFederationToken([
    'Name'              => 'User1',
    'DurationSeconds'    => '3600',
    'Policy'            => json_encode([
        'Statement' => [
            'Sid'               => 'randomstatementid' . time(),
```

```
            'Action'           => ['s3:ListBucket'],
            'Effect'           => 'Allow',
            'Resource'         => 'arn:aws:s3:::' . $bucket
        ]
    ])
]);

// The following will be part of your less trusted code. You provide temporary
// security credentials so the code can send authenticated requests to Amazon S3.

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key' => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token' => $sessionToken['Credentials']['SessionToken']
    ]
]);

try {
    $result = $s3->listObjects([
        'Bucket' => $bucket
    ]);
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

您可以為同盟使用者和應用程式提供臨時安全登入資料，以便他們傳送經過驗證的要求以存取您的 AWS 資源。向 IAM 服務請求這些暫時登入資料時，您必須提供使用者名稱，以及說明您想要授予之資源許可的 IAM 原則。工作階段使用期限預設是一小時。不過，如果您要使用 IAM 使用者登入資料來請求暫時登入資料，則可以在請求聯合身分使用者與應用程式的暫時安全登入資料時明確地設定不同的持續時間值。如需更多有關聯合身分使用者的暫時安全登入資料與應用程式，請參閱 提出要求。

> **Note**
>
> 為了提高安全，請求聯合身分使用者與應用程式的暫時安全登入資料，您可能會想要使用只具有必要存取許可的專用 IAM 使用者。您所建立暫時使用者的許可絕不可高於已請

求暫時安全登入資料的 IAM 使用者。如需詳細資訊，請參閱 AWS Identity and Access Management 常見問答集。

Example

下列 Ruby 程式碼範例允許聯合身分使用者具有有限許可，以設定在指定儲存貯體中的金鑰清單。

```
# Prerequisites:
#  - An existing Amazon S3 bucket.

require "aws-sdk-s3"
require "aws-sdk-iam"
require "json"

# Checks to see whether a user exists in IAM; otherwise,
# creates the user.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Aws::IAM::Types::User] The existing or new user.
# @example
#   iam = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam, 'my-user')
#   exit 1 unless user.user_name
#   puts "User's name: #{user.user_name}"
def get_user(iam, user_name)
  puts "Checking for a user with the name '#{user_name}'..."
  response = iam.get_user(user_name: user_name)
  puts "A user with the name '#{user_name}' already exists."
  return response.user
# If the user doesn't exist, create them.
rescue Aws::IAM::Errors::NoSuchEntity
  puts "A user with the name '#{user_name}' doesn't exist. Creating this user..."
  response = iam.create_user(user_name: user_name)
  iam.wait_until(:user_exists, user_name: user_name)
  puts "Created user with the name '#{user_name}'."
  return response.user
rescue StandardError => e
  puts "Error while accessing or creating the user named '#{user_name}':
 #{e.message}"
end
```

```
# Gets temporary AWS credentials for an IAM user with the specified permissions.
#
# @param sts [Aws::STS::Client] An initialized AWS STS client.
# @param duration_seconds [Integer] The number of seconds for valid credentials.
# @param user_name [String] The user's name.
# @param policy [Hash] The access policy.
# @return [Aws::STS::Types::Credentials] AWS credentials for API authentication.
# @example
#   sts = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_temporary_credentials(sts, duration_seconds, user_name,
#     {
#       'Version' => '2012-10-17',
#       'Statement' => [
#         'Sid' => 'Stmt1',
#         'Effect' => 'Allow',
#         'Action' => 's3:ListBucket',
#         'Resource' => 'arn:aws:s3:::doc-example-bucket'
#       ]
#     }
#   )
#   exit 1 unless credentials.access_key_id
#   puts "Access key ID: #{credentials.access_key_id}"
def get_temporary_credentials(sts, duration_seconds, user_name, policy)
  response = sts.get_federation_token(
    duration_seconds: duration_seconds,
    name: user_name,
    policy: policy.to_json
  )
  return response.credentials
rescue StandardError => e
  puts "Error while getting federation token: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'doc-example-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
```

```
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts "Name => ETag"
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end

# Example usage:
def run_me
  region = "us-west-2"
  user_name = "my-user"
  bucket_name = "doc-example-bucket"

  iam = Aws::IAM::Client.new(region: region)
  user = get_user(iam, user_name)

  exit 1 unless user.user_name

  puts "User's name: #{user.user_name}"
  sts = Aws::STS::Client.new(region: region)
  credentials = get_temporary_credentials(sts, 3600, user_name,
    {
      "Version" => "2012-10-17",
      "Statement" => [
        "Sid" => "Stmt1",
        "Effect" => "Allow",
        "Action" => "s3:ListBucket",
        "Resource" => "arn:aws:s3:::#{bucket_name}"
      ]
    }
  )

  exit 1 unless credentials.access_key_id
```

```
  puts "Access key ID: #{credentials.access_key_id}"
  s3_client = Aws::S3::Client.new(region: region, credentials: credentials)

  exit 1 unless list_objects_in_bucket?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

相關資源

- 使用開發 AWS 套件使用 Amazon S3 進行開發
- AWS SDK for PHP 對於 Amazon S3 AWS\ S3\ S3 客戶端類
- AWS SDK for PHP 文件

使用 REST API 提出要求

本節包含如何使用 REST API 向 Amazon S3 端點提出請求的資訊。如需 Amazon S3 端點的清單，請參閱《AWS 一般參考》中的區域與端點。

為 REST API 請求建構 S3 主機名稱

Amazon S3 端點遵循如下所示的結構：

```
s3.Region.amazonaws.com
```

Amazon S3 存取點端點和雙堆疊端點也遵循標準結構：

- Amazon S3 存取點 -`s3-accesspoint.Region.amazonaws.com`
- 雙堆疊 - `s3.dualstack.Region.amazonaws.com`

如需 Amazon S3 區域和端點的完整清單，請參閱《Amazon Web Services 一般參考》中的 Amazon S3 端點和配額。

虛擬主辦式和路徑式請求

使用 REST API 提出請求時，可以使用 Amazon S3 端點的虛擬託管式或路徑式 URI。如需詳細資訊，請參閱「儲存貯體的虛擬託管」。

Example 虛擬託管式請求

下例是從美國西部 (奧勒岡) 區域中名為 puppy.jpg 的儲存貯體中刪除 examplebucket 檔案的虛擬託管式請求。如需虛擬託管型要求的詳細資訊，請參閱「虛擬託管樣式請求」。

```
DELETE /puppy.jpg HTTP/1.1
Host: examplebucket.s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Example 路徑型要求

下例是相同要求的路徑型版本。

```
DELETE /examplebucket/puppy.jpg HTTP/1.1
Host: s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Amazon S3 目前支援所有 AWS 區域 中的虛擬託管樣式 URL 與路徑樣式 URL 存取權。但是，將來會停產路徑樣式 URL。如需詳細資訊，請參閱下列 Important (重要) 注意事項。

如需路徑式請求的詳細資訊，請參閱路徑樣式請求。

> **Important**
>
> 更新 (2020 年 9 月 23 日) – 我們決定將棄用路徑樣式 URL 的日期延後，以確保客戶有足夠時間轉換為虛擬託管樣式 URL。如需詳細資訊，請參閱 AWS 新聞部落格中的 Amazon S3 路徑廢除計畫 - 其他故事。

使用 REST API 提出雙重堆疊端點要求

使用 REST API 時，可以使用虛擬託管型或路徑型端點名稱 (URI)，直接存取雙重堆疊端點。所有 Amazon S3 雙重堆疊端點名稱，在名稱中都包含區域。與標準型僅 IPv4 端點不同，虛擬託管型和路徑型端點都會使用區域專屬的端點名稱。

Example 虛擬託管式雙重堆疊端點請求

您可在 REST 請求中使用虛擬託管式端點 (如下例所示)，從美國西部 (奧勒岡) 區域中名為 `puppy.jpg` 的儲存貯體擷取 `examplebucket` 物件。

```
GET /puppy.jpg HTTP/1.1
Host: examplebucket.s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Example 路徑型雙重堆疊端點要求

或者，您可在要求中使用路徑型端點，如下例所示。

```
GET /examplebucket/puppy.jpg HTTP/1.1
Host: s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

如需雙重堆疊端點的詳細資訊，請參閱「使用 Amazon S3 雙堆疊端點」。

如需有關使用 REST API 提出請求的詳細資訊，請參閱下方的主題。

主題

- 儲存貯體的虛擬託管
- 要求重新導向與 REST API

儲存貯體的虛擬託管

虛擬託管是從單一 Web 伺服器服務多個網站的實務。其中一種區分 Amazon S3 REST API 要求網站的方式是使用要求 URI 的清楚主機名稱，而不只是 URI 的路徑名稱部分。一般 Amazon S3 REST 請求會使用 Request-URI 路徑的第一個斜線區隔元件，來指定儲存貯體。反之，您可以使用 HTTP `Host` 標頭，以使用 Amazon S3 虛擬託管來處理 REST API 呼叫中的儲存貯體。實務上，Amazon S3 會將 `Host` 解譯為表示可在 `https://`*`bucket-name`*`.s3.`*`region-code`*`.amazonaws.com` 自動存取大部分的儲存貯體 (針對有限類型的請求)。如需 Amazon S3 區域和端點的完整清單，請參閱《Amazon Web Services 一般參考》中的 Amazon S3 端點和配額。

虛擬託管也有其他優勢。在註冊的網域名稱後面命名儲存貯體，以及將該名稱設為 Amazon S3 的 DNS 別名，即可完全自訂 Amazon S3 資源的 URL (例如，`http://my.bucket-name.com/`)。您也可以發佈到儲存貯體虛擬伺服器的「根目錄」。這項能力十分重要，因為許多現有應用程式都會在此標準位置中搜尋檔案。例如，預期都可以在根目錄中找到 `favicon.ico`、`robots.txt` 和 `crossdomain.xml`。

> **Important**
>
> 當您搭配使用虛擬託管型儲存貯體與 SSL 時，SSL 萬用字元憑證只會符合不包含點 (.) 的儲存貯體。若要解決此限制，請使用 HTTP 或撰寫您自己的憑證驗證邏輯。如需詳細資訊，請參閱AWS 新聞部落格上的 Amazon S3 路徑取代計劃。

主題

- 路徑樣式請求
- 虛擬託管樣式請求
- HTTP Host 標頭儲存貯體規格
- 範例
- 使用 CNAME 記錄自訂 Amazon S3 URL
- 如何建立主機名稱與 Amazon S3 儲存貯體的關聯
- 限制
- 回溯相容性

路徑樣式請求

Amazon S3 目前支援所有 AWS 區域 中的虛擬託管樣式 URL 與路徑樣式 URL 存取。但是，將來會停產路徑樣式 URL。如需詳細資訊，請參閱下列 Important (重要) 注意事項。

在 Amazon S3 中，路徑樣式 URL 使用以下格式：

```
https://s3.region-code.amazonaws.com/bucket-name/key-name
```

例如，如果您在美國西部 (奧勒岡) 區域中建立名為 `DOC-EXAMPLE-BUCKET1` 的儲存貯體，且需要存取該儲存貯體中的 `puppy.jpg` 物件，則可使用以下路徑型 URL：

```
https://s3.us-west-2.amazonaws.com/DOC-EXAMPLE-BUCKET1/puppy.jpg
```

> **Important**
>
> 更新 (2020 年 9 月 23 日) – 我們決定將棄用路徑樣式 URL 的日期延後，以確保客戶有足夠時間轉換為虛擬託管樣式 URL。如需詳細資訊，請參閱 AWS 新聞部落格中的 Amazon S3 路徑廢除計畫 - 其他故事。

> **Warning**
>
> 託管將從 Web 瀏覽器存取的網站內容時，請避免使用路徑樣式 URL，這可能會干擾瀏覽器相同來源的安全模型。若要託管網站內容，建議您使用 S3 網站端點或 CloudFront 分發。如需詳細資訊，請參閱《AWS 規定指引模式》中的網站端點和將 React 型單一頁面應用程式部署到 Amazon S3 和 CloudFront。

虛擬託管樣式請求

在虛擬託管式的 URL 中，儲存貯體名稱是 URL 中網域名稱的一部分。

Amazon S3 虛擬託管樣式 URL 使用以下格式：

```
https://bucket-name.s3.region-code.amazonaws.com/key-name
```

在此範例中，DOC-EXAMPLE-BUCKET1 是儲存貯體名稱、美國西部 (奧勒岡)是區域，而 puppy.png 是金鑰名稱。

```
https://DOC-EXAMPLE-BUCKET1.s3.us-west-2.amazonaws.com/puppy.png
```

HTTP **Host** 標頭儲存貯體規格

只要 GET 要求未使用 SSL 端點，就可以使用 HTTP Host 標頭來指定要求的儲存貯體。REST 要求中的 Host 標頭解譯如下：

- 如果省略 Host 標頭，或其數值為 s3.*region-code*.amazonaws.com，則請求的儲存貯體會是 Request-URI 的第一個斜線區隔元件，而請求金鑰會是 Request-URI 的其他部分。這是一般方法，如本節的第一個與第二個範例所述。省略 Host 標頭僅對 HTTP 1.0 要求有效。
- 否則，如果 Host 標頭值的結尾為 .s3.*region-code*.amazonaws.com，則儲存貯體名稱是 Host 標頭值到 .s3.*region-code*.amazonaws.com 的前置元件。要求的金鑰是 Request-URI。

這項解譯會將儲存貯體公開為 .s3.*region-code*.amazonaws.com 的子網域，如本節的第三個與第四個範例所述。

- 否則，要求的儲存貯體是 Host 標頭的小寫值，而且要求的金鑰是 Request-URI。如果您已註冊與儲存貯體名稱相同的 DNS 名稱，並且已將該名稱設為 Amazon S3 的正式名稱 (CNAME) 別名，則這項解譯十分有用。註冊網域名稱與設定 CNAME DNS 記錄的程序不是本指南的討論範圍，但本節的最後一個範例會說明其結果。

範例

本節提供 URL 範例與要求。

Example — 路徑樣式 URL 和請求

此範例使用下列各項：

- 儲存貯體名稱 - example.com
- 區域 - 美國東部 (維吉尼亞北部)
- 金鑰名稱- homepage.html

URL 如下：

```
http://s3.us-east-1.amazonaws.com/example.com/homepage.html
```

要求如下：

```
GET /example.com/homepage.html HTTP/1.1
Host: s3.us-east-1.amazonaws.com
```

使用 HTTP 1.0 且省略 Host 標頭的要求如下：

```
GET /example.com/homepage.html HTTP/1.0
```

如需 DNS 相容名稱的資訊，請參閱限制。如需金鑰的詳細資訊，請參閱鍵。

Example — 虛擬託管樣式 URL 和請求

此範例使用下列各項：

- 儲存貯體名稱 - DOC-EXAMPLE-BUCKET1
- 區域- 歐洲 (愛爾蘭)
- 金鑰名稱- homepage.html

URL 如下：

```
http://DOC-EXAMPLE-BUCKET1.s3.eu-west-1.amazonaws.com/homepage.html
```

要求如下：

```
GET /homepage.html HTTP/1.1
Host: DOC-EXAMPLE-BUCKET1.s3.eu-west-1.amazonaws.com
```

Example — CNAME 別名方法

若要使用此方法，您必須將 DNS 名稱設為 *bucket-name*.s3.us-east-1.amazonaws.com 的 CNAME 別名。如需詳細資訊，請參閱「使用 CNAME 記錄自訂 Amazon S3 URL」。

此範例使用下列各項：

- 儲存貯體名稱 - example.com
- 金鑰名稱- homepage.html

URL 如下：

```
http://www.example.com/homepage.html
```

範例如下：

```
GET /homepage.html HTTP/1.1
Host: www.example.com
```

使用 CNAME 記錄自訂 Amazon S3 URL

根據需求，您可能不想要 s3.*region-code*.amazonaws.com 出現在網站或服務上。例如，如果您在 Amazon S3 上託管網站映像，則可能會想要使用 http://images.example.com/，而不是 http://images.example.com.s3.us-east-1.amazonaws.com/。任何具有 DNS 相容名稱的儲存貯體都可以參考如下： http://*BucketName*.s3.*Region*.amazonaws.com/[*Filename*]

(例如，http://images.example.com.s3.us-east-1.amazonaws.com/mydog.jpg)。使用 CNAME，即可將 images.example.com 對應到 Amazon S3 主機名稱，讓前一個 URL 可以成為 http://images.example.com/mydog.jpg。

您的儲存貯體名稱必須與 CNAME 相同。例如，如果您要建立 CNAME，以將 images.example.com 對應至 images.example.com.s3.us-east-1.amazonaws.com，則 http://images.example.com/filename 和 http://images.example.com.s3.us-east-1.amazonaws.com/filename 將是相同的。

CNAME DNS 記錄應該會將網域名稱別名設為適當的虛擬託管式主機名稱。例如，如果您的儲存貯體名稱與網域名稱是 images.example.com，且您的儲存貯體在美國東部 (維吉尼亞北部) 區域中，CNAME 記錄應該會將別名設為 images.example.com.s3.us-east-1.amazonaws.com。

```
images.example.com CNAME    images.example.com.s3.us-east-1.amazonaws.com.
```

Amazon S3 會使用主機名稱來判斷儲存貯體名稱。因此，儲存貯體名稱必須與 CNAME 相同。例如，假設您已將 www.example.com 設定為 www.example.com.s3.us-east-1.amazonaws.com 的 CNAME。當您存取 http://www.example.com 時，Amazon S3 會收到與下列類似的請求：

Example

```
GET / HTTP/1.1
Host: www.example.com
Date: date
Authorization: signatureValue
```

Amazon S3 只會查看原始主機名稱 www.example.com，並不了解用來解析請求的 CNAME 映射。

您可以在 CNAME 別名中使用任何 Amazon S3 端點。例如，s3.ap-southeast-1.amazonaws.com 可以用於 CNAME 別名中。如需端點的詳細資訊，請參閱請求端點。若要使用自訂網域建立靜態網站，請參閱 教學課程：使用向 Route 53 註冊的自訂網域設定靜態網站。

> **Important**
>
> 搭配 CNAME 使用自訂 URL 時，您必須確定您設定的任何 CNAME 或別名記錄都有相符的儲存貯體。例如，如果您建立的 DNS 項目 www.example.com 和 login.example.com 要使用 S3 發布 Web 內容，您需要建立兩個儲存貯體 www.example.com 和 login.example.com。

當設定 CNAME 或別名記錄指向沒有相符儲存貯體的 S3 端點時，任何 AWS 使用者即使擁有權不同，都可以建立該儲存貯體，並在設定的別名下發布內容。
基於相同原因，我們建議您在刪除儲存貯體時變更或移除對應的 CNAME 或別名。

如何建立主機名稱與 Amazon S3 儲存貯體的關聯

使用 CNAME 別名建立主機名稱與 Amazon S3 儲存貯體的關聯

1. 選取屬於所控制網域的主機名稱。

   此範例使用 `images` 網域的 `example.com` 子網域。
2. 建立與主機名稱相符的儲存貯體。

   在此範例中，主機與儲存貯體名稱為 `images.example.com`。儲存貯體名稱必須完全符合主機名稱。
3. 建立 CNAME DNS 記錄，將主機名稱定義為 Amazon S3 儲存貯體的別名。

   例如：

   ```
   images.example.com CNAME images.example.com.s3.us-west-2.amazonaws.com
   ```

   > **Important**
   > 基於要求路由原因，必須如先前範例所示定義完全相同的 CNAME DNS 記錄。否則，它可能看來操作正常，但最後會導致無法預期的行為。

   設定 CNAME DNS 記錄的程序取決於 DNS 伺服器或 DNS 供應商。如需特定資訊，請參閱伺服器文件或聯絡提供者。

限制

HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。建議您不使用 SOAP，而改用 REST API 或 AWS SDK。

回溯相容性

以下各區段涵蓋 Amazon S3 回溯相容性的各個面向，這些面向與路徑樣式和虛擬託管樣式 URL 請求相關。

舊版端點

某些區域支援舊版端點。您可能會在伺服器存取日誌或 AWS CloudTrail 日誌中看到這些端點。如需詳細資訊，請檢閱下列資訊。如需 Amazon S3 區域和端點的完整清單，請參閱《Amazon Web Services 一般參考》中的 Amazon S3 端點和配額。

> **Important**
>
> 雖然您可能會在記錄檔中看到舊版端點，但建議您永遠使用標準端點語法來存取儲存貯體。
>
> Amazon S3 虛擬託管樣式 URL 使用以下格式：
>
> ```
> https://bucket-name.s3.region-code.amazonaws.com/key-name
> ```
>
> 在 Amazon S3 中，路徑樣式 URL 使用以下格式：
>
> ```
> https://s3.region-code.amazonaws.com/bucket-name/key-name
> ```

s3-Region

某些較舊的 Amazon S3 區域支援在 `s3` 和區域代碼 (例如 `s3-us-west-2`) 之間包含破折號 (`-`)，而不是點 (例如 `s3.us-west-2`) 的端點。如果您的儲存貯體位於這些區域之一，您可能會在伺服器存取日誌或 CloudTrail 日誌中看到下列端點格式：

```
https://bucket-name.s3-region-code.amazonaws.com
```

在此範例中，儲存貯體名稱為 DOC-EXAMPLE-BUCKET1，區域為美國西部 (奧勒岡)：

```
https://DOC-EXAMPLE-BUCKET1.s3-us-west-2.amazonaws.com
```

舊版全域端點

對於某些區域，您可以使用舊版全域端點來建構未指定區域特定端點的請求。舊版全域端點如下所示：

```
bucket-name.s3.amazonaws.com
```

在伺服器存取日誌或 CloudTrail 日誌中，您可能會看到使用舊版全域端點的請求。在此範例中，儲存貯體名稱為 `DOC-EXAMPLE-BUCKET1`，且顯示舊版全域端點：

```
https://DOC-EXAMPLE-BUCKET1.s3.amazonaws.com
```

美國東部 (維吉尼亞北部) 虛擬託管樣式請求

依預設，使用舊版全域端點提出的請求會前往美國東部 (維吉尼亞北部) 區域。因此，有時候會使用舊版全域端點來取代美國東部 (維吉尼亞北部) 的區域端點。如果您在美國東部 (維吉尼亞北部) 建立儲存貯體並使用全球端點，則依預設，Amazon S3 會將您的請求路由至此區域。

其他區域的虛擬託管樣式請求

舊版全域端點也可用於其他支援區域中的虛擬託管樣式請求。當您在 2019 年 3 月 20 日之前推出的區域建立儲存貯體並使用舊版全域端點時，Amazon S3 會更新 DNS 記錄，將請求重新路由至正確位置，這可能需要一些時間。在此期間系統會套用預設規則，而您的虛擬託管樣式請求會傳送至美國東部 (維吉尼亞北部) 區域。然後 Amazon S3 會使用 HTTP 307 暫時重新導向到正確的區域以將其重新導向。

若 S3 儲存貯體所在區域是在 2019 年 3 月 20 日之後推出，則 DNS 伺服器不會將您的請求路由至您儲存貯體所在的 AWS 區域。而是會傳回 HTTP 400 錯誤的請求錯誤。如需更多詳細資訊，請參閱 提出要求。

路徑樣式請求

對於美國東部 (維吉尼亞北部) 區域，您可以將舊版全域端點使用於路徑樣式請求。

對於所有其他區域，在嘗試存取儲存貯體時，路徑型語法會需要您使用區域專用端點。如果您嘗試存取的儲存貯體具有舊版全域端點或與儲存貯體所在區域不同的另一個端點，則會收到 HTTP 回應碼 307 暫時重新導向錯誤和指出資源正確 URI 的訊息。例如，如果您對在美國西部 (奧勒岡) 區域中建立的儲存貯體使用 `https://s3.amazonaws.com/bucket-name`，您會收到 HTTP 307 暫時重新導向錯誤。

要求重新導向與 REST API

主題

- 重新導向與 HTTP 使用者代理程式
- 重新導向與 100-Continue
- 重新導向範例

本節描述如何使用 Amazon S3 REST API 處理 HTTP 重新導向。如需 Amazon S3 重新導向的一般資訊，請參閱《Amazon Simple Storage Service API 參考》中的「提出要求」。

重新導向與 HTTP 使用者代理程式

使用 Amazon S3 REST API 的程式應該會在應用程式層級或 HTTP 層級處理重新導向。許多 HTTP 用戶端程式庫與使用者代理程式會自動設定以正確處理重新導向；不過，還有許多其他重新導向實作不正確或不完整。

在您依賴程式庫完成重新導向需求之前，請測試下列案例：

- 確認包含在重新導向要求 (收到重新導向後的第二個要求) 中的所有 HTTP 要求標頭正確，包括 Authorization 與 Date 等 HTTP 標準。
- 確認非 GET 重新導向 (例如 PUT 與 DELETE) 運作正常。
- 確認大型 PUT 要求正確遵循重新導向。
- 如果 100-continue 回應花很長的時間抵達，請確認 PUT 要求正確遵循重新導向。

當 HTTP 要求方法不是 GET 或 HEAD 時，嚴格遵守 RFC 2616 的 HTTP 使用者代理程式可能需要明確確認，才能遵循重新導向。遵循由 Amazon S3 自動產生的重新導向通常很安全，因為系統只會對 amazonaws.com 網域中的主機發出重新導向，因此重新導向請求的效果會與原始請求相同。

重新導向與 100-Continue

為簡化重新導向處理、提高效率，並避免與傳送重新導向要求本文兩次相關聯的成本，請設定您的應用程式，使用 100-continue 進行 PUT 操作。當您的應用程式使用 100-continue 時，只有在收到確認才會傳送要求本文。如果根據標頭拒絕了訊息，則不會傳送訊息本文。如需 100-continue 的詳細資訊，請前往 RFC 2616 第 8.2.3 節

> **Note**
>
> 根據 RFC 2616，對不明 HTTP 伺服器使用 `Expect: Continue` 時，您應該不會無限期地等候傳送要求本文。這是因為雖然某些 HTTP 伺服器無法識別 100-continue，但 Amazon S3 會識別您的請求是否包含 `Expect: Continue`，並回應暫時性 100-continue 狀態或最終狀態碼。此外，收到暫時性 100 continue go-ahead 之後，不會發生重新導向錯誤。這有助於您避免仍在撰寫要求本文時收到重新導向回應。

重新導向範例

本節提供使用 HTTP 重新導向與 100-continue 的主從互動範例。

以下為 `quotes.s3.amazonaws.com` 儲存貯體的 PUT 範例。

```
PUT /nelson.txt HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 傳回下列回應：

```
HTTP/1.1 307 Temporary Redirect
Location: http://quotes.s3-4c25d83b.amazonaws.com/nelson.txt?rk=8d47490b
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Mon, 15 Oct 2007 22:18:46 GMT

Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the
  specified temporary endpoint. Continue to use the
  original request endpoint for future requests.
  </Message>
  <Endpoint>quotes.s3-4c25d83b.amazonaws.com</Endpoint>
  <Bucket>quotes</Bucket>
</Error>
```

用戶端遵循重新導向回應，並對 quotes.s3-4c25d83b.amazonaws.com 暫存端點發出新的要求。

```
PUT /nelson.txt?rk=8d47490b HTTP/1.1
Host: quotes.s3-4c25d83b.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 傳回 100-continue，指出用戶端應該繼續傳送請求本文。

```
HTTP/1.1 100 Continue
```

用戶端傳送要求本文。

```
ha ha\n
```

Amazon S3 傳回最終回應。

```
HTTP/1.1 200 OK
Date: Mon, 15 Oct 2007 22:18:48 GMT

ETag: "a2c8d6b872054293afd41061e93bc289"
Content-Length: 0
Server: AmazonS3
```

使用 AWS CLI 來透過 Amazon S3 進行開發

請遵循這些步驟來下載及設定 AWS Command Line Interface (AWS CLI)。

如需 Amazon S3 AWS CLI 命令清單，請參閱《AWS CLI 命令參考》中的以下頁面：

- s3
- s3api
- s3control

> **Note**
>
> AWS 服務，例如 Amazon S3，會在您進行存取時，要求您提供登入資料。接著服務可以判斷您是否擁有存取其資源的許可。主控台需要您的密碼。您可以建立 AWS 帳戶 的存取金鑰，用以存取 AWS CLI 或 API。不過，不建議您使用 AWS 帳戶 的登入資料來存取 AWS。建議您改用 AWS Identity and Access Management (IAM)。建立 IAM 使用者，並將使用者新增至擁有管理許可的 IAM 群組，然後將管理許可授予您建立的 IAM 使用者。之後，您可以使用特殊的 URL 和 IAM 使用者的憑證來存取 AWS。如需說明，請前往 IAM 使用者指南中的建立第一個 IAM 使用者與管理員群組。

設定 AWS CLI

1. 下載和設定 AWS CLI。如需說明，請參閱《AWS Command Line Interface 使用者指南》中的下列主題：

- 設定 AWS Command Line Interface
- 設定 AWS Command Line Interface

2. 在 AWS CLI 組態檔中，為管理員使用者新增命名描述檔。當您執行 AWS CLI 命令時，使用此設定檔。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的 AWS CLI 的具名設定檔。

```
[adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

如需可用 AWS 區域 的清單，請參閱《AWS 一般參考》中的區域與端點

3. 在命令提示中輸入下列命令，以驗證設定。

- 嘗試 `help` 命令，驗證您的電腦上已安裝 AWS CLI：

```
aws help
```

- 執行使用您剛建立的 `adminuser` 憑證的 S3 命令。若要執行這項操作，請新增 `--profile` 參數至您的命令來指定描述檔名稱。此範例中，`ls` 命令會列出您帳戶中的儲存貯體。AWS CLI 使用 `adminuser` 登入資料對請求進行身分驗證。

```
aws s3 ls --profile adminuser
```

使用開發 AWS 套件使用 Amazon S3 進行開發

AWS 軟件開發套件（SDK）可用於許多流行的編程語言。每個 SDK 都提供 API、程式碼範例和說明文件，讓開發人員能夠更輕鬆地以偏好的語言建置應用程式。

> **Note**
>
> 您可以用 AWS Amplify 於 Web 和移動應用程序的 end-to-end 全棧開發。Amplify Storage 將檔案儲存和管理功能無縫整合到建置在 Amazon S3 之上的前端 Web 和行動應用程式中。如需詳細資訊，請參閱 Amplify 使用指南中的儲存空間。

搭配 AWS SDK 使用此服務

AWS 軟件開發套件（SDK）可用於許多流行的編程語言。每個 SDK 都提供 API、程式碼範例和說明文件，讓開發人員能夠更輕鬆地以偏好的語言建置應用程式。

SDK 文件	代碼範例
AWS SDK for C++	AWS SDK for C++ 程式碼範例
AWS CLI	AWS CLI 程式碼範例
AWS SDK for Go	AWS SDK for Go 程式碼範例
AWS SDK for Java	AWS SDK for Java 程式碼範例
AWS SDK for JavaScript	AWS SDK for JavaScript 程式碼範例
適用於 Kotlin 的 AWS SDK	適用於 Kotlin 的 AWS SDK 程式碼範例
AWS SDK for .NET	AWS SDK for .NET 程式碼範例
AWS SDK for PHP	AWS SDK for PHP 程式碼範例
AWS Tools for PowerShell	PowerShell 程式碼範例的工具
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 程式碼範例
AWS SDK for Ruby	AWS SDK for Ruby 程式碼範例
適用於 Rust 的 AWS SDK	適用於 Rust 的 AWS SDK 程式碼範例
適用於 SAP ABAP 的 AWS SDK	適用於 SAP ABAP 的 AWS SDK 程式碼範例
適用於 Swift 的 AWS SDK	適用於 Swift 的 AWS SDK 程式碼範例

如需此服務的特定範例，請參閱 使用 AWS 開發套件的 Amazon S3 程式碼範例。

> **可用性範例**
>
> 找不到所需的內容嗎？ 請使用本頁面底部的提供意見回饋連結申請程式碼範例。

SDK 程式設計介面

每個 AWS 開發套件都提供一或多個程式設計界面，以便與 Amazon S3 搭配使用。每個開發套件都為 Amazon S3 提供低階介面，其方法與 API 操作非常類似。有些開發套件為 Amazon S3 提供高階介面，這些介面是旨在簡化常見使用案例的抽象化。

例如，當您使用低階 API 作業執行多部分上傳時，您需要使用作業來啟動上載，另一項上載零件的作業，以及完成上載作業的最終作業。高階分段上傳 API 作業可讓您在單一 API 呼叫中執行上傳所需的所有作業。如需範例，請參閱 使用分段上傳來上傳物件。

低級 API 操作允許更好地控制上傳。如果您需要暫停和繼續上傳、在上傳期間變更零件大小，或者在事先不知道資料大小時開始上傳，建議您使用低階 API 作業。

在請求身分驗證中指定 Signature 版本

Amazon S3 在大多數情況下僅支持 AWS 簽名版本 4 AWS 區域。在某些較舊版本中 AWS 區域，Amazon S3 同時支援簽名版本 4 和簽名版本 2。然而，Signature 第 2 版已結束 (已廢除)。如需 Signature 第 2 版終止支援的詳細資訊，請參閱AWS Amazon S3 的簽名版本 2 已關閉 (已淘汰)。

如需所有 Amazon S3 區域及其支援之簽章版本的清單，請參閱《AWS 一般參考》中的區域與端點。

對於所有 AWS SDK AWS 區域，默認情況下使用簽名版本 4 來驗證請求。使用 2016 年 5 月之前發行的 AWS SDK 時，您可能需要請求簽名版本 4，如下表所示。

SDK	要求 Signature 第 4 版進行要求身分驗證
AWS CLI	針對預設描述檔，執行下列命令： `$ aws configure set default.s3.signature_version s3v4` 針對自訂描述檔，執行下列命令： `$ aws configure set profile.your_profile_name.s3.signature_version s3v4`
Java 開發套件	在程式碼中新增下列程式碼：

SDK	要求 Signature 第 4 版進行要求身分驗證
	`System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");` 或者，在命令列指定下列項目： `-Dcom.amazonaws.services.s3.enableV4`
JavaScript SDK	建構用戶端時，請將 `signatureVersion` 參數設定為 v4： `var s3 = new AWS.S3({signatureVersion: 'v4'});`
PHP SDK	針對 PHP SDK v2 建構 Amazon S3 服務用戶端時，請將 `signature` 參數設為 v4： `<?php` `$client = S3Client::factory([` `'region' => 'YOUR-REGION',` `'version' => 'latest',` `'signature' => 'v4'` `]);` 使用 PHP SDK 第 3 版時，請在建構 Amazon S3 服務用戶端時，將 `signature_version` 參數設為 v4： `<?php` `$s3 = new Aws\S3\S3Client([` `'version' => '2006-03-01',` `'region' => 'YOUR-REGION',` `'signature_version' => 'v4'` `]);`
Python-Boto SDK	在 boto 預設組態檔中，指定以下內容： `[s3] use-sigv4 = True`

SDK	要求 Signature 第 4 版進行要求身分驗證
Ruby 開發套件	Ruby 開發套件 - 第 1 版：建構用戶端時，請將 `:s3_signature_version` 參數設定為 `:v4`： `s3 = AWS::S3::Client.new(:s3_signature_version => :v4)` Ruby 開發套件 - 第 3 版：建構用戶端時，請將 `signature_version` 參數設定為 `v4`： `s3 = Aws::S3::Client.new(signature_version: 'v4')`
.NET 開發套件	在建立 Amazon S3 用戶端之前，將以下內容新增至程式碼： `AWSConfigsS3.UseSignatureVersion4 = true;` 或者，將以下內容新增至組態檔： `<appSettings>` `  <add key="AWS.S3.UseSignatureVersion4" value="true" />` `</appSettings>`

AWS Amazon S3 的簽名版本 2 已關閉 (已淘汰)

Amazon S3 中的 Signature 第 2 版已結束 (已廢除)。Amazon S3 將僅接受使用 Signature 第 4 版簽署的 API 請求。

本節提供有關 Signature 第 2 版終止支援的常見問答。

什麼是 Signature 第 2/4 版，而簽署請求又表示什麼意思？

Signature 第 2 版或 Signature 第 4 版簽署程序用於驗證您的 Amazon S3 API 請求。簽署請求可讓 Amazon S3 識別傳送請求的人員，並保護請求免於受到不良執行者的破壞。

如需簽署 AWS 要求的詳細資訊，請參閱 AWS 一般參考. AWS

更新包含什麼內容？

我們目前支援使用 Signature 第 2 版和 Signature 第 4 版程序簽署的 Amazon S3 API 請求。在此之後，Amazon S3 將僅接受使用 Signature 第 4 版簽署的請求。

如需有關簽署 AWS 要求的詳細資訊，請參閱 AWS 一般參考.

更新的原因？

Signature 第 4 版使用簽署金鑰 (而不使用私密存取金鑰) 來提高安全性。目前所有簽名版本 4 都支援 AWS 區域，而簽名版本 2 僅在 2014 年 1 月之前推出的地區支援。這項更新可讓我們在所有區域中提供更一致的體驗。

我要如何確定正在使用的是 Signature 第 4 版，以及我需要進行哪些更新？

用於簽署請求的 Signature 版本，其常見的設定方法為用戶端上的工具或 SDK。根據預設，我們 AWS SDK 的最新版本使用簽名版本 4。對於第三方軟體，請聯絡適當的軟體支援團隊以確認您需要的版本。如果要向 Amazon S3 傳送直接 REST 呼叫，必須修改應用程式以使用 Signature 第 4 版簽署程序。

如需移至簽名版本 4 時要使用的 AWS SDK 版本的相關資訊，請參閱從 Signature 第 2 版遷移至 Signature 第 4 版。

如需有關在 Amazon S3 REST API 中使用 Signature 第 4 版的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的身分驗證請求 (AWS Signature 第 4 版)。

如果不更新會發生什麼情況？

在此之後使用 Signature 第 2 版簽署的請求，將無法向 Amazon S3 進行身分驗證。請求者將看到系統顯示錯誤，表示必須使用 Signature 第 4 版簽署請求。

即使我使用的預先簽章 URL 要求大於 7 天的簽署時間，是否也需要進行更新？

如果您使用的預先簽章 URL 要求大於 7 天的簽署時間，則目前無需採取任何動作。您可以繼續使用「 AWS 簽名版本 2」來簽署和驗證預先簽署的 URL。我們將持續追蹤，並提供如何針對預先簽章 URL 的情形遷移到 Signature 第 4 版的詳細資訊。

詳細資訊

- 如需使用簽名版本 4 的詳細資訊，請參閱簽署 AWS API 要求。
- 在 Signature 第 4 版的變更中檢視 Signature 第 2 版和 Signature 第 4 版之間的變更清單。

- 在 AWS 論壇中檢視AWS 簽章版本 4 以取代 AWS 簽名版本 2 以簽署 Amazon S3 API 請求的貼文。
- 如果您有其他問題或疑慮，請聯絡 AWS Support。

從 Signature 第 2 版遷移至 Signature 第 4 版

如果您目前使用 Signature 第 2 版進行 Amazon S3 API 請求身分驗證，應改為使用 Signature 第 4 版。系統即將終止對 Signature 第 2 版的支援，如 AWS Amazon S3 的簽名版本 2 已關閉 (已淘汰) 中所述。

如需有關在 Amazon S3 REST API 中使用 Signature 第 4 版的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的身分驗證請求 (AWS Signature 第 4 版)。

下表列出使用 Signature 第 4 版 (SigV4) 所需的最低軟體開發套件版本。如果您將預先簽署的 URL 與 AWS Java、 JavaScript (Node.js) 或 Python (BOTO/ CLI) 開發套件搭配使用，您必須在用戶端設定中設定正確 AWS 區域 的簽章版本 4。如需在用戶端組態中設定 `SigV4` 的資訊，請參閱 在請求身分驗證中指定 Signature 版本。

如果您使用此軟體開發套件/產品	升級至此軟體開發套件版本	用戶端使用 Sigv4 時需要變更程式碼嗎？	軟體開發套件文件的連結
AWS SDK for Java V1	升級至 Java 1.11.201+ 或 v2。	是	在請求身分驗證中指定 Signature 版本
AWS SDK for Java V2	無需升級軟體開發套件。	否	AWS SDK for Java
AWS SDK for .NET V1	升級至 3.1.10 或更新版本。	是	AWS SDK for .NET
AWS SDK for .NET V2	升級至 3.1.10 或更新版本。	否	AWS SDK for .NET v2

如果您使用此軟體開發套件/產品	升級至此軟體開發套件版本	用戶端使用 Sigv4 時需要變更程式碼嗎？	軟體開發套件文件的連結
AWS SDK for .NET V3	升級至 3.3.0.0 或更新版本。	是	AWS SDK for .NET V3
AWS SDK for JavaScript V1	升級至 2.68.0 或更新版本。	是	AWS SDK for JavaScript
AWS SDK for JavaScript V2	升級至 2.68.0 或更新版本。	是	AWS SDK for JavaScript
AWS SDK for JavaScript V3	目前無需採取動作。在 2019 年第 3 季升級至主要版本 V3。	否	AWS SDK for JavaScript
AWS SDK for PHP V1	建議升級至 PHP 的最新版本，或至少升級至 v2.7.4，其中 S3 用戶端組態中的簽署參數應設為 v4。	是	AWS SDK for PHP

如果您使用此軟體開發套件/產品	升級至此軟體開發套件版本	用戶端使用 Sigv4 時需要變更程式碼嗎？	軟體開發套件文件的連結
AWS SDK for PHP V2	建議升級至 PHP 的最新版本，或至少升級至 v2.7.4，其中 S3 用戶端組態中的簽署參數應設為 v4。	否	AWS SDK for PHP
AWS SDK for PHP V3	無需升級軟體開發套件。	否	AWS SDK for PHP
Boto2	升級至 Boto2 v2.49.0。	是	Boto 2 升級
Boto3	升級至 1.5.71 (Botocore)、1.4.6 (Boto3)。	是	博托 3-適用於 AWS Python 的 SDK
AWS CLI	升級至 1.11.108。	是	AWS Command Line Interface
AWS CLI 第 2 版 (預覽版)	無需升級軟體開發套件。	否	AWS Command Line Interface 第二版
AWS SDK for Ruby V1	升級至 Ruby V3。	是	適用於 AWS的 Ruby V3
AWS SDK for Ruby V2	升級至 Ruby V3。	是	適用於 AWS的 Ruby V3

如果您使用此軟體開發套件/產品	升級至此軟體開發套件版本	用戶端使用 Sigv4 時需要變更程式碼嗎？	軟體開發套件文件的連結
AWS SDK for Ruby V3	無需升級軟體開發套件。	否	適用於 AWS的 Ruby V3
Go	無需升級軟體開發套件。	否	AWS SDK for Go
C++	無需升級軟體開發套件。	否	AWS SDK for C++

AWS Tools for Windows PowerShell 或 AWS Tools for PowerShell Core

如果您使用的模組版本早於 3.3.0.0，則必須升級到 3.3.0.0。

如需取得版本資訊，請使用 `Get-Module` cmdlet：

```
Get-Module -Name AWSPowershell
Get-Module -Name AWSPowershell.NetCore
```

若要更新 3.3.0.0 版，請使用 `Update-Module` cmdlet：

```
Update-Module -Name AWSPowershell
Update-Module -Name AWSPowershell.NetCore
```

您可以使用有效期超過 7 天的預先簽署 URL，您將使用該 URL 傳送 Signature 第 2 版流量。

使用 REST API 與 Amazon S3 進行開發

Amazon S3 架構設計成非程式設計語言相關，並使用支援的介面來存放與擷取物件。

Amazon S3 目前提供 REST 界面。使用 REST 時，中繼資料會在 HTTP 標頭中傳回。由於我們最多只支援 4 KB (不包括本文) 的 HTTP 要求，因此您可以提供的中繼資料量有限。REST API 是 Amazon S3 的 HTTP 界面。透過 REST，您可以使用標準 HTTP 要求來建立、擷取與刪除儲存貯體與物件。

您可以使用支援 HTTP 的任何工具組來使用 REST API。您甚至可以使用瀏覽器來擷取物件，只要物件是可匿名讀取即可。

REST API 使用標準 HTTP 標頭與狀態碼，因此標準瀏覽器與工具組會如預期般運作。在某些區域中，我們已新增功能至 HTTP (例如，我們已新增標頭來支援存取控制)。在此情況下，我們會盡力以符合標準 HTTP 使用風格的方式來新增功能。

如需使用 REST API 傳送請求的詳細資訊，請參閱 使用 REST API 提出要求。使用 REST API 時需要注意一些注意事項，請參閱下列主題。

如需 Amazon S3 REST API 的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》。

主題

要求路由

對使用 CreateBucket API (包含 CreateBucketConfiguration) 建立的儲存貯體提出請求的程式必須支援重新導向。此外，未採用 DNS TTL 的某些用戶端可能會遇到問題。

本節說明設計服務或應用程式以搭配 Amazon S3 使用時須考量的路由與 DNS 問題。

要求重新導向與 REST API

Amazon S3 會使用網域名稱系統 (DNS) 將請求路由到能夠處理請求的設備。此系統有效率執行作業，但是也可能會發生暫時性的路由錯誤。若請求抵達錯誤的 Amazon S3 位置，則 Amazon S3 會以暫時重新導向回應，指示申請者將請求重新傳送到新端點。若請求的格式錯誤，Amazon S3 會使用永久重新導向來指示如何正確執行請求。

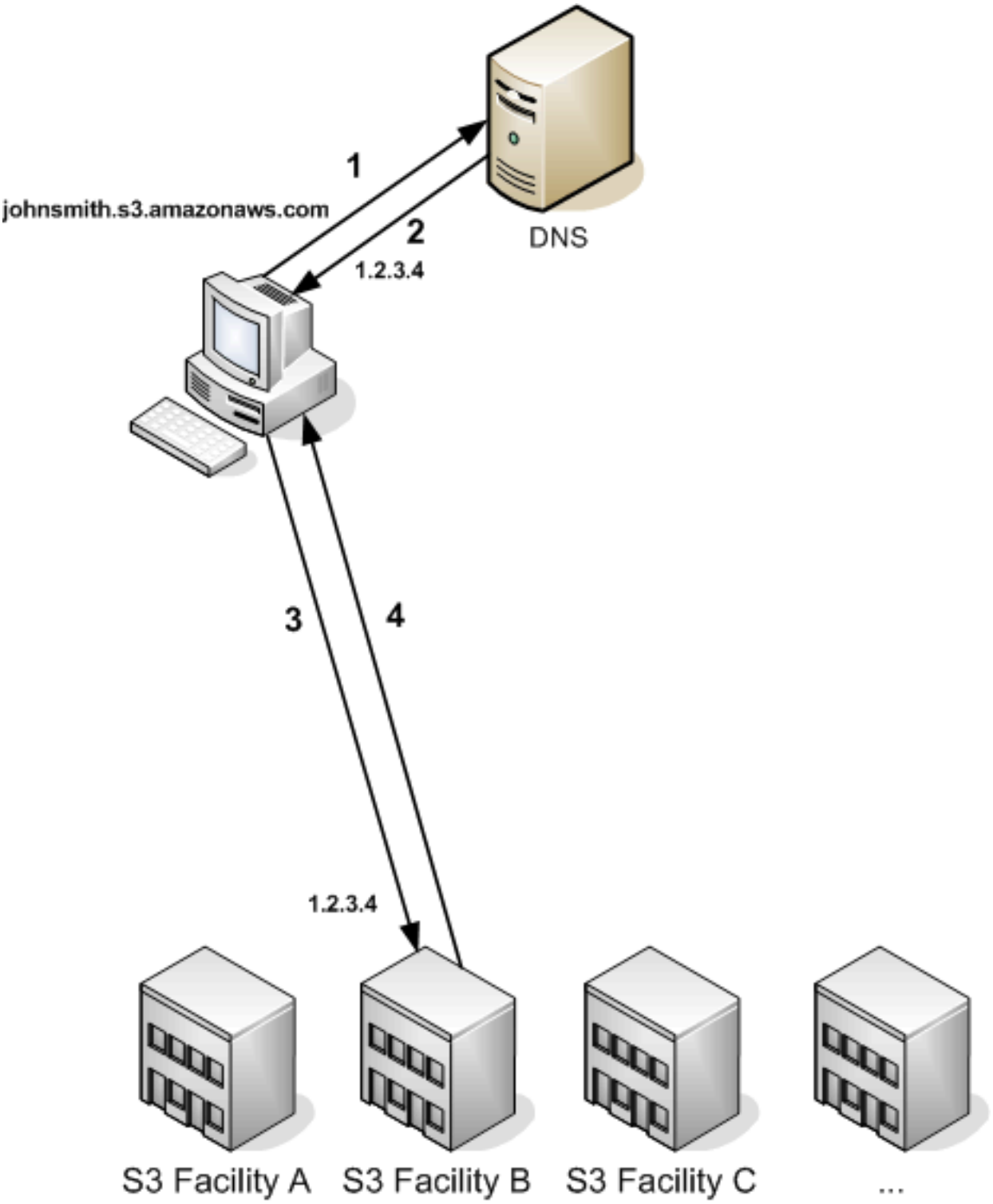
> **Important**
>
> 您必須擁有可以處理 Amazon S3 重新導向回應的應用程式，才可以使用此功能。此應用程式唯一的例外是專為不使用 `<CreateBucketConfiguration>` 建立儲存貯體。如需位置限制條件的詳細資訊，請參閱 存取及列出 Amazon S3 儲存貯體。

如果區域是在 2019 年 3 月 20 日後才推出，則當請求抵達錯誤的 Amazon S3 位置時，Amazon S3 會傳回「HTTP 400 錯誤的請求」錯誤。
如需啟用或停用 AWS 區域的詳細資訊，請參閱《AWS 一般參考》中的 AWS 區域 與端點。

主題

DNS 路由

DNS 路由會將請求路由到合適的 Amazon S3 設備。下圖範例顯示 DNS 路由過程。



DNS 路由要求步驟

1. 用戶端提出 DNS 請求，以取得存放在 Amazon S3 上的物件。
2. 用戶端收到一或多個可處理要求的設備 IP 位址。在此範例中，IP 地址為 B 設施。

3. 用戶端向 Amazon S3 設備 B 提出請求。
4. 設備 B 傳回物件的複本至客戶端。

暫時要求重新導向

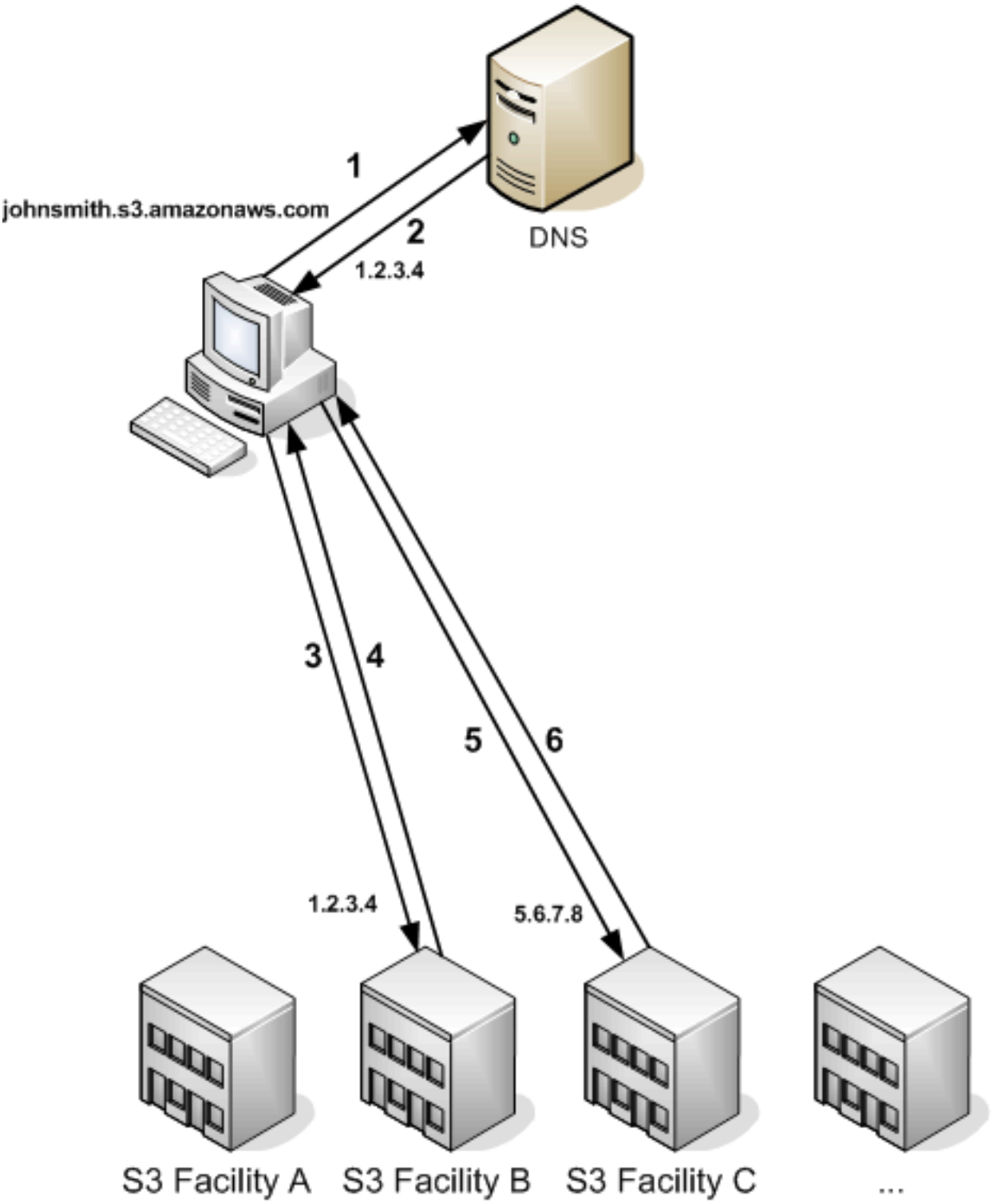
暫時性重新導向是一種錯誤回應，會向申請者發出訊息，要求其必須將要求重新傳送至其他端點。Amazon S3 的分散特性可能會使請求暫時路由至錯誤的設備。這是相當少見的情況，最可能在建立儲存貯體之後立即發生。

例如，若您在建立新儲存貯體後立即向儲存貯體提出要求，就可能會收到暫時性重新導向，這會視儲存貯體的位置限制條件而定。如果您是在美國東部 (維吉尼亞北部) AWS 區域 建立儲存貯體，就不會看到重新引導，因為其本身也屬於預設的 Amazon S3 端點。

但是，若在任何其他區域建立儲存貯體，則任何對儲存貯體提出的要求都會前往預設端點，而儲存貯體的 DNS 項目會傳播。預設端點會要求重新導向到正確的端點，並傳回 HTTP 302 回應。暫時性重新導向會包含正確設備的 URI，您可使用來立即重新傳送要求。

> **Important**
>
> 請勿重複使用前述重新導向回應提供的端點。該端點可能看似運作正常 (即使是長時間運作)，但其可能會提供無法預測的結果，最終導致失敗且不會產生任何通知。

下圖過程顯示暫時性重新導向路由範例。



暫時要求重新導向步驟

1. 用戶端提出 DNS 請求，以取得存放在 Amazon S3 上的物件。
2. 用戶端收到一或多個可處理要求的設備 IP 位址。

3. 用戶端向 Amazon S3 設備 B 提出請求。
4. 設備 B 傳回重新導向，表示可從位置 C 使用該物件。
5. 用戶端將要求重新傳送至設備 C。
6. 設備 C 傳回物件的複本。

永久性要求重新導向

永久性重新導向表示要求設定的資源位置錯誤。例如，若您使用路徑型要求來存取使用 `<CreateBucketConfiguration>` 建立的儲存貯體，就會發生永久性重新導向。如需詳細資訊，請參閱「存取及列出 Amazon S3 儲存貯體」。

為協助您在部署期間找出這些錯誤，這類重新導向都不會包含可讓您自動跟隨要求前往正確位置的位置 HTTP 標頭。如需使用正確 Amazon S3 端點方面的說明，請參閱產生的 XML 錯誤文件。

要求重新導向範例

下圖為暫時重新導回應範例。

REST API 暫時要求重新導向回應

```
HTTP/1.1 307 Temporary Redirect
Location: http://awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com/photos/puppy.jpg?
rk=e2c69a31
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Fri, 12 Oct 2007 01:12:56 GMT
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the specified temporary endpoint.
  Continue to use the original request endpoint for future requests.</Message>
  <Endpoint>awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com</Endpoint>
</Error>
```

SOAP API 暫時要求重新導向回應

> **Note**
>
> HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。建議您不使用 SOAP，而改用 REST API 或 AWS SDK。

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.TemporaryRedirect</Faultcode>
    <Faultstring>Please re-send this request to the specified temporary endpoint.
    Continue to use the original request endpoint for future requests.</Faultstring>
    <Detail>
      <Bucket>images</Bucket>
      <Endpoint>s3-gztb4pa9sq.amazonaws.com</Endpoint>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

DNS 考量

Amazon S3 的設計要求之一是極高的可用性。符合此要求的其中一種方式是視需要更新與 DNS 中 Amazon S3 端點相關聯的 IP 地址。這些變更會自動反映至短期用戶端，而不是一些長期用戶端。長期用戶端需要定期採取特殊動作來重新解析 Amazon S3 端點，以從這些變更受益。如需虛擬機器 (VM) 的詳細資訊，請參閱下列項目：

- 針對 Java，根據預設，Sun 的 JVM 會永久快取 DNS 查閱；如需如何變更此行為的資訊，請前往 InetAddress 文件的「InetAddress 快取」一節。
- 針對 PHP，除非重新啟動 VM，否則最常見部署組態中執行的持久性 PHP VM 會快取 DNS 查閱。前往 getHostByName PHP 文件。

處理 REST 與 SOAP 錯誤

主題

本節說明 REST 與 SOAP 錯誤及其處理方法。

> **Note**
>
> HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。建議您不使用 SOAP，而改用 REST API 或 AWS SDK。

REST 錯誤回應

當 REST 要求造成錯誤時，HTTP 回覆會包含：

- XML 錯誤文件作為回應內文
- Content-Type: application/xml
- 適當的 3xx、4xx 或 5xx HTTP 狀態碼

以下是 REST 錯誤回應範例。

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The resource you requested does not exist</Message>
  <Resource>/mybucket/myfoto.jpg</Resource>
  <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

如需 Amazon S3 錯誤的詳細資訊，請前往 ErrorCodeList。

回應標頭

以下是所有操作傳回的回應標頭：

- `x-amz-request-id`：指派給每個要求的專用 ID。當 Amazon S3 發生問題時，Amazon 可能會使用此 ID 協助疑難排解問題，但這個可能性極低。
- `x-amz-id-2`：協助我們疑難排解問題的特殊字符。

錯誤回應

當 Amazon S3 請求發生錯誤時，用戶端會收到錯誤回應。錯誤回應的確切格式隨 API 而異。例如，REST 錯誤回應與 SOAP 錯誤回應就不相同。但所有錯誤回應都有共同的元素。

> **Note**
>
> HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。建議您不使用 SOAP，而改用 REST API 或 AWS SDK。

錯誤代碼

錯誤代碼是一個字串，專門用於識別錯誤狀況。偵測及依類型處理錯誤的程式必須讀取及了解此代碼。SOAP 與 REST API 有許多共通的錯誤代碼，但有些會隨 API 而異。例如，NoSuchKey 兩者都有，但 UnexpectedContent 只會出現在無效 REST 要求的回應中。在所有案例中，SOAP 錯誤碼都包含錯誤代碼表中指出的字首，因此，SOAP 中傳回的 NoSuchKey 錯誤，實際會顯示為 Client.NoSuchKey。

> **Note**
>
> HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。建議您不使用 SOAP，而改用 REST API 或 AWS SDK。

錯誤訊息

錯誤訊息包含錯誤狀況的一般英文描述，目標對象是人。一般簡單的程式在發生其無法處理的錯誤狀況時，會直接對最終使用者顯示訊息。複雜一些的程式因為錯誤處理更加周密，而且經過適當的國際化，所以很可能會忽略錯誤訊息。

更多詳細資訊

許多錯誤回應包含額外的結構化資料，負責診斷程式設計錯誤的開發人員應詳加閱讀及了解。例如，若您傳送之 REST PUT 要求中所包含的 Content-MD5 標頭，不符合伺服器計算所得的摘要，您就會收到 BadDigest 錯誤。錯誤回應也會包含在詳細資訊元素中計算所得的摘要，以及您告訴我們應有的摘要。您可以在開發時使用此資訊診斷錯誤。在生產環境中，一支運作良好的程式可能將此資訊包含在其錯誤日誌中。

SOAP 錯誤回應

> **Note**
>
> HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。建議您不使用 SOAP，而改用 REST API 或 AWS SDK。

在 SOAP 中，傳回給用戶端的錯誤結果是 SOAP 錯誤，其 HTTP 回應碼為 500。若未收到 SOAP 錯誤，表示要求成功。Amazon S3 的 SOAP 錯誤碼是由標準的 SOAP 1.1 錯誤碼加上 Amazon S3 專用錯誤代碼 (「Server」或「Client」) 組成。例如："Server.InternalError" 或 "Client.NoSuchBucket"。SOAP 錯誤字串元素包含可供人閱讀的一般英文錯誤訊息。最後，SOAP 錯誤詳細資訊元素也包含錯誤相關的其他資訊。

例如，當您嘗試刪除不存在的物件 "Fred" 時，SOAP 回應的內文中將會出現 "NoSuchKey" SOAP 錯誤。

Example

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.NoSuchKey</Faultcode>
    <Faultstring>The specified key does not exist.</Faultstring>
    <Detail>
      <Key>Fred</Key>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

如需 Amazon S3 錯誤的詳細資訊，請前往 ErrorCodeList。

Amazon S3 錯誤的最佳實務

當您設計可與 Amazon S3 搭配使用的應用程式時，正確處理 Amazon S3 錯誤十分重要。本節說明設計應用程式時須考量的問題。

重試 InternalError

內部錯誤是指 Amazon S3 環境內發生的錯誤。

收到 InternalError 回應的要求可能尚未處理。例如，若 PUT 要求傳回 InternalError，後續的 GET 可能會擷取舊值，也可能會擷取更新後的值。

若 Amazon S3 傳回 InternalError 回應，請重試該請求。

針對重複出現的 SlowDown 錯誤微調應用程式

一如所有的分散式系統，S3 也設有保護機制，可以偵測有意或無意的資源過度使用，同時採取相應的回應。當要求率觸發其中一項機制時，即會產生 SlowDown 錯誤。降低要求率即可減少或消除此類型的錯誤。一般來說，大多數使用者不會定期遇到這些錯誤；不過，如果您想要更多資訊，或是遇到高度或意外的 SlowDown 錯誤，請將問題張貼到我們的 Amazon S3 開發人員論壇或註冊 AWS Support https://aws.amazon.com/premiumsupport/。

隔離錯誤

> **Note**
>
> HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。建議您不使用 SOAP，而改用 REST API 或 AWS SDK。

Amazon S3 提供一組錯誤代碼供 SOAP 與 REST API 使用。SOAP API 會傳回標準的 Amazon S3 錯誤代碼。REST API 的設計類似標準的 HTTP 伺服器，會與現有的 HTTP 用戶端 (例如瀏覽器、HTTP 用戶端程式庫、代理伺服器、快取等等) 互動。為確保 HTTP 用戶端正確處理錯誤，每個 Amazon S3 錯誤都會對應到一個 HTTP 狀態碼。

HTTP 狀態碼不如 Amazon S3 錯誤碼易懂，而且包含的錯誤資訊也較少。例如 `NoSuchKey` 及 `NoSuchBucket` 兩個 Amazon S3 錯誤都可對應到 `HTTP 404 Not Found` 狀態碼。

雖然 HTTP 狀態碼包含的錯誤資訊較少，但了解 HTTP 卻不了解 Amazon S3 API 的用戶端，通常都能正確處理錯誤。

因此在處理錯誤或將 Amazon S3 錯誤回報給使用者時，會使用 Amazon S3 錯誤代碼，而不會使用 HTTP 狀態碼，因為前者包含了大部分的錯誤資訊。此外，在對應用程式執行除錯時，您也應該參考 XML 錯誤回應之 <Details> 元素中可供人閱讀的內容。

開發人員參考

此附錄包含下列章節。

主題

附錄 A：使用 SOAP API

> **Note**
>
> HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。我們建議您使用 REST API 或 AWS 開發套件，而不是使用 SOAP。

本節包含 Amazon S3 SOAP API 專屬的資訊。

> **Note**
>
> SOAP 請求 (已驗證與匿名) 必須透過 SSL 傳送至 Amazon S3。當您透過 HTTP 傳送 SOAP 請求時，Amazon S3 會傳回錯誤。

主題

常見的 SOAP API 元素

> **Note**
>
> HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。建議您不使用 SOAP，而改用 REST API 或 AWS SDK。

您可以使用 SOAP 1.1 over HTTP 來與 Amazon S3 互動。Amazon S3 WSDL 以機器可讀取方式描述 Amazon S3 API，可在下列網址取得：https://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl。在 https://doc.s3.amazonaws.com/2006-03-01/AmazonS3.xsd 上可取得 Amazon S3 結構描述。

大多數使用者會使用針對其語言和開發環境而量身打造的 SOAP 工具組，與 Amazon S3 進行互動。不同的工具組會以不同的方式公開 Amazon S3 API。請參閱適用於您的工具組文件來了解其使用方式。本節展示線上呈現的 XML 請求和回應，以無關於工具組的方式闡明 Amazon S3 SOAP 作業。

常見元素

您可以在任何 SOAP 要求中納入下列與授權相關的元素：

- `AWSAccessKeyId`: 申請者的 AWS 存取金鑰 ID
- `Timestamp`: 系統上目前的時間
- `Signature`: 要求的簽章

對 SOAP 要求進行身分驗證

> **Note**
>
> HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。建議您不使用 SOAP，而改用 REST API 或 AWS SDK。

每個非匿名要求都必須包含身分驗證資訊，才能建立提出要求之委託人的身分。在 SOAP 中，身分驗證資訊會放入 SOAP 要求的下列元素中：

- 您的 AWS 存取金鑰 ID

  > **Note**
  >
  > 提出經過身分驗證的 SOAP 要求時，不支援暫時性安全登入資料。如需登入資料類型的詳細資訊，請參閱「提出要求」。

- `Timestamp`: 這必須是國際標準時間 (格林威治標準時間) 時區 (例如 `2009-01-01T12:00:00.000Z`) 的 dateTime (請前往 http://www.w3.org/TR/xmlschema-2/#dateTime)。如果此時間戳記與 Amazon S3 伺服器上的時鐘相差超過 15 分鐘，則授權會失敗。
- `Signature`: 使用 AWS 私密存取金鑰作為金鑰，連接「AmazonS3」+ OPERATION + Timestamp 的 RFC 2104 HMAC-SHA1 摘要 (請前往 http://www.ietf.org/rfc/rfc2104.txt)。例如，在下列 CreateBucket 要求範例中，簽章元素將會包含值為 "AmazonS3CreateBucket2009-01-01T12:00:00.000Z" 的 HMAC-SHA1 摘要：

例如，在下列 CreateBucket 要求範例中，簽章元素將會包含值為 "AmazonS3CreateBucket2009-01-01T12:00:00.000Z" 的 HMAC-SHA1 摘要：

Example

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Acl>private</Acl>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2009-01-01T12:00:00.000Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

Note

SOAP 請求 (已驗證與匿名) 必須透過 SSL 傳送至 Amazon S3。當您透過 HTTP 傳送 SOAP 請求時，Amazon S3 會傳回錯誤。

Important

因為對於如何去除多餘的時間精確度有不同的解釋，.NET 使用者應該注意不要將過於精確的時間戳記傳送給 Amazon S3。這可以透過手動建構只有毫秒精準度的 `DateTime` 物件來完成。

使用 SOAP 設定存取原則

Note

HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。我們建議您使用 REST API 或 AWS 開發套件，而不是使用 SOAP。

您可以在寫入值區或物件時設定存取控制，方法是將 "AccessControlList" 元素與請求一起加入`CreateBucket`、`PutObjectInline`、或`PutObject`。 AccessControlList 元素會在中進行描述適用於 Amazon S3 的 Identity and Access Management。如果這些作業未指定存取控制清單，則會使用預設存取原則來建立資源，該原則會提供請求者 FULL_CONTROL 存取權 (即使請求是對已存在物件的 PutObjectInline 或要 PutObject 求，也會發生這種情況)。

下列要求會將資料寫入物件、允許匿名委託人讀取物件，以及將儲存貯體的 FULL_CONTROL 權限授予指定的使用者 (大部分的開發人員會授予自己其擁有之儲存貯體的 FULL_CONTROL 存取權)。

Example

下列要求會將資料寫入物件，並允許匿名委託人讀取物件。

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2009-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>&quot828ef3fdfa96f00ad9f27c383fc9ac7f&quot</ETag>
```

```
    <LastModified>2009-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

使用 `GetBucketAccessControlPolicy`、`GetObjectAccessControlPolicy`、`SetBucketAccessContro` 及 `SetObjectAccessControlPolicy` 方法可以讀取或設定現有儲存貯體或物件的存取控制政策。如需詳細資訊，請參閱這些方法的詳細說明。

附錄 b：驗證請求（AWS 簽名版本 2）

Important

本節說明如何使用 AWS 簽章版本 2 驗證要求。Signature 第 2 版開始停止支援 (已淘汰)，Amazon S3 再來只接受使用 Signature 第 4 版簽署的 API 請求。如需詳細資訊，請參閱「AWS Amazon S3 的簽名版本 2 已關閉 (已淘汰)」

所有簽名版本 4 都受到支持 AWS 區域，並且它是新區域支持的唯一版本。如需詳細資訊，請參閱 Amazon 簡單儲存服務 API 參考中的驗證請求 (AWS 簽名版本 4)。

Amazon S3 可讓您識別簽署請求是使用哪個 API 簽章版本。您必須識別是否有任何工作流程使用 Signature 第 2 版進行簽署，並將這些工作流程升級為使用 Signature 第 4 版，以避免對您的業務造成影響。

- 如果您使用的是 CloudTrail 事件日誌（建議選項），請參閱使用識別 Amazon S3 簽名版本 2 請求 CloudTrail如何查詢和識別此類請求。
- 如果您使用 Amazon S3 伺服器存取日誌，請參閱使用 Amazon S3 存取日誌來識別簽章第 2 版請求

主題

- 使用 REST API 驗證範例
- 簽署與驗證 REST 要求
- 使用 POST 進行瀏覽器上傳 (AWS 簽名版本 2)

使用 REST API 驗證範例

使用 REST 存取 Amazon S3 時，必須在您的請求中提供下列項目，才可驗證請求：

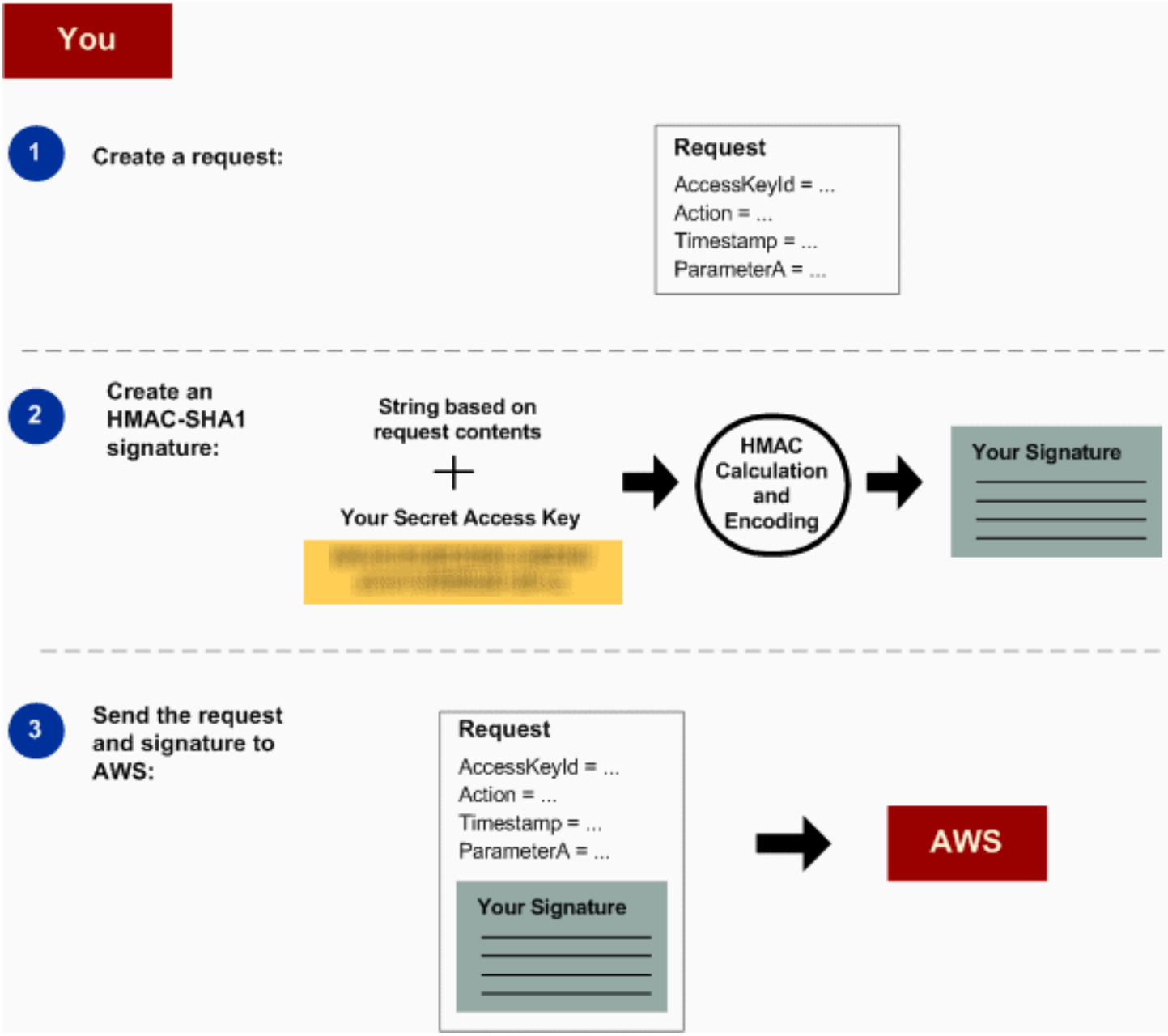
請求元素

- AWS 存取金鑰 ID — 每個要求都必須包含您用來傳送請求之身分的存取金鑰 ID。
- 簽章 – 每個請求都必須包含有效的請求簽章，否則會拒絕請求。

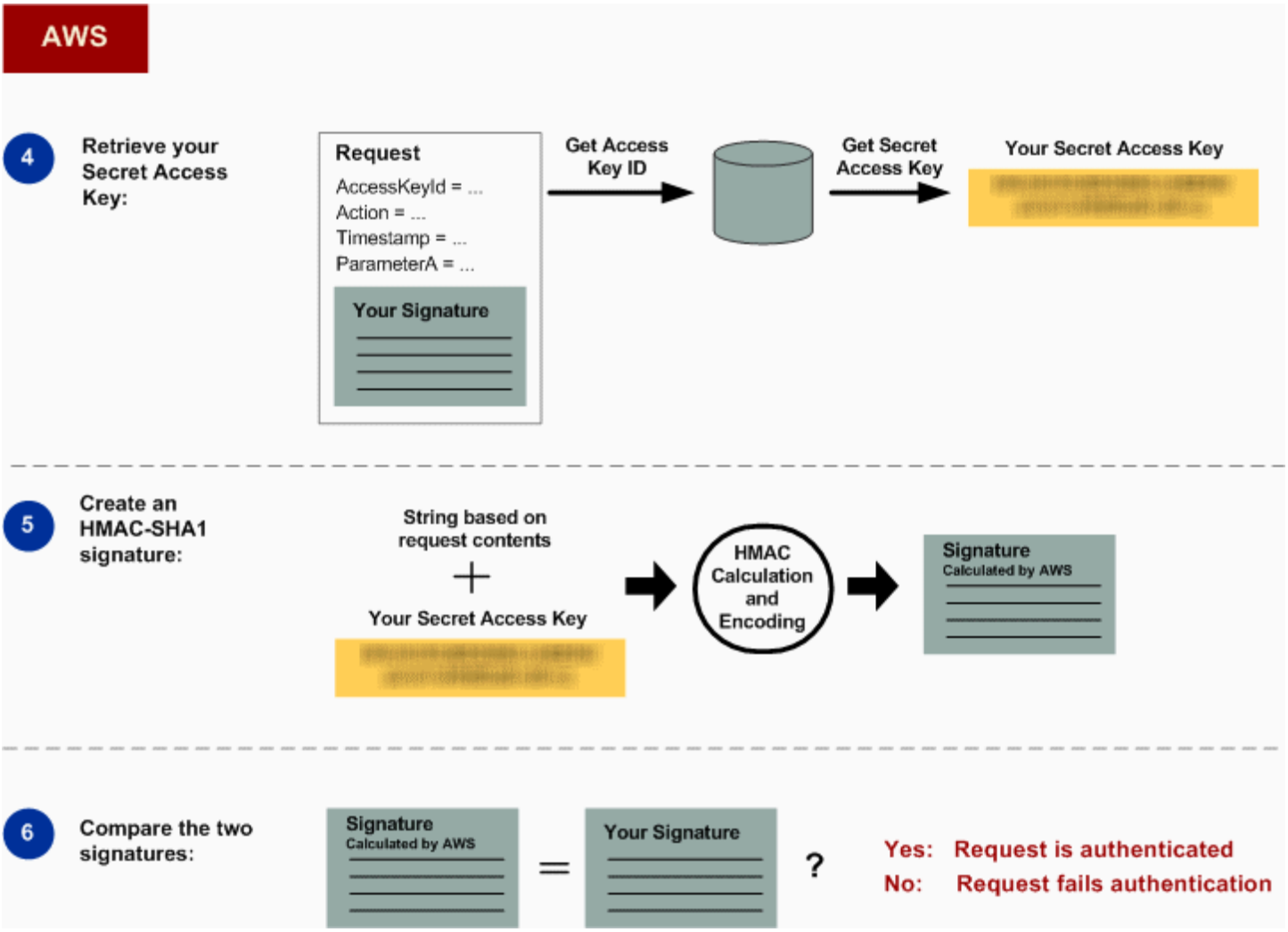
  使用您的私密存取金鑰即可計算出請求簽章，而該私密存取金鑰只有您和 AWS 知道。
- 時間戳記 – 每個請求都必須包含建立請求的日期和時間，以 UTC 字串表示。
- 日期 – 每個請求都必須包含請求的時間戳記。

  視目前使用的 API 動作之不同，您可以改為提供要求過期日期與時間，或是與時間戳記一起提供。請參閱特定動作的驗證主題，以了解所需要的資訊為何。

以下是對傳給 Amazon S3 的請求進行身分驗證的一般步驟。假設您已有必要的安全性登入資料、存取金鑰 ID，以及私密存取金鑰。



1	建構要求 AWS。
2	使用私密存取金鑰計算簽章。
3	將請求傳送至 Amazon S3。在要求中放入您的存取金鑰 ID 及簽章。Amazon S3 會執行接下來的三個步驟。



4	Amazon S3 使用存取金鑰 ID 來查詢您的私密存取金鑰。
5	Amazon S3 按照您對請求中傳送的簽章所使用的計算演算法，採用同樣的演算法來計算請求資料及私密存取金鑰中的簽章。
6	如果 Amazon S3 產生的簽章與請求中傳送的簽章相符，該請求即視為真實。若比對失敗，則會捨棄請求，Amazon S3 會傳回錯誤回應。

詳細的驗證資訊

如需 REST 驗證的詳細資訊，請參閱「簽署與驗證 REST 要求」。

簽署與驗證 REST 要求

主題

> **Note**
>
> 本主題說明如何使用 Signature 第 2 版驗證要求。Amazon S3 現在支援最新的 Signature 第 4 版。所有區域都支援此最新的 Signature 版本，而且 2014 年 1 月 30 日以後的任何新區域只會支援 Signature 第 4 版。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的身分身分驗證請求 (AWS Signature 第 4 版)。

身分驗證是向系統表明身分的程序。身分是 Amazon S3 存取控制決策中很重要的因素。要求的允許或拒絕部分取決於要求者的身分。例如，建立儲存貯體的權利會保留給已註冊的開發人員，而 (根據預設) 建立儲存貯體中物件的權利則保留給討論中之儲存貯體的擁有者。身為開發人員，您會提出要求來叫用這些權限，因此您需要透過驗證要求，向系統表明身分。本節將告訴您如何做到。

> **Note**
>
> 本節中的內容並不適用於 HTTP POST。如需詳細資訊，請參閱「使用 POST 進行瀏覽器上傳 (AWS 簽名版本 2)」。

Amazon S3 REST API 採用基於金鑰式 HMAC (雜湊訊息驗證碼) 的自訂 HTTP 機制進行身分驗證。若要驗證要求，請先將要求的選取元素串連成一個字串。然後，使用您的 AWS 私密存取金鑰計算該字串

的 HMAC。我們將此程序通稱為「簽署要求」，並將 HMAC 演算法的輸出通稱為簽章，因為它模擬實際簽章的安全屬性。最後，請使用本節中所述的語法，將此簽章新增為要求的參數。

當系統收到經過身分驗證的請求時，它會擷取您聲稱擁有的 AWS 私密存取金鑰，並將它同樣用在運算所收到訊息的簽章。然後，它會將所計算的簽章與要求者提供的簽章進行比較。如果兩個簽章相符，系統會認定申請者一定具有 AWS 私密存取金鑰的存取權，因此獲得金鑰核發對象委託人的授權行事。如果兩個簽章不符，則會捨棄要求，且系統會回應錯誤訊息。

Example 已驗證的 Amazon S3 REST 請求

```
GET /photos/puppy.jpg HTTP/1.1
Host: awsexamplebucket1.us-west-1.s3.amazonaws.com
Date: Tue, 27 Mar 2007 19:36:42 +0000

Authorization: AWS AKIAIOSFODNN7EXAMPLE:
qgk2+6Sv9/oM7G3qLEjTH1a1l1g=
```

使用臨時安全登入資料

如果您使用暫時性安全登入資料簽署要求 (請參閱「提出要求」)，您必須新增 `x-amz-security-token` 標頭，以在您的要求中包含對應的安全字符。

當您使用 AWS Security Token Service API 取得暫時性安全登入資料時，回應會包含暫時性安全登入資料與工作階段字符。當您將請求傳送至 Amazon S3 時，您會在 `x-amz-security-token` 標頭中提供此工作階段字符值。如需 IAM 提供之 AWS Security Token Service API 的相關資訊，請移至《AWS Security Token Service API 參考指南》中的 Action (動作)。

Authentication 標頭

Amazon S3 REST API 使用標準 HTTP `Authorization` 標頭來傳遞身分驗證資訊。(標準標頭的名稱並不適當，因為它傳遞身分驗證資訊，而不是身分驗證)。根據 Amazon S3 身分驗證機制，Authorization 標頭的格式如下：

```
Authorization: AWS AWSAccessKeyId:Signature
```

當開發人員註冊時，系統會將 AWS 存取金鑰 ID 與 AWS 私密存取金鑰核發給他們。進行要求身分驗證時，`AWSAccessKeyId` 元素會識別用來計算簽章的存取金鑰 ID；間接來說，就是提出要求的開發人員。

`Signature` 元素是要求中所選元素的 RFC 2104 HMAC-SHA1，因此 Authorization 標頭的 `Signature` 部分會因要求而有所不同。如果由系統計算的請求簽章與請求中所包含的 `Signature` 相

符，即證明申請者擁有 AWS 私密存取金鑰。此要求接著會以金鑰核發對象的開發人員身分 (具授權) 進行處理。

下列虛擬語法說明 Authorization 要求標頭的建構 (在此範例中，\n 表示 Unicode 字碼指標 U+000A，通常稱為新行字元)。

```
Authorization = "AWS" + " " + AWSAccessKeyId + ":" + Signature;

Signature = Base64( HMAC-SHA1( UTF-8-Encoding-Of(YourSecretAccessKey), UTF-8-Encoding-
Of( StringToSign ) ) );

StringToSign = HTTP-Verb + "\n" +
 Content-MD5 + "\n" +
 Content-Type + "\n" +
 Date + "\n" +
 CanonicalizedAmzHeaders +
 CanonicalizedResource;

CanonicalizedResource = [ "/" + Bucket ] +
 <HTTP-Request-URI, from the protocol name up to the query string> +
 [ subresource, if present. For example "?acl", "?location", or "?logging"];

CanonicalizedAmzHeaders = <described below>
```

HMAC-SHA1 是 RFC 2104 - Keyed-Hashing for Message Authentication 所定義的演算法。該演算法接受兩個位元組字串、一個金鑰與一則訊息的輸入。進行 Amazon S3 請求身分驗證時，請使用您的 AWS 私密存取金鑰 (YourSecretAccessKey) 作為金鑰，並使用 UTF-8 編碼的 StringToSign 作為訊息。HMAC-SHA1 的輸出也是位元組字串，稱為 Digest。Signature 要求參數是由編碼此 Digest 的 Base64 所建構。

將要求標準化以供簽署

之前提到，當系統收到經過驗證的要求時，它會將計算的要求簽章與要求中 StringToSign 所提供的簽章進行比較。因此，您必須使用 Amazon S3 所用的相同方法來計算簽章。我們將此以一致格式提出簽署請求的程序稱為標準化。

構建元素 CanonicalizedResource

CanonicalizedResource 代表所要請求的 Amazon S3 資源。您可以透過下列方式針對 REST 要求建構此元素：

啟動程序

1	從空字串 ("") 開始。
2	如果要求使用 HTTP Host 標頭 (虛擬託管型) 指定儲存貯體，請在儲存貯體名稱前面加上 "/" (例如 "/bucketname")。針對路徑型要求及未定址儲存貯體的要求，則不用執行任何操作。如需虛擬託管型要求的詳細資訊，請參閱「儲存貯體的虛擬託管」。 針對虛擬託管型請求 "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg"，CanonicalizedResource 會是 "/awsexamplebucket1"。 針對路徑型請求 "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg"，CanonicalizedResource 會是 ""。
3	附加未解碼之 HTTP 要求 URI 的路徑部分，一直到 (但不包括) 查詢字串。 針對虛擬託管型請求 "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg"，CanonicalizedResource 會是 "/awsexamplebucket1/photos/puppy.jpg"。 針對路徑型的請求 "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg"，CanonicalizedResource 會是 "/awsexamplebucket1/photos/puppy.jpg"。此時，虛擬託管型要求與路徑型要求的 CanonicalizedResource 會相同。 針對未定址儲存貯體的請求 (例如 GET Service)，請附加 "/"。
4	如果請求定址一項子資源 (例如 ?versioning 、?location 、?acl、?lifecycle 或 ?versionid)，請附加該子資源、其值 (若有) 與問號。請注意，如有多項子資源，則必須按詞典編纂方式依子資源名稱排序並以 '&' 分隔，例如 ?acl&versionId=*value*。 構建 CanonicalizedResource 元素時必須包含的子資源是 ACL，生命週期，位置，日誌記錄，通知，partNumber，策略，requestPayment，UpLoaDid，uploadId，versionId，版本，版本和網站。 如果請求指定查詢字串參數來覆寫回應標頭值 (請參閱 Get 物件)，請附加查詢字串參數及其值。簽署時，您不會編碼這些值；不過，您必須在提出要求時編碼這些參數值。GET 要求中的查詢字串參數包含 response-content-type 、response-content-language 、response-expires 、response-cache-control 、response-content-disposition 與 response-content-encoding 。 當您 CanonicalizedResource 為多物件刪除請求建立時，必須包含delete查詢字串參數。

來自 HTTP 請求 URI 的 CanonicalizedResource 元素應該按照字面上的方式進行簽名，因為它們出現在 HTTP 請求中，包括 URL 編碼中繼字符。

CanonicalizedResource 可能與 HTTP 要求 URI 不同。特別是如果您的要求使用 HTTP Host 標頭指定儲存貯體，該儲存貯體不會出現在 HTTP 要求 URI 中。不過，CanonicalizedResource 會繼續包含該儲存貯體。查詢字串參數也可能出現在要求 URI 中，但未包含在 CanonicalizedResource 中。如需詳細資訊，請參閱「儲存貯體的虛擬託管」。

構建元素 CanonicalizedAmzHeaders

若要建構的 CanonicalizedAmzHeaders 部分StringToSign，請選取以 'x-amz-' 開頭的所有 HTTP 要求標頭 (使用不區分大小寫的比較)，然後使用下列程序。

CanonicalizedAmzHeaders 程序

1	將每個 HTTP 標頭名稱轉換成小寫。例如，'X-Amz-Date ' 會變成 'x-amz-date '。
2	按詞典編纂方式依標頭名稱排序標頭集合。
3	按照 RFC 2616 第 4.2 節規定將具有相同名稱的標題字段合併為一個「標題名稱：comma-separated-value-list」對，值之間沒有任何空格。例如，兩個中繼資料標頭 'x-amz-meta-username: fred ' 與 'x-amz-meta-username: barney ' 會合併成單一標頭 'x-amz-meta-username: fred,barney '。
4	「展開」跨多行的長標頭 (如 RFC 2616 第 4.2 節所允許)，方法是以單一空格取代摺疊的空格 (包括換行符號)。
5	修剪標頭中冒號前後的任何空格。例如，標頭 'x-amz-meta-username: fred,barney ' 會變成 'x-amz-meta-username:fred,barney '
6	最後，將新行字元 (U+000A) 附加至所產生之清單中的每個標準化標頭。通過將此列表中的所有頭文 CanonicalizedResource 件連接成一個字符串來構造元素。

位置與命名的 HTTP 頭元素 StringToSign

StringToSign 的前幾個標頭元素 (Content-Type、日期和 Content-MD5) 屬於位置性標頭。StringToSign 不包含這些標頭的名稱，僅包含其在請求中的值。相對的，'x-amz-' 元素是具名的。其標頭名稱與標頭值都會出現在 StringToSign 中。

如果 StringToSign 定義中所需的位置性標頭不會出現在您的要求中 (例如 Content-Type 或 Content-MD5 對 PUT 要求為選用但對 GET 要求則沒有意義)，請以空字串 ("") 取代該位置。

時間戳記需求

經過驗證的要求必須具有有效的時間戳記 (使用 HTTP Date 標頭或 x-amz-date 替代項)。此外，經過驗證的請求所包含的用戶端時間戳記，必須在收到要求後不超過 Amazon S3 系統時間的 15 分鐘。否則，要求會失敗並出現 RequestTimeTooSkewed 錯誤碼。這些限制的用意是為了降低要求可能遭對手攔截而被重播的可能性。若要增強防範竊聽的保護，請使用 HTTPS 傳輸經過驗證的要求。

> **Note**
>
> 要求日期的驗證限制僅適用於未使用查詢字串身分驗證之經過驗證的要求。如需詳細資訊，請參閱「查詢字串要求身分驗證替代項」。

某些 HTTP 用戶端程式庫無法設定要求的 Date 標頭。如果您無法在標準化標頭中包含 'Date' 標頭的值，您可以改用 'x-amz-date' 標頭來設定要求的時間戳記。x-amz-date 標頭的值必須是其中一個 RFC 2616 格式 (http://www.ietf.org/rfc/rfc2616.txt)。如果要求中有 x-amz-date 標頭，系統會在計算要求簽章時略過任何 Date 標頭。因此，如果您包含 x-amz-date 標頭，請在建構 Date 時，使用空字串來表示 StringToSign。如需範例，請參閱下一節。

身分驗證範例

本節中的範例使用下表中的 (非工作) 登入資料。

參數	值
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecret AccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

在範例 StringToSign 中，格式並不重要；而 \n 表示 Unicode 字碼指標 U+000A，通常稱為新行字元。此外，範例使用 "+0000" 來指定時區。您可以改用 "GMT" 來指定時區，但範例中將會顯示不同的簽章。

Object GET

此範例會從 awsexamplebucket1 儲存貯體取得物件。

請求	StringToSign
`GET /photos/puppy.jpg HTTP/1.1` `Host: awsexamplebucket1.us-west-1.s3.amazonaws.com` `Date: Tue, 27 Mar 2007 19:36:42 +0000` `Authorization: AWS AKIAIOSFODNN7EXAMPLE:qgk2+6Sv9/oM7G3qLEjTH1a1l1g=`	`GET\n` `\n` `\n` `Tue, 27 Mar 2007 19:36:42 +0000\n` `/awsexamplebucket1/photos/puppy.jpg`

請注意， CanonicalizedResource 包括存儲桶名稱，但 HTTP 請求 URI 不包含。(儲存貯體是由 Host 標頭指定)。

> **Note**
>
> 下列 Python 指令碼會使用提供的參數計算上述簽章。您可以使用此腳本來構建自己的簽名，並 StringToSign 在適當情況下替換密鑰。

```
import base64
import hmac
from hashlib import sha1

access_key = 'AKIAIOSFODNN7EXAMPLE'.encode("UTF-8")
secret_key = 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'.encode("UTF-8")

string_to_sign = 'GET\n\n\nTue, 27 Mar 2007 19:36:42 +0000\n/awsexamplebucket1/photos/puppy.jpg'.encode("UTF-8")
signature = base64.b64encode(
                                hmac.new(
                                        secret_key, string_to_sign, sha1
                                        ).digest()
                                ).strip()


print(f"AWS {access_key.decode()}:{signature.decode()}")
```

Object PUT

這個例子將物件放入 awsexamplebucket1 儲存貯體中。

請求	StringToSign
`PUT /photos/puppy.jpg HTTP/1.1` `Content-Type: image/jpeg` `Content-Length: 94328` `Host: awsexamplebucket1.s3.us-wes` `t-1.amazonaws.com` `Date: Tue, 27 Mar 2007 21:15:45 +0000` `Authorization: AWS AKIAIOSFODNN7EXAMP` `LE:` `iqRzw+ileNPu1fhspnRs8nOjjIA=`	`PUT\n` `\n` `image/jpeg\n` `Tue, 27 Mar 2007 21:15:45 +0000\n` `/awsexamplebucket1/photos/puppy.jpg`

請注意要求和中的內容類型標頭。 StringToSign另請注意，內容 MD5 在中保留空白 StringToSign，因為它不存在於請求中。

清單

此範例列出 awsexamplebucket1 儲存貯體的內容。

請求	StringToSign
`GET /?prefix=photos&max-keys=50&marker=puppy` ` HTTP/1.1` `User-Agent: Mozilla/5.0` `Host: awsexamplebucket1.s3.us-west-1.amazo` `naws.com` `Date: Tue, 27 Mar 2007 19:42:41 +0000` `Authorization: AWS AKIAIOSFODNN7EXAMPLE:` `m0WP8eCtspQl5Ahe6L1SozdX9YA=`	`GET\n` `\n` `\n` `Tue, 27 Mar 2007 19:42:41` ` +0000\n` `/awsexamplebucket1/`

請注意查詢字串參數 CanonicalizedResource 和不存在的結尾斜線。

擷取

此範例會擷取 'awsexamplebucket1' 儲存貯體的存取控制政策子資源。

請求	StringToSign
`GET /?acl HTTP/1.1` `Host: awsexamplebucket1.s3.us-west-1.amazonaws.com` `Date: Tue, 27 Mar 2007 19:44:46 +0000` `Authorization: AWS AKIAIOSFODNN7EXAMPLE:82ZHiFIjc+WbcwFKGUVEQspPn+0=`	`GET\n` `\n` `\n` `Tue, 27 Mar 2007 19:44:46 +0000\n` `/awsexamplebucket1/?acl`

請注意子資源查詢字串參數如何包含在中 CanonicalizedResource。

Delete

此範例會使用路徑型與 Date 替代項，從 'awsexamplebucket1' 儲存貯體中刪除物件。

請求	StringToSign
`DELETE /awsexamplebucket1/photos/puppy.jpg HTTP/1.1` `User-Agent: dotnet` `Host: s3.us-west-1.amazonaws.com` `Date: Tue, 27 Mar 2007 21:20:27 +0000` `x-amz-date: Tue, 27 Mar 2007 21:20:26 +0000` `Authorization: AWS AKIAIOSFODNN7EXAMPLE:XbyTlbQdu9Xw5o8P4iMwPktxQd8=`	`DELETE\n` `\n` `\n` `Tue, 27 Mar 2007 21:20:26 +0000\n` `/awsexamplebucket1/photos/puppy.jpg`

請注意我們如何使用指定日期的替代 x-amz-date " 方法（因為我們的客戶端庫阻止我們設置日期，比如說）。在此情況下，`x-amz-date` 會優先於 `Date` 標頭。因此，簽章中的日期項目必須包含 `x-amz-date` 標頭的值。

上傳

此範例會將物件連同中繼資料一起上傳至 CNAME 型虛擬託管儲存貯體。

請求	StringToSign
`PUT /db-backup.dat.gz HTTP/1.1` `User-Agent: curl/7.15.5` `Host: static.example.com:8080` `Date: Tue, 27 Mar 2007 21:06:08 +0000` `x-amz-acl: public-read` `content-type: application/x-download` `Content-MD5: 4gJE4saaMU4BqNR0kLY+lw==` `X-Amz-Meta-ReviewedBy: joe@example.com` `X-Amz-Meta-ReviewedBy: jane@exam` `ple.com` `X-Amz-Meta-FileChecksum: 0x02661779` `X-Amz-Meta-ChecksumAlgorithm: crc32` `Content-Disposition: attachment;` ` filename=database.dat` `Content-Encoding: gzip` `Content-Length: 5913339` *` Authorization: AWS AKIAIOSFODNN7EXAMP`* *`LE:`* *`jtBQa0Aq+DkULFI8qrpwIjGEx0E=`*	`PUT\n` `4gJE4saaMU4BqNR0kLY+lw==\n` `application/x-download\n` `Tue, 27 Mar 2007 21:06:08 +0000\n` `x-amz-acl:public-read\n` `x-amz-meta-checksumalgorithm:c` `rc32\n` `x-amz-meta-filechecksum:0x026` `61779\n` `x-amz-meta-reviewedby:` `joe@example.com,jane@example.com` `\n` `/static.example.com/db-backup.dat` `.gz`

請注意 'x-amz-' 標頭如何排序、修剪多餘空格及轉換成小寫。另請注意，已聯結多個具有相同名稱的標頭，並使用逗號分隔其值。

請注意，只有 Content-Type 與 Content-MD5 HTTP 實體標頭會出現在 StringToSign 中。其他 Content-* 實體標頭則否。

同樣地，請注意 CanonicalizedResource 包含儲存貯體名稱，但 HTTP 要求 URI 則否。(儲存貯體是由 Host 標頭指定)。

列出我的所有儲存貯體

請求	StringToSign
`GET / HTTP/1.1` `Host: s3.us-west-1.amazonaws.com` `Date: Wed, 28 Mar 2007 01:29:59 +0000` `Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdERIC03wnaRNKh6OqZehG9s=`	`GET\n` `\n` `\n` `Wed, 28 Mar 2007 01:29:59 +0000\n` `/`

Unicode 金鑰

請求	StringToSign
`GET /dictionary/fran%C3%A7ais/pr%c3%a9f%c3%a8re HTTP/1.1` `Host: s3.us-west-1.amazonaws.com` `Date: Wed, 28 Mar 2007 01:49:49 +0000` `Authorization: AWS AKIAIOSFODNN7EXAMPLE:DNEZGsoieTZ92F3bUfSPQcbGmlM=`	`GET\n` `\n` `\n` `Wed, 28 Mar 2007 01:49:49 +0000\n` `/dictionary/fran%C3%A7ais/pr%c3%a9f%c3%a8re`

> **Note**
>
> 衍生自要求 URI 的 `StringToSign` 元素必須依原狀採用，包括 URL 編碼與大寫。

REST 要求簽署問題

當 REST 要求身分驗證失敗時，系統會以 XML 錯誤文件回應要求。此錯誤文件中所包含的資訊是為了協助開發人員診斷問題。特別是 `StringToSign` 錯誤文件中的 `SignatureDoesNotMatch` 元素會告訴您系統正在使用的要求標準化。

某些工具組會在您事先不知情的情況下，以無訊息方式插入標頭，例如在 PUT 期間加入標頭 `Content-Type`。在大部分情況下，插入的標頭值仍然保持不變，因此您可以使用 Ethereal 或 tcpmon 等工具來探索遺漏的標頭。

查詢字串要求身分驗證替代項

您可以將必要資訊當作查詢字串參數來傳遞，以驗證特定類型的要求，而不是使用 Authorization HTTP 標頭。這有助於讓第三方瀏覽器直接存取您的私有 Amazon S3 資料，而不需要代理請求。此概念是建構「預先簽章」的請求，然後將其編碼為最終使用者可以擷取的 URL。此外，您可以指定過期時間，限制預先簽章的請求。

如需有關使用查詢參數來驗證請求的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的對請求進行身分驗證：使用查詢參數 (AWS Signature 第 4 版)。關於使用 AWS 開發套件產生預先簽章 URL 的範例，請參閱「使用預先簽章的 URL 來共用物件」。

建立簽章

以下為經過查詢字串驗證的 Amazon S3 REST 請求範例。

```
GET /photos/puppy.jpg
?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Expires=1141889120&Signature=vjbyPxybdZaNmGa
%2ByT272YEAiv4%3D HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
Date: Mon, 26 Mar 2007 19:37:58 +0000
```

查詢字串要求身分驗證方法不需要任何特殊 HTTP 標頭。相反地，必要的身分驗證元素會指定為查詢字串參數。

查詢字串參數名稱	範例值	描述
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE	您的 AWS 存取金鑰 ID。指定用來簽章請求的 AWS 私密存取金鑰，間接來說，就是提出請求的開發人員身分。
Expires	1141889120	簽章的過期時間，指定為自 epoch (1970 年 1 月 1 日 00:00:00 UTC) 起經過的秒數。在此時間 (根據伺服器) 以後收到的要求會遭到拒絕。
Signature	vjbyPxybdZaNmGa%2ByT272YEAiv4%3D	的 HMAC-SHA1 的 Base64 編碼的網址編碼。 StringToSign

查詢字串要求身分驗證方法與一般方法稍有不同，但只是 Signature 要求參數與 StringToSign 元素之間的格式不同。下列虛擬語法說明查詢字串要求身分驗證方法。

```
Signature = URL-Encode( Base64( HMAC-SHA1( YourSecretAccessKey, UTF-8-Encoding-
Of( StringToSign ) ) ) );

StringToSign = HTTP-VERB + "\n" +
    Content-MD5 + "\n" +
    Content-Type + "\n" +
    Expires + "\n" +
    CanonicalizedAmzHeaders +
    CanonicalizedResource;
```

YourSecretAccessKey 是當您註冊成為 Amazon Web Services 開發人員時，由 Amazon 指派給您的 AWS 私密存取金鑰 ID。請注意 Signature 如何經過 URL 編碼，以便適當放在查詢字串中。另請注意，在 StringToSign 中，HTTP Date 位置性元素已取代為 Expires。CanonicalizedAmzHeaders 與 CanonicalizedResource 相同。

> **Note**
>
> 在查詢字串身分驗證方法中，您不會使用 Date 或 x-amz-date request 標頭來計算 StringToSign。

查詢字串要求身分驗證

請求	StringToSign
`GET /photos/puppy.jpg?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&` `    Signature=NpgCjnDzrM%2BWFzoENXmpNDUsSn8%3D&` `    Expires=1175139620 HTTP/1.1` `Host: awsexamplebucket1.s3.us-west-1.amazonaws.com`	`GET\n` `\n` `\n` `1175139620\n` `/awsexamplebucket1/photos/puppy.jpg`

假設當瀏覽器提出 GET 要求時未提供 Content-MD5 或 Content-Type 標頭，也未設定任何 x-amz- 標頭，因此 `StringToSign` 的這些部分會保留空白。

使用 Base64 編碼

HMAC 要求簽章必須經過 Base64 編碼。Base64 編碼會將簽章轉換成可附加至要求的簡單 ASCII 字串。簽章字串中可能出現的字元 (例如加號 (+)、正斜線 (/) 與等號 (=)) 若用於 URI，則必須經過編碼。例如，如果驗證碼包含加號 (+)，請在要求中將它編碼為 %2B。正斜線會編碼為 %2F，而等號則會編碼為 %3D。

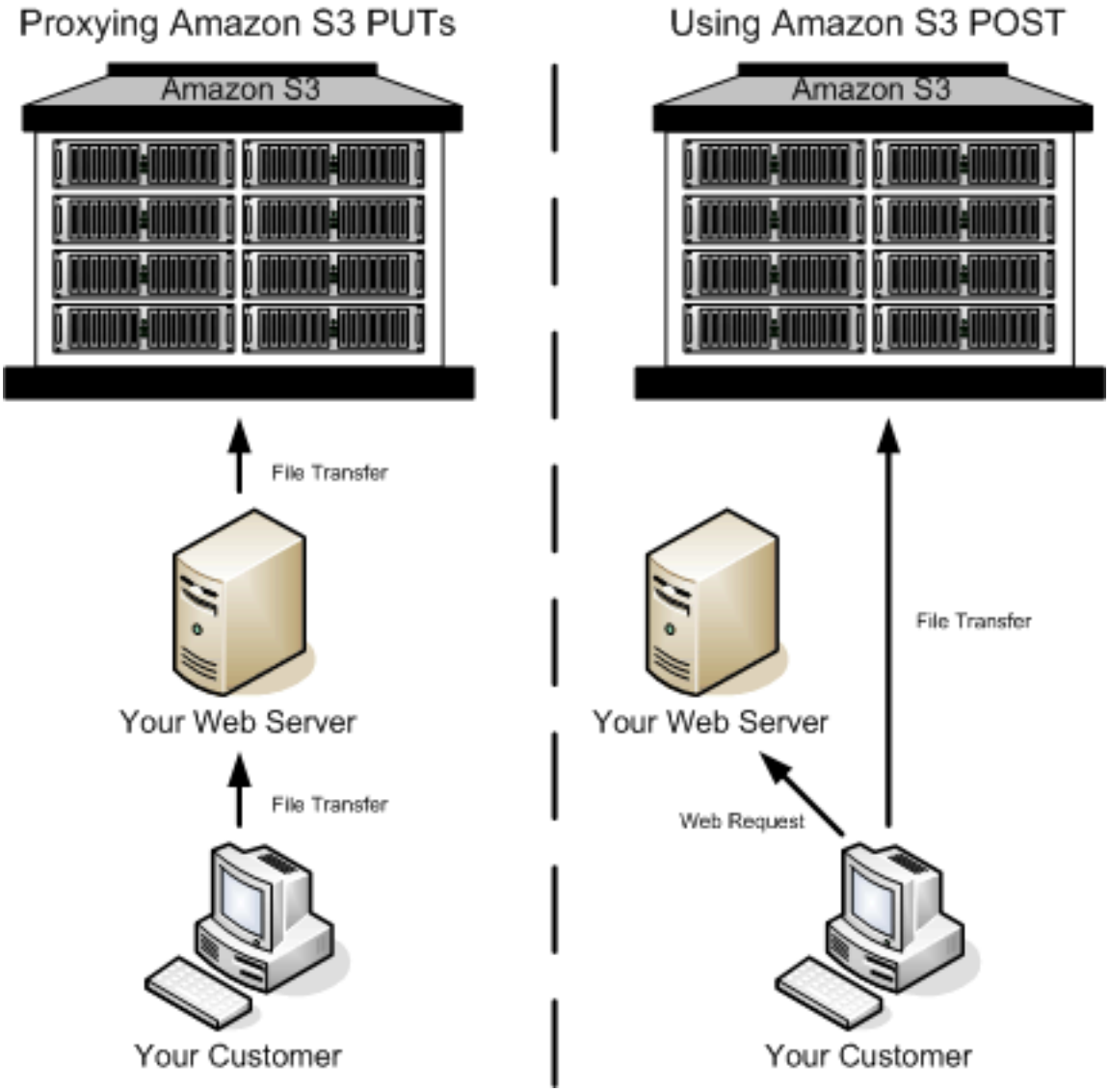
如需 Base64 編碼的範例，請參閱 Amazon S3 身分驗證範例。

使用 POST 進行瀏覽器上傳 (AWS 簽名版本 2)

Amazon S3 支援 POST，可讓您的使用者直接將內容上傳至 Amazon S3。POST 旨在簡化上傳、降低上傳延遲，並省下您花在應用程式上讓使用者上傳資料以存放於 Amazon S3 的成本。

> **Note**
>
> 本節中討論的要求驗證是以 AWS 簽章版本 2 為基礎，這是一種驗證對 AWS 服務之輸入 API 要求的通訊協定。
> Amazon S3 現在完全 AWS 區域支援簽名版本 4，這是一種用於驗證對 AWS 服務的輸入 API 請求的協定。此時，在 2014 年 1 月 30 日之前 AWS 區域 創建的將繼續支持以前的協議，簽名版本 2。2014 年 1 月 30 日以後的任何新區域只會支援 Signature 第 4 版，因此傳送至這些區域的所有要求都必須使用 Signature 第 4 版提出。如需詳細資訊，請參閱 Amazon 簡單儲存服務 API 參考中的使用 POST (AWS 簽名版本 4) 在以瀏覽器為基礎的上傳中驗證請求。

下圖示範如何使用 Amazon S3 POST 進行上傳。



使用 POST 進行上傳

1	使用者開啟 Web 瀏覽器並存取您的網頁。
2	您的網頁包含一個 HTTP 表單，其中含有讓使用者將內容上傳至 Amazon S3 的所有必要資訊。
3	使用者直接將內容上傳至 Amazon S3。

> **Note**
>
> POST 不支援查詢字串身分驗證。

HTML 表單（AWS 簽名版本 2）

主題

當您與 Amazon S3 通訊時，您通常會使用 REST 或 SOAP API 來執行放置、取得、刪除及其他作業。使用 POST 時，使用者經由瀏覽器直接將資料上傳至 Amazon S3，因此無法處理 SOAP API 或建立 REST PUT 請求。

> **Note**
>
> HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得 SOAP。SOAP 不支援新的 Amazon S3 功能。我們建議您使用 REST API 或 AWS 開發套件，而不是使用 SOAP。

若要允許使用者使用瀏覽器將內容上傳至 Amazon S3，您可以使用 HTML 表單。HTML 表單是由一個表單宣告與多個表單欄位所組成。表單宣告包含要求的高階資訊。表單欄位包含要求的詳細資訊，以及用來對要求進行身分驗證及確保其符合所指定條件的政策。

> **Note**
>
> 表單資料與邊界 (檔案內容除外) 不得超過 20 KB。

本節說明如何使用 HTML 表單。

HTML 表單編碼

表單與政策必須經過 UTF-8 編碼。您可以在 HTML 標題中指定 UTF-8 編碼，或將它指定作為要求標頭來套用至表單。

> **Note**
>
> HTML 表單宣告不接受查詢字串身分驗證參數。

以下為 HTML 標題中的 UTF-8 編碼範例：

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

以下為要求標頭中的 UTF-8 編碼範例：

```
Content-Type: text/html; charset=UTF-8
```

HTML 表單宣告

表單宣告有三個元件：動作、方法與封閉類型。若有任何值未正確設定，要求會失敗。

動作指定處理要求的 URL，必須將它設定為儲存貯體的 URL。例如，如果儲存貯體的名稱是 awsexamplebucket1，而區域是美國西部 (加利佛尼亞北部)，則 URL 為 https://awsexamplebucket1.s3.us-west-1.amazonaws.com/。

> **Note**
>
> 金鑰名稱是在表單欄位中指定。

方法必須是 POST。

封閉類型 (enctype) 必須加以指定，而且針對檔案上傳與文字區域上傳都必須設定為 multipart/form-data。如需詳細資訊，請參閱 RFC 1867。

Example

下列範例是儲存貯體 "awsexamplebucket1" 的表單宣告。

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"

enctype="multipart/form-data">
```

HTML 表單欄位

下表說明可在 HTML 表單中使用的欄位。

> **Note**
>
> 系統會自動以使用者提供的檔案名稱來取代變數 `${filename}`，所有表單欄位都可識別此變數。如果瀏覽器或用戶端提供檔案的完整或部分路徑，則只會使用最後一個正斜線 (/) 或反斜線 (\) 後面的文字。例如，"C:\Program Files\directory1\file.txt" 會解譯為 "file.txt"。若未提供檔案或檔案名稱，則會以空字串取代變數。

欄位名稱	描述	必要
`AWSAccessKeyId`	值區擁有者的 AWS 存取金鑰 ID，該值區擁有者會針對符合原則中限制集合的要求授與匿名使用者存取權。如果要求包含政策文件，則需要此欄位。	有條件
`acl`	Amazon S3 存取控制清單 (ACL)。如果指定了無效的存取控制清單，則會產生錯誤。如需 ACL 的詳細資訊，請參閱「存取控制清單 (ACL)」。 類型：字串 預設值：private 有效值: `private | public-read | public-read-write | aws-exec-`	否

欄位名稱	描述	必要
	`read	authenticated-read	bucket-owner-read	bucket-owner-full-control`	
`Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires`	REST 特定標頭。如需詳細資訊，請參閱 PUT 物件。	否
`key`	上傳的金鑰名稱。 若要使用使用者提供的檔名，請使用 ${filename} 變數。例如，如果使用者 Betty 上傳檔案 lolcatz.jpg 且您指定 /user/betty/${filename}，則檔案會另存為 /user/betty/lolcatz.jpg。 如需詳細資訊，請參閱「使用物件中繼資料」。	是
`policy`	說明要求中所允許之內容的安全政策。未使用安全政策的要求會視為匿名，只有在可公開寫入的儲存貯體上才會成功。	否

欄位名稱	描述	必要
success_action_redirect, redirect	用戶端成功上傳時被重新導向的目標 URL。Amazon S3 會將 bucket、key 和 etag 值當作查詢字串參數附加至 URL。 若未指定 success_action_redirect，Amazon S3 會傳回 success_action_status 欄位中所指定的空白文件類型。 如果 Amazon S3 無法解譯 URL，則會忽略 欄位。 如果上傳失敗，Amazon S3 會顯示錯誤，而不會將使用者重新導向至 URL。 如需詳細資訊，請參閱重新導向。 **Note** redirect 欄位名稱已被取代，未來將會移除對 redirect 欄位名稱的支援。	否

欄位名稱	描述	必要
success_action_status	成功上傳時，傳回給用戶端的狀態碼 (若未指定 success_action_redirect)。 有效值為 200、201 或 204 (預設值)。 如果值設定為 200 或 204，Amazon S3 會傳回狀態碼為 200 或 204 的空白文件。 如果值設定為 201，Amazon S3 會傳回狀態碼為 201 的 XML 文件。如需 XML 文件內容的相關資訊，請參閱 POST 物件。 如果值未設定或設定為無效的值，Amazon S3 會傳回狀態碼為 204 的空白文件。 **Note** Adobe Flash Player 的某些版本無法正確處理具有空白主體的 HTTP 回應。若要支援經由 Adobe Flash 上傳，建議將 success_action_status 設定為 201。	否
signature	通過使用對應於提AWSAccessKeyId供的秘密訪問密鑰構建的 HMAC 簽名。如果要求包含政策文件，則需要此欄位。 如需詳細資訊，請參閱「適用於 Amazon S3 的 Identity and Access Management」。	有條件

欄位名稱	描述	必要
x-amz-security-token	工作階段登入資料所使用的安全字符 如果請求使用 Amazon， DevPay 則需要兩個x-amz-security-token 表單字段：一個用於產品令牌，另一個用於用戶令牌。 如果要求使用工作階段登入資料，則需要一個 x-amz-security-token 表單。如需詳細資訊，請參閱《IAM 使用者指南》中的臨時安全登入資料。	否
其他字段名稱前綴 x-amz-meta 為-	使用者指定的中繼資料。 Amazon S3 不會驗證或使用此資料。 如需詳細資訊，請參閱 PUT 物件。	否
file	檔案或文字內容。 檔案或內容必須是表單中的最後一個欄位。其下方的任何欄位都會被忽略。 您無法一次上傳多個檔案。	是

政策建構

主題

- 過期
- 條件
- 條件比對
- 字元逸出

政策是 UTF-8 與 Base64 編碼的 JSON 文件，其指定要求必須符合的條件，並用來對內容進行身分驗證。根據您設計政策文件的方式，您可以針對每次上傳、每位使用者、所有上傳，或根據符合您需求的其他設計來使用這些文件。

> **Note**
>
> 雖然您不一定要使用政策文件，但強烈建議在將儲存貯體設為可公開寫入時使用它。

以下為政策文件的範例：

```
{ "expiration": "2007-12-01T12:00:00.000Z",

  "conditions": [

    {"acl": "public-read" },

    {"bucket": "awsexamplebucket1" },

    ["starts-with", "$key", "user/eric/"],

  ]

}
```

政策文件包含過期與條件。

過期

過期元素是以 ISO 8601 UTC 日期格式來指定政策的過期日期。例如，"2007-12-01T12:00:00.000Z" 指定政策在過了 2007-12-01 午夜 UTC 後無效。政策中需要有過期資訊。

條件

政策文件中的條件會驗證上傳的物件內容。您在表單中指定的每個表單欄位 (簽名AWSAccessKeyId、檔案、策略和具有 x-ignore-前置詞的欄位名稱除外) 都必須包含在條件清單中。

> **Note**
>
> 如果您有多個同名的欄位，則必須以逗號分隔其值。例如，如果您有兩個名為 "x-amz-meta-tag" 的欄位，而第一個欄位的值為「Ninja」，第二個欄位的值為「Stallman」，則應將政策文件設定為。`Ninja,Stallman`
> 系統會展開表單中的所有變數，再驗證政策。因此，必須對展開的欄位執行所有條件比對。例如，如果將金鑰欄位設定為 `user/betty/${filename}`，您的政策可能是 `[ "starts-with", "$key", "user/betty/" ]`。請勿輸入 `[ "starts-with", "$key", "user/betty/${filename}" ]`。如需詳細資訊，請參閱「條件比對」。

下表說明政策文件條件。

元素名稱	描述
acl	指定 ACL 必須符合的條件。 支援完全符合與 `starts-with` 。
content-length-range	指定上傳內容允許的大小上下限。 支援範圍比對。
Cache-Control、Content-Type、Content-Disposition、Content-Encoding、Expires	REST 特定標頭。 支援完全符合與 `starts-with` 。
金鑰	上傳的金鑰名稱。 支援完全符合與 `starts-with` 。
success_action_redirect、redirect	用戶端成功上傳時被重新導向的目標 URL。 支援完全符合與 `starts-with` 。
success_action_status	

元素名稱	描述
	成功上傳時，傳回給用戶端的狀態碼 (若未指定 success_action_redirect)。 支援完全符合。
x-amz-security-token	Amazon DevPay 安全令牌。 每個使用 Amazon 的請求都 DevPay 需要兩個`x-amz-security-token` 表單欄位：一個用於產品權杖，另一個用於使用者權杖。因此，必須以逗號分隔其值。例如，如果使用者字符為 `eW91dHViZQ==` ，而產品字符為 `b0hnNVNKWVJIQTA=` ，您會將政策項目設定為：`{ "x-amz-security-token": "eW91dHViZQ==,b0hnNVNKWVJIQTA=" }` 。
其他字段名稱前綴 x-amz-meta為-	使用者指定的中繼資料。 支援完全符合與 `starts-with` 。

> **Note**
>
> 如果您的工具組新增其他欄位 (例如 Flash 新增檔案名稱)，則必須將它新增至政策文件。如果您可以控制此功能，請在欄位前面加上 `x-ignore-`，讓 Amazon S3 忽略此功能，這不會影響此功能的未來版本。

條件比對

下表說明條件比對類型。雖然您必須針對表單中所指定的每個表單欄位指定一個條件，但您可以藉由針對表單欄位指定多個條件，來建立更複雜的符合條件。

Condition	描述
完全符合	完全符合會確認欄位符合特定值。下列範例表示 ACL 必須設定為 public-read：

Condition	描述
	`{"acl": "public-read" }` 您也可以使用下列範例來表示 ACL 必須設定為 public-read： `[ "eq", "$acl", "public-read" ]`
開頭為	如果值必須以特定值開頭，請使用 starts-with。下列範例表示金鑰必須以 user/betty 開頭： `["starts-with", "$key", "user/betty/"]`
符合任何內容	若要設定政策以允許欄位中的任何內容，請使用 starts-with 並提供空白值。下列範例允許任何 success_action_redirect： `["starts-with", "$success_action_redirect", ""]`
指定範圍	針對接受範圍的欄位，請以逗號分隔範圍的上下限。下列範例允許 1 到 10 MB 的檔案大小： `["content-length-range", 1048579, 10485760]`

字元逸出

下表說明政策文件中必須逸出的字元。

逸出序列	描述
\\	反斜線

逸出序列	描述
\$	貨幣符號
\b	退格鍵
\f	Form Feed
\n	新行
\r	歸位字元
\t	水平標籤
\v	垂直標籤
\u*xxxx*	所有 Unicode 字元

建構簽章

步驟	描述
1	使用 UTF-8 進行政策的編碼。
2	使用 Base64 進行這些 UTF-8 位元組的編碼。
3	使用 HMAC SHA-1 透過私密存取金鑰簽署政策。
4	使用 Base64 進行 SHA-1 簽章的編碼。

如需身分驗證的相關一般資訊，請參閱「適用於 Amazon S3 的 Identity and Access Management」。

重新導向

本節說明如何處理重新導向。

一般重新導向

完成 POST 要求後，使用者會被重新導向至您在 `success_action_redirect` 欄位中指定的位置。如果 Amazon S3 無法解譯 URL，則會忽略 `success_action_redirect` 欄位。

若未指定 `success_action_redirect`，Amazon S3 會傳回 `success_action_status` 欄位中所指定的空白文件類型。

如果 POST 請求失敗，Amazon S3 會顯示錯誤，而不會提供重新導向。

上傳前重新導向

如果您的值區是使用 < CreateBucketConfiguration > 建立的，您的使用者可能需要重新導向。如果發生此情況，某些瀏覽器可能無法正確處理重新導向。這是相當少見的情況，最可能在建立儲存貯體之後立即發生。

上傳示例（AWS 簽名版本 2）

主題

- 檔案上傳
- 文字區域上傳

> **Note**
>
> 本節中討論的要求驗證是以 AWS 簽章版本 2 為基礎，這是一種驗證對 AWS 服務之輸入 API 要求的通訊協定。
> Amazon S3 現在完全 AWS 區域支援簽名版本 4，這是一種用於驗證對 AWS 服務的輸入 API 請求的協定。此時，在 2014 年 1 月 30 日之前 AWS 區域 創建的將繼續支持以前的協議，簽名版本 2。2014 年 1 月 30 日以後的任何新區域只會支援 Signature 第 4 版，因此傳送至這些區域的所有要求都必須使用 Signature 第 4 版提出。如需詳細資訊，請參閱 Amazon 簡單儲存服務 API 參考中的範例：使用 HTTP POST (使用 AWS 簽名版本 4) 進行以瀏覽器為基礎的上傳。

檔案上傳

此範例顯示建構政策與表單以用來上傳檔案附件的完整程序。

政策與表單建構

下列政策支援上傳至 Amazon S3 的 awsexamplebucket1 儲存貯體。

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

此政策需要進行下列操作：

- 上傳必須發生在 2007 年 12 月 1 日 12:00 UTC 以前。
- 內容必須上傳到 awsexamplebucket1 儲存貯體。
- 金鑰必須以 "user/eric/" 開頭。
- ACL 已設定為 public-read。
- success_action_redirect 已設定為 https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html。
- 物件是映像檔。
- 此標 x-amz-meta-uuid 籤必須設定為 143651274。
- x-amz-meta-tag 可以包含任何值。

以下為此政策的 Base64 編碼版本。

```
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN
```

使用您的登入資料建立簽章；例如，0RavWzkygo6QX9caELEqKi9kDbU= 是上述政策文件的簽章。

下列表單支援對使用此政策的 DOC-EXAMPLE-BUCKET 儲存貯體提出 POST 要求。

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
  ...
  <form action="https://DOC-EXAMPLE-BUCKET.s3.us-west-1.amazonaws.com/" method="post"
 enctype="multipart/form-data">
    Key to upload: <input type="input" name="key" value="user/eric/" /><br />
    <input type="hidden" name="acl" value="public-read" />
    <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html" />
    Content-Type: <input type="input" name="Content-Type" value="image/jpeg" /><br />
    <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
    Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
    <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
    <input type="hidden" name="Policy" value="POLICY" />
    <input type="hidden" name="Signature" value="SIGNATURE" />
    File: <input type="file" name="file" /> <br />
    <!-- The elements after this will be ignored -->
    <input type="submit" name="submit" value="Upload to Amazon S3" />
  </form>
  ...
</html>
```

請求範例

此要求假設上傳的映像為 117,108 個位元組 (不包含映像資料)。

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
 Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
```

```
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: 118698

--9431149156168
Content-Disposition: form-data; name="key"

user/eric/MyPicture.jpg
--9431149156168
Content-Disposition: form-data; name="acl"

public-read
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html
--9431149156168
Content-Disposition: form-data; name="Content-Type"

image/jpeg
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

Some,Tag,For,Picture
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--9431149156168
Content-Disposition: form-data; name="Policy"

eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN
--9431149156168
Content-Disposition: form-data; name="Signature"

0RavWzkygo6QX9caELEqKi9kDbU=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
```

```
Content-Type: image/jpeg

...file content...
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

回應範例

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
successful_upload.html?bucket=awsexamplebucket1&key=user/eric/
MyPicture.jpg&etag=&quot;39d459dfbc0faabbb5e179358dfb94c3&quot;
Server: AmazonS3
```

文字區域上傳

主題

下列範例顯示建構政策與表單以上傳文字區域的完整程序。上傳文字區域有助於提交使用者建立的內容，例如部落格張貼內容。

政策與表單建構

下列政策支援將文字區域上傳至 Amazon S3 的 awsexamplebucket1 儲存貯體。

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
```

```
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
new_post.html"},
    ["eq", "$Content-Type", "text/html"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

此政策需要進行下列操作：

- 上傳必須發生在 2007-12-01 12:00 GMT 以前。
- 內容必須上傳到 awsexamplebucket1 儲存貯體。
- 金鑰必須以 "user/eric/" 開頭。
- ACL 已設定為 public-read。
- success_action_redirect 已設定為 https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html。
- 物件是 HTML 文字。
- 此標 x-amz-meta-uuid 籤必須設定為 143651274。
- x-amz-meta-tag 可以包含任何值。

以下為此政策的 Base64 編碼版本。

```
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXR
pb25zIjogWwogICAgeyJidWNrZXQiOiAiam9obnNtaXRoIn0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXkiLCAidXNlci
LAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3Rpb25fcmVkaXJlY3QiOiAiaHR0cDovL2p
C5zMy5hbWF6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSIsICJ0ZXh0L2h0
CAgIHsieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiR4LWFtei1t
IsICIiXQogIF0KfQo=
```

使用您的登入資料建立簽章。例如，qA7FWXKq6VvU68lI9KdveT1cWgF= 是上述政策文件的簽章。

下列表單支援對使用此政策的 DOC-EXAMPLE-BUCKET 儲存貯體提出 POST 要求。

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
```

```
  <body>
  ...
  <form action="https://DOC-EXAMPLE-BUCKET.s3.us-west-1.amazonaws.com/" method="post"
 enctype="multipart/form-data">
    Key to upload: <input type="input" name="key" value="user/eric/" /><br />
    <input type="hidden" name="acl" value="public-read" />
    <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html" />
    <input type="hidden" name="Content-Type" value="text/html" />
    <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
    Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
    <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
    <input type="hidden" name="Policy" value="POLICY" />
    <input type="hidden" name="Signature" value="SIGNATURE" />
    Entry: <textarea name="file" cols="60" rows="10">

Your blog post goes here.

    </textarea><br />
    <!-- The elements after this will be ignored -->
    <input type="submit" name="submit" value="Upload to Amazon S3" />
  </form>
  ...
</html>
```

請求範例

此要求假設上傳的映像為 117,108 個位元組 (不包含映像資料)。

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
 Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=178521717625888
Content-Length: 118635

-178521717625888
```

```
Content-Disposition: form-data; name="key"

ser/eric/NewEntry.html
--178521717625888
Content-Disposition: form-data; name="acl"

public-read
--178521717625888
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html
--178521717625888
Content-Disposition: form-data; name="Content-Type"

text/html
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-tag"

Interesting Post
--178521717625888
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--178521717625888
Content-Disposition: form-data; name="Policy"
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN

--178521717625888
Content-Disposition: form-data; name="Signature"

qA7FWXKq6VvU68lI9KdveT1cWgF=
--178521717625888
Content-Disposition: form-data; name="file"

...content goes here...
--178521717625888
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
```

```
--178521717625888--
```

回應範例

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html?
bucket=awsexamplebucket1&key=user/eric/
NewEntry.html&etag=40c3271af26b7f1672e41b8a274d28d4
Server: AmazonS3
```

POST 與 Adobe Flash

本節說明如何搭配 Adobe Flash 使用 POST。

Adobe Flash Player 安全

根據預設，Adobe Flash Player 安全模型禁止 Adobe Flash Player 與提供 SWF 檔案之網域外部的伺服器建立網路連線。

若要覆寫預設值，您必須將可公開讀取的 crossdomain.xml 檔案上傳至接受 POST 上傳的儲存貯體。以下為 crossdomain.xml 檔案範例。

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="*" secure="false" />
</cross-domain-policy>
```

> **Note**
>
> 如需 Adobe Flash 安全模型的詳細資訊，請前往 Adobe 網站。
> 將 crossdomain.xml 檔案新增至您的儲存貯體，可讓任何 Adobe Flash Player 連接到儲存貯體中的 crossdomain.xml 檔案；但這不會授予對實際 Amazon S3 儲存貯體的存取權。

Adobe Flash 考量

Adobe Flash 中的 FileReference API 會將`Filename`表單欄位新增至 POST 要求。當您建立使用 FileReference API 動作上傳至 Amazon S3 的 Adobe Flash 應用程式時，請在您的政策中包含下列條件：

```
['starts-with', '$Filename', '']
```

Adobe Flash Player 的某些版本無法正確處理具有空白主體的 HTTP 回應。若要設定 POST 以傳回沒有空白主體的回應，請將 `success_action_status` 設定為 201。Amazon S3 將會傳回狀態碼為 201 的 XML 文件。如需 XML 文件內容的相關資訊，請參閱 POST 物件。如需表單欄位的資訊，請參閱「HTML 表單欄位」。

最佳實務設計模式：最佳化 Amazon S3 效能

您的應用程式從 Amazon S3 上傳和擷取儲存體時，可輕鬆達到請求中每秒數千筆交易的效能。Amazon S3 會自動調高請求率。例如，您的應用程式可以達成每個分割的 Amazon S3 字首每秒至少 3,500 個 PUT/COPY/POST/DELETE 和 5,500 個 GET/HEAD 請求。在儲存貯體中的字首數不受限制。您可以使用並行化提升讀取或寫入的效能。例如，如果您在 Amazon S3 儲存貯體裡建立 10 個字首，平行讀取，您可以縮放讀取效能至每秒 55,000 讀取要求。同樣，您可以透過寫入多個前綴來縮放寫入操作。在讀取和寫入操作的情況下，擴展會逐漸發生，而不是立即發生。Amazon S3 不斷地擴展到新的更高請求率時，您可能會看到一些 503 (減慢) 錯誤。擴展完成時，這些錯誤會消失。如需有關建立和使用字首的詳細資訊，請參閱 使用字首整理物件。

例如，Amazon S3 上的部分資料湖應用程式在查詢超過數 PB 資料時，可能掃描數百萬或數十億個物件。這些資料湖應用程式實現單一執行個體傳輸率，讓 Amazon EC2 執行個體充分利用網路界面，在單一執行個體上最高可達到 100 Gb/s。然後，這些應用程式會彙總多個執行個體之間的傳輸量，以取得每秒多個 TB。

其他應用程式為對延遲敏感的應用程式，例如社交媒體簡訊應用程式。這些應用程式可達到大約 100—200 毫秒的一致小型物件 first-byte-out 延遲 (以及較大物件的延遲)。

其他 AWS 服務也有助於加快不同應用程式架構的效能。例如，如果您希望透過單一 HTTP 連線或低於 10 毫秒的延遲提高傳輸速率，請使用 Amazon CloudFront 或 Amazon 快 ElastiCache取 Amazon S3。

此外，如果您希望在相距遙遠的用戶端與 S3 儲存貯體之間快速傳輸資料，請使用 使用 Amazon S3 Transfer Acceleration 設定快速安全的檔案傳輸。傳輸加速使用中的全球分散式邊緣位置 CloudFront 來加速在地理距離內的資料傳輸速度。如果您的 Amazon S3 工作負載使用伺服器端加密 AWS KMS，請參閱 AWS Key Management Service 開發人員指南中的AWS KMS 限制，以取得您使用案例支援的請求費率的相關資訊。

下列主題針對使用 Amazon S3 的應用程式，說明最佳化效能的最佳實務指導方針和設計模式。請參閱 Amazon S3 的效能指導方針和 Amazon S3 的效能設計模式，以取得 Amazon S3 效能最佳化的最新資訊。

> **Note**
>
> 如需將 Amazon S3 Express One Zone 儲存類別與目錄儲存貯體搭配使用的詳細資訊，請參閱 什麼是 S3 Express One Zone？ 和 目錄值區。

主題

Amazon S3 的效能指導方針

建置從 Amazon S3 上傳和擷取物件的應用程式時，請依照我們的最佳實務指導方針來最佳化效能。我們還提供更詳細的效能設計模式。

為了讓您在 Amazon S3 上的應用程式獲得最佳效能，建議採用下列指導方針。

主題

測量效能

最佳化效能時，請查看網路傳輸量、CPU 和 DRAM 要求。根據這些不同資源的混合需求，可能值得評估不同的 Amazon EC2 執行個體類型。如需執行個體類型的詳細資訊，請參閱 Amazon EC2 使用者指南中的執行個體類型。

測量效能時，使用 HTTP 分析工具來查看 DNS 查閱時間、延遲和資料傳輸速度也很有幫助。

若要了解效能需求並最佳化您應用程式的效能，您也可以監控收到的 503 錯誤回應。監控某些效能指標可能會產生額外費用。如需詳細資訊，請參閱 Simple Storage Service (Amazon S3) 定價。

監控 503 (減慢) 狀態錯誤回應的數目

若要監控您取得的 503 狀態錯誤回應數目，您可以使用下列其中一個選項：

- 使用 Amazon S3 的亞馬遜 CloudWatch 請求指標。要 CloudWatch 求量度包含 5xx 狀態回應的量度。如需 CloudWatch 要請求測量結果的詳細資訊，請參閱使用 Amazon 監控指標 CloudWatch。
- 使用 Amazon S3 Storage Lens 的進階指標區段中提供的 503 (服務無法使用) 錯誤計數。如需詳細資訊，請參閱 使用 S3 Storage Lens 指標來改善效能。
- 使用 Amazon S3 伺服器存取記錄 使用伺服器存取記錄，您可以篩選並檢閱接收 503 (內部錯誤) 回應的所有請求。您也可以使用 Amazon Athena 來剖析日誌。如需伺服器存取記錄日誌的詳細資訊，請參閱「使用伺服器存取記錄記錄要求」。

透過監控 HTTP 503 狀態錯誤碼的數量，您通常可以獲得寶貴的洞見，了解哪些字首、金鑰或儲存貯體得到最多的限流請求。

水平擴展儲存體連線

跨許多連線傳播請求是常用的設計模式，藉此水平擴展效能。建置高效能應用程式時，請將 Amazon S3 視為一個非常大的分散式系統，而不是像傳統儲存伺服器那樣的單一網路端點。您可以向 Amazon S3 發出多個並行請求，以達到最佳效能。將這些請求分散到不同的連線，以從 Amazon S3 獲得最大可存取的頻寬。Amazon S3 對儲存貯體的連線數沒有任何限制。

使用位元組範圍擷取

您可以在 GET 物件請求中使用 Range HTTP 標頭，以從物件中擷取位元組範圍，僅傳輸指定的部分。您可以對 Amazon S3 使用並行連線，從同一物件內擷取不同的位元組範圍。與單一整個物件請求相比，這可協助您實現更高的彙總傳輸量。擷取大型物件的更小範圍也可讓您的應用程式在請求中斷時改善重試時間。如需詳細資訊，請參閱「下載物件」。

位元組範圍請求的一般大小為 8 MB 或 16 MB。如果物件是使用多部件上傳的 PUT，則良好實務為以相同部件大小 (或至少與部件界限一致) GET 它們，以取得最佳效能。GET 請求可以直接處理個別組件；例如，`GET ?partNumber=N`.

對延遲敏感的應用程式重試請求

積極的逾時和重試可協助推動一致的延遲。由於 Amazon S3 的範圍很大，如果第一個請求很慢，則重試的請求可能會採取不同路徑且很快成功。 AWS SDK 具有可設定的逾時值和重試值，您可以根據特定應用程式的容差進行調整。

將 Amazon S3（存儲）和 Amazon EC2（運算）結合在一起 AWS 區域

雖然 S3 儲存貯體名稱是全域唯一，但每個儲存貯體都存放在您建立儲存貯體時選取的區域中。為了最佳化效能，建議您盡可能從 Amazon EC2 執行個體存取儲存貯體。 AWS 區域 這可協助減少網路延遲和資料傳輸成本。

如需資料傳輸成本的詳細資訊，請參閱 Amazon S3 定價。

使用 Amazon S3 Transfer Acceleration 將距離產生的延遲最小化

使用 Amazon S3 Transfer Acceleration 設定快速安全的檔案傳輸 可讓用戶端與 S3 儲存貯體之間長地理距離的檔案傳輸變得迅速、簡單又安全。傳輸加速充分利用 Amazon 中全球分佈的節點 CloudFront。資料到達節點時會經由最佳化的網路路徑而路由至 Amazon S3。Transfer Acceleration 非常適合在各大洲定期傳輸數 GB 到數 TB 的資料。也有助於從世界各地上傳至集中型儲存貯體的用戶端。

您可以使用 Amazon S3 Transfer Acceleration 速度比較工具，比較各 Amazon S3 區域的加速和非加速上傳速度。速度比較工具在使用和不使用 Amazon S3 Transfer Acceleration 的情況下，透過分段上傳，將檔案從您的瀏覽器傳送至各個 Amazon S3 區域。

使用 AWS 開發套件的最新版本

開 AWS 發套件為最佳化 Amazon S3 效能的許多建議準則提供內建支援。這些開發套件提供更簡單的 API 以方便從應用程式內善用 Amazon S3，並定期更新來遵循最新的最佳實務。例如，軟體開發套件包括在發生 HTTP 503 錯誤時自動重試請求的邏輯，並投資於程式碼以回應並適應慢速連線。

這些開發套件也提供 Transfer Manager，在適當情況下會使用位元組範圍請求，以自動水平擴展連線，達到每秒數千個請求。請務必使用最新版本的 AWS SDK 來取得最新的效能最佳化功能。

使用 HTTP REST API 請求時，您也可以最佳化效能。使用 REST API 時，您應該遵循屬於軟體開發套件的相同最佳實務。允許逾時和重試慢速請求，並有多個連線允許平行擷取物件資料。如需有關使用 REST API 的詳細資訊，請參閱 Amazon Simple Storage Service API 參考。

Amazon S3 的效能設計模式

設計應用程式以從 Amazon S3 上傳和擷取物件時，請採用我們的最佳實務設計模式，讓您的應用程式達到最佳效能。我們也提供效能指導方針，供您在規劃應用程式架構時考量。

若要最佳化效能，您可以使用下列設計模式。

主題

對經常存取的內容使用快取

許多將資料存放在 Amazon S3 的應用程式會提供資料「工作集」，供使用者重複請求。如果工作負載傳送一組通用物件的重複 GET 請求，您可以使用快取 (例如 Amazon CloudFront ElastiCache、Amazon) 或AWS Elemental MediaStore優化效能。成功採用快取可以產生低延遲和高資料傳輸率。使用快取的應用程式傳送至 Amazon S3 的直接請求也較少，有助於減少請求成本。

Amazon CloudFront 是一種快速內容交付網路 (CDN)，可透明地從 Amazon S3 快取資料，並在一組分散各地的存在點 (PoPs) 中快取資料。當物件可從多個區域或透過網際網路存取時，可 CloudFront 讓資料快取靠近存取物件的使用者。這樣能夠高效能傳遞熱門的 Amazon S3 內容。如需相關資訊 CloudFront，請參閱 Amazon CloudFront 開發人員指南。

Amazon ElastiCache 是一個受管的，內存緩存。您可以使用佈建 Amazon EC2 執行個體 ElastiCache，以快取記憶體中的物件。此快取會導致 GET 延遲數量級減少，以及下載傳輸量顯著增加。若要使用 ElastiCache，您可以修改應用程式邏輯，以便在快取中填入熱物件，並在從 Amazon S3 請求這些物件之前檢查快取是否有熱物件。如需用 ElastiCache 來改善 Amazon S3 GET 效能的範例，請參閱部落格文章使用 Amazon ElastiCache 為 Redis 渦輪增壓 Amazon S3。

AWS Elemental MediaStore 是專為 Amazon S3 視訊工作流程和媒體交付而建置的快取和內容分發系統。 MediaStore 提供專門用於視訊的 end-to-end 儲存 API，建議用於對效能敏感的視訊工作負載使用。若要取得有關資訊 MediaStore，請參閱《AWS Elemental MediaStore 使用指南》。

適用於對延遲敏感之應用程式的逾時和重試

在某些情況下，應用程式會收到 Amazon S3 的回應，這表示有必要重試。Amazon S3 會將儲存貯體和物件名稱對應至相關聯的物件資料。如果應用程式產生高請求率 (通常對少數物件維持每秒超過 5,000 個請求的速率)，則它可能會收到 HTTP 503 slowdown 回應。如果發生這些錯誤，每個 AWS 開發套件會使用指數退避來實作自動重試邏輯。如果您未使用 AWS 開發套件，則應該在收到 HTTP 503 錯誤時實作重試邏輯。如需有關退期技術的資訊，請參閱中的錯誤重試和指數輪詢。 AWSAmazon Web Services 一般參考

Amazon S3 會隨著持續的新請求率而自動擴展，動態地最佳化效能。當 Amazon S3 在內部針對新請求率最佳化時，您將會暫時收到 HTTP 503 請求回應，直到最佳化完成為止。在 Amazon S3 於內部針對新請求率來最佳化效能之後，通常就能處理所有請求而不必重試。

對於需要低延遲的應用程式，Amazon S3 建議追蹤並積極重試較慢的操作。當您重試請求時，我們建議對 Amazon S3 使用新連線，並執行全新的 DNS 查閱。

當您進行大量易變大小的請求 (例如，超過128 MB) 時，我們建議追蹤正要實現的傳輸量並重試最慢的 5% 請求。當您提出更小的要求 (例如，少於 512 KB)，其中中位數延遲通常在幾十毫秒範圍內時，良好實務為 2 秒後重試 GET 或 PUT 操作。如果需要額外重試，最佳實務為退避。例如，我們建議 2 秒後發出一次重試，再過 4 秒後發出第二次重試。

如果您的應用程式對 Amazon S3 提出固定大小的請求，則這些請求的回應時間都應該會更一致。在此情況下，簡單策略為識別最慢的 1% 請求，然後重試它們。即使單次重試也常常有效地減少延遲。

如果您使用 AWS Key Management Service (AWS KMS) 進行伺服器端加密，請參閱AWS Key Management Service 開發人員指南中的限制，以取得您使用案例支援的請求率的相關資訊。

高傳輸量的水平擴展和請求並行化

Amazon S3 是非常大的分散式系統。為了協助您善用其規模，建議您水平擴展對 Amazon S3 服務端點的並行請求。除了在 Amazon S3 內分發要求，這種擴展方法還有助於將負載分散至網路上的多個路徑。

對於高傳輸量傳輸，Amazon S3 建議使用對 GET 或 PUT 資料並行使用多個連線的應用程式。例如，Amazon S3 傳輸管理員在 AWS Java 開發套件中支援此功能，而大多數其他開 AWS 發套件則提供類似的建構。對於部分應用程式，您可以實現並行連線，方法為在不同的應用程式執行緒中或在不同的應用程式執行個體中並行啟動多個請求。採取的最佳方法取決於您的應用程式，以及您正在存取的物件結構。

您可以使用 AWS SDK 直接發出 GET 和 PUT 請求，而不是採用 AWS SDK 中的傳輸管理。此方法可讓您更直接調整工作負載，同時仍能受益於軟體開發套件支援重試，以及處理任何可能發生的 HTTP 503 回應。當您在區域內將大型物件從 Amazon S3 下載至 Amazon EC2 時，我們通常建議您對物件的位元組範圍提出並行請求，精細程度為 8–16 MB。針對所需網路輸送量的每個 85–90 MB/s，提出一個並行請求。若要使 10 Gb/s 網路界面卡 (NIC) 飽和，您可以透過個別連線使用大約 15 個並行請求。您可以透過更多連線來擴增並行請求，使更快的 NIC 飽和，例如 25 Gb/s 或 100 Gb/s NIC。

當您調整要同時發出的請求數時，測量效能很重要。我們建議從一次提出單一請求開始。測量要實現的網路頻寬，以及您的應用程式在處理資料時其他資源的使用情形。然後，您可以識別瓶頸資源 (亦即，具有最高用量的資源)，因此識別可能有用的請求數。例如，如果一次處理一個請求導致使用 25%

CPU，則其建議最多只能容納四個並行請求。測量是必要的操作，且隨著請求率的增加，確認資源用量是值得的。

如果您的應用程式使用 REST API 直接向 Amazon S3 發出請求，我們建議您使用 HTTP 連線集區，並針對一系列請求重複使用每個連線。避免每個請求的連線設定可移除在每個請求上執行 TCP 緩慢啟動和 Secure Sockets Layer (SSL) 交握的需求。如需有關使用 REST API 的詳細資訊，請參閱 Amazon Simple Storage Service API 參考。

最後，值得注意 DNS，並仔細檢查請求是否分散在廣泛的 Amazon S3 IP 地址集區。Amazon S3 的 DNS 查詢會在大量 IP 端點之中循環進行。但是快取解析程式或重複使用單一 IP 地址的應用程式碼不會受益於地址多樣性和隨之而來的負載平衡。網路公用程式工具 (例如 `netstat` 命令列工具) 可以顯示用來與 Amazon S3 通訊的 IP 地址，我們提供指導方針供 DNS 組態使用。如需這些指導方針的詳細資訊，請參閱「提出要求」。

使用 Amazon S3 Transfer Acceleration 加速異地資料傳輸

使用 Amazon S3 Transfer Acceleration 設定快速安全的檔案傳輸在分散全球的客戶端與使用 Amazon S3 的區域應用程式之間， 可有效盡量降低或消除地理距離所引起的延遲。傳輸加速會使用中的全域分散式邊緣位置 CloudFront 進行資料傳輸。邊 AWS 緣網路在 50 多個位置有存在點。如今，它可用於透過分發內容， CloudFront 並為對 Amazon Route 53 進行的 DNS 查詢提供快速回應。

邊緣網路也有助於加速進出 Amazon S3 的資料傳輸。它非常適合於各大洲之間傳輸資料的應用程式、具有快速網際網路連線的應用程式、使用大型物件的應用程式，或具有許多內容要上傳的應用程式。當資料到達節點時，資料會經由最佳化的網路路徑而路由至 Amazon S3。一般而言，離 Amazon S3 區域越遠，使用 Transfer Acceleration 改善速度越明顯。

您可以在新的或現有的儲存貯體上設定 Transfer Acceleration。您可以使用單獨的 Amazon S3 Transfer Acceleration 端點來使用 AWS 節點。測試 Transfer Acceleration 是否可提升用戶端請求效能的最好方式，就是使用 Amazon S3 Transfer Acceleration 速度比較工具。網路組態和狀況會隨著時間和位置而有所不同。因此，只有在 Amazon S3 Transfer Acceleration 有可能改善上傳效能時，才會向您收取傳輸費用。如需將傳輸加速與不同 AWS SDK 搭配使用的相關資訊，請參閱啟用和使用 S3 Transfer Acceleration。

什麼是 Amazon S3 on Outposts？

AWS Outposts 是一項完全受控的服務，可為幾乎任何資料中心、主機代管空間或內部部署設施提供相同的 AWS 基礎架構、 AWS 服務、API 和工具，以獲得真正一致的混合體驗。 AWS Outposts 非常適合需要低延遲存取內部部署系統、本機資料處理、資料存放區，以及移轉具有本機系統相互依存性之應用程式的工作負載。如需詳細資訊，請參閱《 AWS Outposts使用者指南》中的什麼是AWS Outposts ？。

使用 Amazon S3 on Outposts，您可以在 Outposts 上建立 S3 儲存貯體並輕鬆存放和擷取內部部署物件。S3 on Outposts 提供一個全新的儲存類別，即 OUTPOSTS，使用 Amazon S3 API，目的是在您的 Outposts 上的多個裝置和伺服器上以持久、備援的方式存放資料。您可以使用存取點和透過虛擬私有雲端 (VPC) 的端點連線，與您的 Outposts 儲存貯體進行通訊。

就像在 Amazon S3 一樣，您在 Outpost 儲存貯體上可以使用同樣的 API 和功能，包括存取政策、加密和標記。您可以透過 AWS Management Console、 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 REST API 在 Outposts 上使用 S3。

- S3 on Outposts 如何工作
- S3 on Outposts 功能
- 相關服務
- 存取 S3 on Outposts
- 支付 S3 on outposts
- 後續步驟

S3 on Outposts 如何工作

S3 on Outposts 是將資料當做物件存放在您的 Outpost 儲存貯體中的物件儲存服務。物件是一個資料檔案和任何描述該檔案的中繼資料。儲存貯體是物件的容器。

若要將資料存放在 S3 on Outposts 中，您首先要建立儲存貯體。建立儲存貯體時，您需要指定儲存貯體名稱和將保存儲存貯體的 Outpost。若要存取 S3 on SOutposts 儲存貯體並執行物件操作，接下來需要建立並設定一個存取點。您還必須建立端點，以將請求路由到存取點。

存取點可簡化任何 AWS 服務 或將資料存放在 S3 中的客戶應用程式的資料存取。存取點為連接到儲存貯體的指定網路端點，這些端點可用於執行物件操作，例如 `GetObject` 和 `PutObject`。每個存取點都有不同的許可和網路控制。

您可以使用、、 AWS 開發套件或 REST API 在 Outposts 儲存貯體、存取點和端點上建立和管理 S3。AWS Management Console AWS CLI若要在 Outposts 儲存貯體上上傳和管理 S3 中的物件，您可以使用 AWS CLI、 AWS 開發套件或 REST API。

區域

在 AWS Outposts 佈建期間，您或 AWS 建立服務連結連線，將 Outpost 連線回您選擇的 AWS 區域 或 Outposts 本地區域，以進行儲存貯體作業和遙測。Outpost 依賴於與父節點之間的連接 AWS 區域。Outposts 機架不適用於斷開連接的操作或連接受限制的環境。如需詳細資訊，請參閱《AWS Outposts 使用者指南》中的Outpost 連線 AWS 區域。

儲存貯體

儲存貯體是 S3 on Outposts 中用於存放物件的容器。您可以在儲存貯體中存放任意數目的物件，並且每個 Outpost 中的每個帳戶最多可有 100 個儲存貯體。

建立儲存貯體時，請輸入儲存貯體名稱並選擇儲存貯體將駐留的 Outpost。建立儲存貯體後，便無法變更儲存貯體名稱或移動儲存貯體到不同 Outpost。儲存貯體名稱必須遵循Amazon S3 儲存貯體命名規則。在 Outposts 的 S3 中，存儲桶名稱對於前哨和. AWS 帳戶 S3 on Outposts 儲存貯體需要 `outpost-id`、`account-id` 和儲存貯體名稱來識別。

下列範例顯示了 S3 on Outposts 儲存貯體 的 Amazon 資源名稱 (ARN) 格式。ARN 由您的 Outpost 所在區域、您的 Outpost 帳戶、Outpost ID 和儲存貯體名稱組成。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/bucket/bucket-name
```

每個物件都包含在儲存貯體中。您必須使用存取點來存取 Outposts 儲存貯體中的任何物件。針對物件操作指定儲存貯體時，您可以使用存取點 ARN 或存取點別名。如需存取點別名的詳細資訊，請參閱 針對您的 S3 on Outposts 儲存貯體存取點使用儲存貯體樣式別名。

下列範例顯示了 S3 on Outposts 的存取點 ARN 格式，其中包含了 `outpost-id`、`account-id` 和存取點名稱：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

如需儲存貯體的詳細資訊，請參閱「使用 S3 on Outposts 儲存貯體」。

物件

物件是存放在 S3 on Outposts 中的基本實體。物件是由物件資料與中繼資料構成。中繼資料是一組成對的名稱與數值，會說明該物件。其中包含一些預設中繼資料 (如上次修改日期) 以及標準 HTTP 中繼資料 (如 `Content-Type`)。您也可以在存放物件時指定自訂中繼資料。在儲存貯體中，每個物件都是由金鑰 (名稱) 與版本 ID 來唯一識別。

對於 Amazon S3 on Outposts，物件資料始終存放在 Outpost 上。 AWS 安裝 Outpost 機架時，您的資料會保留在 Outpost 的本機，以符合資料駐留需求。您的物件永遠不會離開您的 Outpost，也不會在 AWS 區域中。由於在區域內託管，因 AWS Management Console 此您無法使用控制台上傳或管理 Outpost 中的物件。但是，您可以使用 REST API、 AWS Command Line Interface (AWS CLI) 和 AWS SDK，透過存取點上傳和管理物件。

鍵

物件金鑰 (或金鑰名稱) 是儲存貯體內的物件的唯一識別符。儲存貯體中的每個物件只能有一個索引鍵。儲存貯體和物件金鑰的組合唯一識別每個物件。

下列範例顯示 Outposts 物件上 S3 的 ARN 格式，其中包括前哨所在區域的 AWS 區域 程式碼、 AWS 帳戶 ID、前哨識別碼、儲存貯體名稱和物件金鑰：

```
arn:aws:s3-outposts:us-west-2:123456789012:outpost/ op-01ac5d28a6a232904/
bucket/example-s3-bucket1/object/myobject
```

如需物件金鑰的詳細資訊，請參閱「使用 S3 on Outposts 物件」。

S3 版本控制

您可以在 Outposts 上使用 S3 版本控制，以在相同的儲存貯體中保留物件的多個變體。使用 S3 版本控制功能，您即可保留、擷取和還原在儲存貯體中所存放每個物件的各個版本。S3 版本控制可協助您從意外的使用者動作和應用程式失敗中復原。

如需詳細資訊，請參閱 針對您的 S3 on Outposts 儲存貯體管理 S3 版本控制。

版本 ID

當您在儲存貯體中啟用 S3 版本控制時，S3 on Outposts 會針對每個新增至儲存貯體的物件產生唯一的版本 ID。啟用版本控制時已存在於儲存貯體中的物件的版本 ID 為 `null`。如果您使用其他操作（例如）修改這些（或任何其他）對象 PutObject，則新對象將獲得唯一的版本 ID。

如需詳細資訊，請參閱 針對您的 S3 on Outposts 儲存貯體管理 S3 版本控制。

儲存方案和加密

S3 on Outposts 提供新的儲存體方案 S3 Outposts (0UTP0STS)。S3 Outposts 儲存方案適用於只存放在 AWS Outposts儲存貯體中的物件。如果您嘗試與 S3 on Outposts 一起使用其他 S3 儲存方案，S3 on Outposts 會返回 `InvalidStorageClass` 錯誤。

根據預設，物件存放在 S3 Outposts (0UTP0STS) 儲存體方案中的物件一律使用伺服器端加密與 Amazon S3 受管加密金鑰 (SSE-S3) 進行加密。如需詳細資訊，請參閱 S3 on Outposts 中的資料加密。

儲存貯體政策

值區政策是以資源為基礎的 AWS Identity and Access Management (IAM) 政策，可用來授與值區及其中物件的存取權限。只有儲存貯體擁有者可建立政策與儲存貯體的關聯。連接到儲存貯體的許可會套用至儲存貯體擁有者帳戶擁有的所有儲存貯體物件。儲存貯體政策的大小限制為 20 KB。

儲存貯體政策使用 AWS標準的以 JSON 為基礎的 IAM 政策語言。您可以使用儲存貯體政策來新增或拒絕儲存貯體中物件的許可。儲存貯體政策允許或拒絕請求以政策中的元素為基礎。這些元素可以包括請求的申請者、S3 on Outposts 動作、資源以及其他方面或條件 (例如，用來傳送要求的 IP 地址)。例如，您可以建立儲存貯體政策，授予跨帳戶許可，以將物件上傳至 S3 on Outposts 儲存貯體，同時確保儲存貯體擁有者可完全控制上傳物件。如需詳細資訊，請參閱 Amazon S3 儲存貯體政策範例。

在儲存貯體政策中，您可以在 ARN 中的 (*) 上使用萬用字元和其他值上使用許可授予物件子集。例如，您可以控制對以常用字首或以給定的擴展名結束，例如 `.html`。

S3 on Outposts 存取點

S3 on Outposts 存取點是含有專用存取政策的命名網路端點，其中說明了如何使用該端點存取資料。存取點針對 S3 on Outposts 中的共用資料集，簡化了對大規模資料存取的管理。存取點為連接到儲存貯體，您可以使用這些端點來執行 S3 物件操作，例如 `GetObject` 和 `PutObject`。

針對物件操作指定儲存貯體時，您可以使用存取點 ARN 或存取點別名。如需存取點別名的詳細資訊，請參閱 針對您的 S3 on Outposts 儲存貯體存取點使用儲存貯體樣式別名。

存取點有 S3 on Outposts 對於透過該存取點進行的任何請求所套用的不同許可和網路控制。每個存取點都會強制執行自訂的存取點政策，該政策可結合附加至基礎儲存貯體的儲存貯體政策運作。

如需詳細資訊，請參閱 存取 S3 on Outposts 儲存貯體和物件。

S3 on Outposts 功能

存取管理

S3 on Outposts 提供稽核和管理對儲存貯體和物件的存取的功能。根據預設，S3 on Outposts 儲存貯體與物件皆為私有。您只能存取您建立的 S3 on Outposts 資源。

若要授予可支援特定使用案例的精密資源使用權限，或稽核 S3 on Outposts 資源的許可，您可以使用下列功能。

- S3 區塊公有存取 – 儲存貯體與物件的區塊公有存取。對於 Outposts 上的儲存貯體，預設情況下始終啟用區塊公有存取。
- AWS Identity and Access Management (IAM) — IAM 是一種 Web 服務，可協助您安全地控制對 AWS 資源的存取，包括 Outposts 資源上的 S3。使用 IAM，您可以集中管理控制使用者可以存取哪些 AWS 資源的許可。您可以使用 IAM 來控制能通過身分驗證 (登入) 和授權使用資源的 (具有許可) 的人員。
- S3 on Outposts 存取點 - 針對 S3 on Outposts 中的共用資料集對資料存取的管理。存取點以專屬存取政策命名網路端點。存取點與儲存貯體相關聯，可用於執行物件操作，例如 `GetObject` 和 `PutObject`。
- 儲存貯體政策 – 使用以 IAM 為基礎的政策語言，為 S3 儲存貯體及其中的物件設定以資源為基礎的許可。
- AWS Resource Access Manager (AWS RAM) — 在 AWS Organizations您的組織或組織單位 (OU) 內 AWS 帳戶，安全地在 Outposts 容量上共用您的 S3。

儲存記錄和監控

S3 on Outposts 提供記錄和監控工具，您可以使用這些工具來監控和控制 S3 on Outposts 資源的使用方式。如需詳細資訊，請參閱監控工具。

- Outposts 上 S3 的 Amazon CloudWatch 指標 — 追蹤資源的運作狀態並了解您的容量可用性。
- Outposts S3 的亞馬遜 CloudWatch 活動事件 — 為 Outposts 上的任何 S3 API 事件建立規則，以便透過所有支援CloudWatch 的事件目標接收通知，包括亞馬遜簡單佇列服務 (Amazon SQS)、亞馬遜簡單通知服務 (Amazon SNS) 和。 AWS Lambda
- AWS CloudTrail Outposts 上 S3 的日誌 — 記錄前 Outposts 上使用者、角色或 S3 AWS 服務 中採取的動作。 CloudTrail 日誌可為您提供 S3 儲存貯體層級和物件層級作業的詳細 API 追蹤。

高度的一致性

Outposts 上的 S3 為 Outposts 存儲桶上 S3 中對象的 PUT 和刪除請求提供了強大的 read-after-write 一致性。 AWS 區域這一行為適用於新物件的寫入，和覆寫現有物件的 PUT 請求與 DELETE 請求。此外，S3 on Outposts 物件標籤和物件中繼資料 (例如 HEAD 物件) 高度一致。如需詳細資訊，請參閱 Amazon S3 資料一致性模式。

相關服務

將資料載入至 S3 on Outposts 之後，您可以搭配其他 AWS 服務使用資料。以下為您可能最常使用的服務：

- Amazon Elastic Compute Cloud (Amazon EC2) – 在 AWS 雲端中提供安全且可擴展的運算容量。使用 Amazon EC2 可減少前期所需的硬體投資，讓您更快速開發並部署應用程式。您可使用 Amazon EC2 按需要啟動任意數量的虛擬伺服器，設定安全性和聯網功能以及管理儲存。
- Outpost 上的 Amazon Elastic Block Store (Amazon EBS) - 使用 Amazon EBS 本機快照 將 Outpost 上的磁碟區快照以本機方式儲存在 Outpost 上的 S3 on Outposts 本身。
- Amazon Relational Database Service (Amazon RDS) - 使用 Amazon RDS 本地備份將您的 Amazon RDS 備份本地存儲在您的 Outpost 上。
- AWS DataSync— 自動在 Outposts 之間傳輸數據 AWS 區域，並選擇要傳輸的內容，何時傳輸以及使用多少網絡帶寬。Outposts 上的 S3 與 AWS DataSync. 對於需要高輸送量本機處理的內部部署應用程式，S3 on Outposts 提供內部部署物件儲存體，以盡量減少因網路變化而產生的資料傳輸和緩衝區，同時還能讓您輕鬆地在 Outpost 與 AWS 區域之間傳輸資料。

存取 S3 on Outposts

您可以透過以下任何方式來使用 S3 on Outposts：

AWS Management Console

主控台是可用來管理 S3 on Outposts 和 AWS 資源的 Web 型使用者介面。如果您已經註冊了 AWS 帳戶，則可以通過登錄 AWS Management Console 並從 AWS Management Console 首頁選擇 S3 來訪問 Outposts 上的 S3。然後，從左側導覽窗格中選擇 Outposts buckets (Outposts 儲存貯體)。

AWS Command Line Interface

您可以使用命 AWS 令列工具在系統的命令列上發出命令或建置指令碼，以執行 AWS (包括 S3) 工作。

AWS Command Line Interface (AWS CLI) 為廣泛的集合提供指令 AWS 服務。視窗、macOS 和 Linux 上支援。 AWS CLI 若要開始使用，請參閱《 使用者指南AWS Command Line Interface》。如需可與 S3 on Outposts 搭配使用之命令的詳細資訊，請參閱 《AWS CLI 命令參考》中的 s3api、s3control 和 s3outposts。

AWS 開發套件

AWS 提供包含各種程式設計語言和平台 (Java、Python、Ruby、.NET、iOS、安卓等) 的程式庫和範例程式碼的 SDK (軟體開發套件)。開 AWS 發套件提供了一種便捷的方式，可在 Outposts 和. AWS由於 S3 on Outposts 使用與 Amazon S3 相同的軟件開發工具包，所以 S3 on Outposts 使用相同的 S3 API、自動化和工具提供一致的體驗。

S3 on Outposts 是 REST 服務。您可以使用 AWS 開發套件程式庫 (其會包裝基礎 REST API)，傳送請求至 S3 on Outposts，從而簡化程式設計任務。例如，開發套件會負責的工作諸如計算簽章、以密碼演算法簽署請求、管理錯誤以及自動重試請求。如需 AWS SDK 的相關資訊，包括如何下載和安裝它們，請參閱建置的 AWS工具。

支付 S3 on outposts

您可以購買各種 AWS Outposts 機架組態，其中包括 Amazon EC2 執行個體類型、Amazon EBS 一般用途固態硬碟 (SSD) 磁碟區 (gp2) 和 Outposts 上的 S3。定價包括運輸、安裝、基礎設施服務維護 、以及軟體修補程式和升級。

如需詳細資訊，請參閱 AWS Outposts rack 定價。

後續步驟

如需有關使用 S3 on Outposts 的詳細資訊，請參閱下列主題：

- 設定 Outpost
- Amazon S3 on Outposts 與 Amazon S3 有何不同？
- Amazon S3 on Outposts 入門
- 適用於 S3 on Outposts 的網路

- 使用 S3 on Outposts 儲存貯體
- 使用 S3 on Outposts 物件
- S3 on Outposts 中的安全性
- 管理 S3 on Outposts 儲存貯體
- 使用 Amazon S3 on Outposts 進行開發

設定 Outpost

若要開始使用 Amazon S3 on Outposts，您需要在設施中部署有 Amazon S3 容量的 Outpost。如需訂購 Outpost 和 S3 容量選項的詳細資訊，請參閱 AWS Outposts。若要檢查您的 Outposts 是否有 S3 容量，您可以使用 ListOutpostsWithS3 API 呼叫。有關規範以及要瞭解 S3 on Outposts 與 Amazon S3 有何不同，請參閱 Amazon S3 on Outposts 與 Amazon S3 有何不同？。

如需詳細資訊，請參閱下列主題。

主題

- 訂購新 Outpost

訂購新 Outpost

如果您需要訂購含 S3 容量的新 Outpost，請參閱 AWS Outposts 定價，以瞭解 Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Elastic Block Store (Amazon EBS) 和 Amazon S3 的容量選項。

選取您的組態之後，請依照 AWS Outposts 使用者指南中的建立 Outpost 和訂購 Outpost 容量的步驟操作。

Amazon S3 on Outposts 與 Amazon S3 有何不同？

Amazon S3 on Outposts 會將物件儲存交付到您的內部部署 AWS Outposts 環境。使用 S3 on Outposts 可透過將資料保留在靠近內部部署應用程式的位置，協助您滿足本機處理、資料駐留、和嚴苛的效能需求。因為它會使用 Amazon S3 API 和功能，所以 S3 on Outposts 可讓您輕鬆存放、保護、標記、報告和控制對 Outposts 上的資料存取，並將 AWS 基礎設施延伸到您的內部部署設施，以獲得一致的混合體驗。

如需 S3 on Outposts 特點的詳細資訊，請參閱下列主題。

主題

S3 on Outposts 規格

- Outposts 儲存貯體大小上限為 50 TB。
- 每個 AWS 帳戶 的 Outposts 儲存貯體數目上限為 100。
- 只能使用存取點和端點來存取 Outposts 儲存貯體。
- 每個 Outpost 儲存貯體的存取點數目上限為 10。
- 存取點政策的大小限制為 20 KB。
- Outpost 擁有者可以使用 AWS Resource Access Manager 在 AWS Organizations 中管理您組織內的存取權。所有需要存取 Outpost 的帳戶必需在與 AWS Organizations 中的擁有者帳戶相同的組織內。
- S3 on Outpost 儲存貯體擁有者帳戶一律是儲存貯體中所有物件的擁有者。
- S3 on Outpost 儲存貯體擁有者帳戶只能對儲存貯體執行作業。
- 物件大小限制與 Amazon S3 一致。
- 所有儲存在 S3 on Outposts 上的物件都儲存在 OUTPOSTS 儲存類別中。
- 存放在 OUTPOSTS 儲存體類別中的所有物件預設為使用伺服器端加密與 Amazon S3 受管加密金鑰 (SSE-S3) 進行儲存。您也可以明確選擇使用伺服器端加密與客戶提供的加密金鑰 (SSE-C) 來存放物件。
- 如果無足夠的空間可以在您的 Outpost 上儲存物件，API 將傳回容量不足例外狀況 (ICE)。

S3 on Outposts 支援的 API 操作

如需 S3 on Outposts 支援的 API 操作清單，請參閱 Amazon S3 on Outposts API 操作。

S3 on Outposts 不支援的 Amazon S3 功能

Amazon S3 on Outposts 目前不支援下列 Amazon S3 功能。任何嘗試使用它們操作都會遭拒絕。

- 存取控制清單 (ACL)

- 跨來源資源分享 (CORS)
- S3 批次操作
- S3 庫存報告
- 變更預設儲存貯體加密
- 公有儲存貯體
- 多重要素驗證 (MFA) 刪除
- S3 生命週期轉移 (物件刪除和停用不完整的分段上傳除外)
- S3 物件鎖定法務保存
- 物件鎖定保留
- 以 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 為伺服器端加密。
- S3 複寫時間控制 (S3 RTC)
- Amazon CloudWatch 請求指標
- 指標組態
- Transfer Acceleration
- S3 事件通知
- 申請者付款儲存貯體
- S3 Select
- AWS Lambda 事件
- Server access logging (伺服器存取記錄日誌)
- HTTP POST 請求
- SOAP
- 網站存取

S3 on Outposts 網路需求

- 若要將請求路由至 S3 on Outpost 存取點，您必須建立和設定 S3 on Outposts 端點。下列限制適用於 S3 on Outpost 的端點：
  - Outpost 上的每個 Virtual Private Cloud (VPC) 可以有一個相關聯的端點，而且每個 Outpost 最多可以有 100 端點。
  - 您可以將多個存取點對應至同一個端點。
  - 您只能將端點新增至具有以下 CIDR 範圍子空間中 CIDR 區塊的 VPC：

  - 10.0.0.0/8
  - 172.16.0.0/12
  - 192.168.0.0/16
- 您只能從具有非重疊 CIDR 區塊的 VPC 建立 Outpost 端點。
- 您只能從其 Outpost 子網路內建立端點。
- 您用於建立端點的子網路必須包含可供 S3 on Outpost 使用的四個 IP 地址。
- 如果您指定客戶擁有的 IP 地址集區 (CoIP 集區)，它必須包含可供 S3 on Outposts 使用的四個 IP 地址。
- 每個 VPC 的每個 Outpost 只能建立一個端點。

Amazon S3 on Outposts 入門

透過 Amazon S3 on Outposts，您可以在 AWS Outposts 上建立 S3 儲存貯體，並針對需要本機資料存取權限、本機資料處理和資料駐留的應用程式，在內部部署輕鬆存放和擷取物件。S3 on Outposts 提供一個全新的儲存類別，即 S3 Outposts (OUTPOSTS)，它使用 Amazon S3 API，目的是在您的 AWS Outposts Outposts 上的多個裝置和伺服器上以持久、備援的方式存放資料。您可以使用存取點和透過 Virtual Private Cloud (VPC) 的端點連線，與您的 Outpost 儲存貯體進行通訊。就像在 Amazon S3 儲存貯體一樣，您在 Outpost 儲存貯體上可以使用同樣的 API 和功能，包括存取政策、加密和標記。您可以透過 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。

有了 Amazon S3 on Outposts，您就可以像在 Amazon S3 上操作一樣，在 AWS Outposts 上使用 Amazon S3 API 和物件儲存、存取政策、加密和標記等功能。如需 S3 on Outposts 的資訊，請參閱 什麼是 Amazon S3 on Outposts？。

主題

- 使用 S3 on Outposts 設定 IAM
- 使用 AWS Management Console 開始使用
- 使用 AWS CLI 和 SDK for Java 開始使用

使用 S3 on Outposts 設定 IAM

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員可控制哪些人員可進行身分驗證 (登入) 並獲得授權 (具有許可) 以使用 Amazon S3 on

Outposts 資源。IAM 是一種您可以免費使用的 AWS 服務。根據預設，使用者不具備 S3 on Outposts 資源和操作的許可。若要授予 S3 on Outposts 資源和 API 操作的存取權限，您可以使用 IAM 建立使用者、群組或角色，並附加許可。

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 使用者和群組位於 AWS IAM Identity Center：

  建立權限合集。請按照 AWS IAM Identity Center 使用者指南 中的 建立權限合集 說明進行操作。
- 透過身分提供者在 IAM 中管理的使用者：

  建立聯合身分的角色。請按照 IAM 使用者指南 的 為第三方身分提供者 (聯合) 建立角色 中的指示進行操作。
- IAM 使用者：
  - 建立您的使用者可擔任的角色。請按照 IAM 使用者指南 的 為 IAM 使用者建立角色 中的指示進行操作。
  - (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循 IAM 使用者指南的新增許可到使用者 (主控台)中的指示。

除了 IAM 身分型政策外，S3 on Outposts 也同時支援儲存貯體和存取點政策。儲存貯體政策與存取點政策是連接到 S3 on Outposts 資源的資源型政策。

- 儲存貯體政策連接到儲存貯體，並根據政策中的元素來允許或拒絕對儲存貯體和其中物件的請求。
- 相比之下，存取點原則連接到存取點，並允許或拒絕對存取點的要求。

存取點政策可搭配連接到基礎 S3 on Outposts 儲存貯體的儲存貯體政策。若要讓應用程式或使用者能夠透過 S3 on Outposts 存取點來存取 S3 on Outposts 儲存貯體中的物件，則存取點政策和儲存貯體政策皆必須允許該請求。

您在存取點政策中包含的限制僅適用透過該存取點進行的請求。例如，如果存取點連接到儲存貯體，則無法使用存取點政策來允許或拒絕直接向儲存貯體提出的請求。不過，您套用至儲存貯體政策的限制可能會允許或拒絕直接向儲存貯體或透過存取點提出的請求。

在 IAM 政策或資源型政策中，您可以定義哪些 S3 on Outposts 動作會受到允許或拒絕。S3 on Outposts 動作對應特定的 S3 on Outposts API 操作。S3 on Outposts 動作會使用 `s3-outposts:` 命名空間字首。在 Outposts 控制 API 上向 S3 發出的請求，以 AWS 區域 及對 Outpost 上的物件 API 端點發出的請求會使用 IAM 進行身份驗證，並根據`s3-outposts:`命名空間前綴進行授權。若要使用 S3 on Outposts，請設定您的 IAM 使用者，並針對 `s3-outposts:` IAM 命名空間授權使用者。

如需詳細資訊，請參閱《服務授權參考》中的「S3 on Outposts 的動作、資源和條件金鑰」。

> **Note**
>
> - S3 on Outposts 不支援存取控制清單 (ACL)。
> - Posts 上的 S3 預設會將儲存貯體擁有者做為物件擁有者，以協助確保儲存貯體的擁有者無法存取或刪除物件。
> - Outpost 上的 S3 一律會啟用 S3 封鎖公有存取權限，以協助確保物件永遠不會有公有存取權限。

如需有關 S3 on Outposts 設定 IAM 的詳細資訊，請參下列主題。

主題

- 適用於 S3 on Outposts 政策的主體
- 適用於 S3 on Outposts 的資源 ARN
- 適用於 S3 on Outposts 的範例政策
- 適用於 S3 on Outposts 端點的許可
- S3 on Outposts 的服務連結角色

適用於 S3 on Outposts 政策的主體

當您建立資源型政策以授予 S3 on Outposts 儲存貯體的存取權時，您必須使用 `Principal` 元素來指定可對該資源提出動作或操作請求的人員或應用程式。對於 S3 on Outposts 政策，您可以使用下列其中一項主體：

- 一個 AWS 帳戶
- IAM 使用者
- IAM 角色
- 政策中使用 `Condition` 元素來限制對特定 IP 範圍存取的所有主體 (指定萬用字元 *)

> **Important**
>
> 您無法針對在 `Principal` 元素中使用萬用字元 (*) 的 S3 on Outposts 儲存貯體來撰寫政策，除非該政策還包含限制對特定 IP 地址範圍存取的 `Condition`。此限制協助確保無法公有存取 S3 on Outposts 儲存貯體。如需範例，請參閱適用於 S3 on Outposts 的範例政策。

如需有關 `Principal` 元素的詳細資訊，請參閱《IAM 使用者指南》中的「AWS JSON 政策元素：主體」。

適用於 S3 on Outposts 的資源 ARN

Outposts S3 的 Amazon 資源名稱 (ARN) 除了前哨所在地、ID 和資源名稱之外，還包含前哨 AWS 帳戶 識別碼。 AWS 區域 若要存取並對 Outposts 儲存貯體和物件執行動作，您必須使用下表中顯示的其中一個 ARN 格式。

ARN 中的*`partition`*值是指的是一組 AWS 區域。每個分區 AWS 帳戶 的範圍都是一個分區。以下是支援的分割區：

- `aws` – AWS 區域
- `aws-us-gov`— AWS GovCloud (US) 地區

S3 on Outposts ARN 格式

Amazon S3 on Outposts ARN	ARN 格式	範例
儲存貯體 ARN	`arn:`*`partition`*` :s3-outposts: `*`region`*`: `*`account_id`*` :outpost/`*`outpost_id`*`/bucket/`*`bucket_name`	`arn:`*`aws`*`:s3-outposts: `*`us-west-2`*` :`*`123456789012`*` :outpost/ `*`op-01ac5d28a6a232904`*` /bucket/`*`example-s3-bucket1`
存取點 ARN	`arn:`*`partition`*` :s3-outposts: `*`region`*`: `*`account_id`*` :outpost/ `*`op-01ac5d`	`arn:`*`aws`*`:s3-outposts: `*`us-west-2`*` :`*`123456789012`*` :outpost/ `*`op-01ac5d`

Amazon S3 on Outposts ARN	ARN 格式	範例
	/ *outpost_id* /accesspoint/ *accesspoint_name*	*28a6a232904* /accesspoint/ *access-point-name*
物件 ARN	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /bucket/*bucket_name* /object/*object_key*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /bucket/*example-s3-bucket1* /object/*myobject*
S3 on Outposts 存取點物件 ARN (用於原則)	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /accesspoint/ *accesspoint_name* /object/*object_key*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /accesspoint/ *access-point-name*/object/*myobject*
S3 on Outposts ARN	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904*

適用於 S3 on Outposts 的範例政策

Example ：S3 在 Outposts 存儲桶政策上具有主體 AWS 帳戶

下列儲存貯體政策使用 AWS 帳戶 主體授與 Outposts 儲存貯體上 S3 的存取權。若要使用此儲存貯體政策，請以您自己的資訊取代 *user input placeholders*。

```
{
   "Version":"2012-10-17",
```

```
   "Id":"ExampleBucketPolicy1",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"123456789012"
         },
         "Action":"s3-outposts:*",
         "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket"
      }
   ]
}
```

Example ：使用萬用字元 (*) 主體和條件金鑰的 S3 on Outposts 儲存貯體政策，以限制對特定 IP 地址範圍的存取

下列儲存貯體政策使用萬用字元主體 (*) 搭配 aws:SourceIp 條件以限制對特定 IP 地址範圍的存取。若要使用此儲存貯體政策，請以您自己的資訊取代 *user input placeholders*。

```
{
    "Version": "2012-10-17",
    "Id": "ExampleBucketPolicy2",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": { "AWS" : "*" },
            "Action":"s3-outposts:*",
            "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket",
            "Condition" : {
                "IpAddress" : {
                    "aws:SourceIp": "192.0.2.0/24"
                },
                "NotIpAddress" : {
                    "aws:SourceIp": "198.51.100.0/24"
                }
            }
        }
```

```
    ]
}
```

適用於 S3 on Outposts 端點的許可

S3 on Outposts 需要 IAM 中擁有的許可，來管理 S3 on Outposts 端點動作。

> **Note**
>
> - 對於使用客戶擁有的 IP 地址集區 (CoIP 集區) 存取類型的端點，您也必須具有從 CoIP 集區使 IP 地址的許可，如下表所述。
> - 對於在 Outposts 上使用存取 S3 的共用帳戶 AWS Resource Access Manager，這些共用帳戶中的使用者無法在共用子網路上建立自己的端點。如果共用帳戶中的使用者想要管理自己的端點，則共用帳戶必須在 Outpost 上建立自己的子網路。如需詳細資訊，請參閱 the section called “共用 S3 on Outposts”。

S3 on Outposts 端點相關的 IAM 許可

動作	IAM 許可
`CreateEndpoint`	`s3-outposts:CreateEndpoint` `ec2:CreateNetworkInterface` `ec2:DescribeNetworkInterfaces` `ec2:DescribeVpcs` `ec2:DescribeSecurityGroups` `ec2:DescribeSubnets` `ec2:CreateTags` `iam:CreateServiceLinkedRole` 對於使用內部部署客戶擁有的 IP 地址集區 (CoIP 集區) 存取類型的端點，需要下列額外的許可：

動作	IAM 許可
	s3-outposts:CreateEndpoint ec2:DescribeCoipPools ec2:GetCoipPoolUsage ec2:AllocateAddress ec2:AssociateAddress ec2:DescribeAddresses ec2:DescribeLocalGatewayRou teTableVpcAssociations
DeleteEndpoint	s3-outposts:DeleteEndpoint ec2:DeleteNetworkInterface ec2:DescribeNetworkInterfaces 對於使用內部部署客戶擁有的 IP 地址集區 (CoIP 集區) 存取類型的端點，需要下列額外的許可： s3-outposts:DeleteEndpoint ec2:DisassociateAddress ec2:DescribeAddresses ec2:ReleaseAddress
ListEndpoints	s3-outposts:ListEndpoints

> **Note**
>
> 您可以在 IAM 政策中使用資源標籤來管理許可。

S3 on Outposts 的服務連結角色

S3 on Outposts 使用 IAM 服務連結角色代表您建立一些網路資源。如需更多詳細資訊，請參閱 針對 Amazon S3 on Outposts 使用服務連結角色。

使用 AWS Management Console 開始使用

透過 Amazon S3 on Outposts，您可以在 AWS Outposts 上建立 S3 儲存貯體，並針對需要本機資料存取權限、本機資料處理和資料駐留的應用程式，在內部部署輕鬆存放和擷取物件。S3 on Outposts 提供一個全新的儲存類別，即 S3 Outposts (0UTP0STS)，它使用 Amazon S3 API，目的是在您的 AWS Outposts Outposts 上的多個裝置和伺服器上以持久、備援的方式存放資料。您可以使用存取點和透過 Virtual Private Cloud (VPC) 的端點連線，與您的 Outpost 儲存貯體進行通訊。就像在 Amazon S3 儲存貯體一樣，您在 Outpost 儲存貯體上可以使用同樣的 API 和功能，包括存取政策、加密和標記。您可以透過 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。如需詳細資訊，請參閱 什麼是 Amazon S3 on Outposts？。

若要開始透過主控台使用 S3 on Outposts，請參閱下列主題。若要透過使用 AWS CLI 或 AWS SDK for Java 來開始，請參閱 使用 AWS CLI 和 SDK for Java 開始使用。

主題

- 建立儲存貯體、存取點和端點
- 後續步驟

建立儲存貯體、存取點和端點

下列程序示範如何在 S3 on Outposts 中建立第一個儲存貯體。當您使用主控台建立儲存貯體時，您還會建立一個存取點和與儲存貯體關聯的端點，以便您可以立即開始在儲存貯體中存儲物件。

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇 Create Outposts bucket (建立 Outposts 儲存貯體)。
4. 在 Bucket name (儲存貯體名稱) 中，為儲存貯體輸入符合網域名稱系統 (DNS) 規範的名稱。

   儲存貯體名稱必須；

   - 在 AWS 帳戶、Outpost，以及 Outpost 所在 AWS 區域之內都是獨一無二的。

- 長度必須介於 3–63 個字元之間。
- 不含大寫字元。
- 以小寫字母或數字開頭。

建立儲存貯體後，便無法變更其名稱。如需命名儲存貯體的資訊，請參閱「儲存貯體命名規則」。

> **Important**
>
> 避免在儲存貯體名稱中包含敏感資訊，例如帳戶號碼。在指向儲存貯體中之物件的 URL 中，會顯示儲存貯體名稱。

5. 在 Outpost 中，選擇您要儲存貯體駐留的 Outpost。
6. 在 Bucket Versioning (儲存貯體版本控制) 下，將 S3 on Outposts 儲存貯體的 S3 版本控制狀態設定為下列其中一個選項：

   - Disable (停用) (預設) - 儲存貯體會保留未版本控制的狀態。
   - Enable (啟用) - 針對儲存貯體中的物件啟用 S3 版本控制。所有新增至儲存貯體的物件都會收到唯一的版本 ID。

   如需 S3 版本控制的詳細資訊，請參閱「針對您的 S3 on Outposts 儲存貯體管理 S3 版本控制」。
7. (選擇性) 新增要與 Outposts 儲存貯體建立關聯的任何 optional tags (選用標籤)。您可以使用標籤來追蹤個別專案或一組專案的條件，或者使用成本分配標籤來標示儲存貯體。

   存放在 Outposts 儲存貯體中的所有物件預設為使用伺服器端加密與 Amazon S3 受管加密金鑰 (SSE-S3) 進行儲存。您也可以明確選擇使用伺服器端加密與客戶提供的加密金鑰 (SSE-C) 來存放物件。要更改加密類型，必須使用 REST API、AWS Command Line Interface (AWS CLI) 或 AWS 開發套件。
8. 在 Outposts access point settings (Outposts 存取點設定) 區段中，輸入存取點名稱。

   S3 on Outposts 存取點針對 S3 on Outposts 中的共用資料集，簡化了對大規模資料存取的管理。存取點為連接到 Outposts 儲存貯體的具名網路端點，您可以使用這些端點來執行 S3 物件操作。如需更多詳細資訊，請參閱 存取點。

   存取點名稱在此區域和 Outpost 的帳戶中必須是唯一的，並且符合 存取點的法規與限制。
9. 對此 Amazon S3 on Outposts 存取點選擇 VPC。

如果您沒有 VPC，請選擇 Create VPC (建立 VPC)。如需更多詳細資訊，請參閱 建立受限於 Virtual Private Cloud 的存取點。

Virtual Private Cloud (VPC) 可讓您將 AWS 資源啟動到您定義的虛擬網路。這個虛擬網路與您在資料中心中操作的傳統網路非常相似，且具備使用 可擴展基礎設施的優勢AWS

10. (對於現有 VPC 可選) 請為您的端點選擇Endpoint subnet (端點子網路)。

    子網路是您的 VPC 中的 IP 地址範圍。如果您沒有想要的子網路，請選擇 Create subnet (建立子網路)。如需更多詳細資訊，請參閱 適用於 S3 on Outposts 的網路。

11. (對於現有 VPC 可選) 請為您的端點選擇Endpoint security group (端點安全群組)。

    安全群組做為虛擬防火牆以控制傳入及傳出流量。

12. (對於現有 VPC 可選) 請選擇 Endpoint access type (端點存取類型)：

    - 私有 - 與 VPC 一起使用。
    - 客戶擁有的 IP - 與內部部署網路內的客戶擁有的 IP 地址集區 (CoIP 集區) 一起使用。

13. (選用) 指定Outpost 存取點政策。主控台會自動顯示存取點的 Amazon Resource Name (ARN)，您可以在政策中使用該名稱。
14. 選擇 Create Outposts bucket (建立 Outposts 儲存貯體)。

> **Note**
>
> 建立您的 Outpost 端點以及您的儲存貯體準備就緒可以使用，可能需要 5 分鐘。若要設定其他儲存貯體設定，請選擇 View details (檢視詳細資訊)

後續步驟

對於 Amazon S3 on Outposts，物件資料始終存放在 Outpost 上。當 AWS 安裝 Outpost 機架時，您的資料將保留在本機 Outpost 上，以滿足資料駐留的要求。您的物件永遠不會離開您的 Outpost，也不會在 AWS 區域 中。由於 AWS Management Console 託管在區域內，您無法使用主控台上傳或管理 Outpost 中的物件。然而，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 以及 AWS SDK 透過存取點上傳和管理您的物件。

您建立 S3 on Outposts 儲存貯體、存取點和端點之後，便可以使用 AWS CLI 或適用於 Java 的 SDK，將物件上傳至您的儲存貯體中。如需更多詳細資訊，請參閱 將對象上傳到 Outposts 存儲桶上的 S3。

使用 AWS CLI 和 SDK for Java 開始使用

使用 Outposts 上的 Amazon S3，您可以在 Outposts 上建立 S3 儲存貯體，並輕鬆地在現場部署存放和擷取物件，以供需要本機資料存取、本機資料處理和資料存放區的應用程式使用。 AWS Outposts 上的 S3 提供新的儲存類別 S3 Outposts (OUTPOSTS)，該類別使用 Amazon S3 API，其設計用於在您的多個裝置和伺服器上持久且冗餘地存放資料。 AWS Outposts您可以使用存取點和透過 Virtual Private Cloud (VPC) 的端點連線，與您的 Outpost 儲存貯體進行通訊。就像在 Amazon S3 儲存貯體一樣，您在 Outpost 儲存貯體上可以使用同樣的 API 和功能，包括存取政策、加密和標記。您可以透過 AWS Management Console、 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 REST API 在 Outposts 上使用 S3。如需詳細資訊，請參閱 什麼是 Amazon S3 on Outposts？。

若要開始使用 S3 on Outposts，您必須建立儲存貯體、存取點和端點。接著，您可以將物件上傳至您的儲存貯體。下列範例說明如何使用 AWS CLI 和 SDK for Java，在 Outposts 上開始使用 S3。若要開始使用主控台，請參閱 使用 AWS Management Console 開始使用。

主題

步驟 1：建立儲存貯體

以下 AWS CLI 和 SDK for Java 示例向您展示瞭如何在 Outposts 存儲桶上創建 S3。

AWS CLI

Example

下列範例使用 AWS CLI建立 S3 on Outposts 儲存貯體 (`s3-outposts:CreateBucket`)。若要執行此命令，請以您自己的資訊取代 *`user input placeholders`*。

```
aws s3control create-bucket --bucket example-outposts-bucket --outpost-
id op-01ac5d28a6a232904
```

SDK for Java

Example

下列範例使用適用於 Java 的開發套件建立 S3 on Outposts 儲存貯體 (`s3-outposts:CreateBucket`)。

```
import com.amazonaws.services.s3control.model.*;

public String createBucket(String bucketName) {

    CreateBucketRequest reqCreateBucket = new CreateBucketRequest()
            .withBucket(bucketName)
            .withOutpostId(OutpostId)
            .withCreateBucketConfiguration(new CreateBucketConfiguration());

    CreateBucketResult respCreateBucket =
 s3ControlClient.createBucket(reqCreateBucket);
    System.out.printf("CreateBucket Response: %s%n", respCreateBucket.toString());

    return respCreateBucket.getBucketArn();

}
```

步驟 2：建立存取點

若要存取 Amazon S3 on Outposts 儲存貯體，您必須建立和設定存取點。這些範例如何使用 AWS CLI 和 Java 的 SDK 來建立存取點。

存取點針對 Amazon S3 中的共用資料集，簡化管理大規模的資料存取。存取點為連接到儲存貯體的指定網路端點，您可以使用這些端點來執行 Amazon S3 物件操作，例如 `GetObject` 和 `PutObject`。使用 S3 on Outposts，您必須使用存取點來存取 Outposts 儲存貯體中的任何物件。存取點僅支援 virtual-host-style 定址。

AWS CLI

Example

下列 AWS CLI 範例會為 Outposts 值區建立存取點。若要執行此命令，請以您自己的資訊取代 *`user input placeholders`*。

```
aws s3control create-access-point --account-id 123456789012
 --name example-outposts-access-point --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket" --vpc-configuration VpcId=example-vpc-12345
```

SDK for Java

Example

下列適用於 Java 的開發套件範例建立 Outposts 儲存貯體的存取點。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
import com.amazonaws.services.s3control.model.*;

public String createAccessPoint(String bucketArn, String accessPointName) {

    CreateAccessPointRequest reqCreateAP = new CreateAccessPointRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withName(accessPointName)
            .withVpcConfiguration(new VpcConfiguration().withVpcId("vpc-12345"));

    CreateAccessPointResult respCreateAP =
 s3ControlClient.createAccessPoint(reqCreateAP);
    System.out.printf("CreateAccessPoint Response: %s%n", respCreateAP.toString());

    return respCreateAP.getAccessPointArn();

}
```

步驟 3：建立端點

若要將請求路由至 Amazon S3 on Outpost 存取點，您必須建立和設定 S3 on Outposts 端點。若要建立端點，您需要與 Outposts 主要區域的服務連結有效連接。Outpost 上的每個 Virtual Private Cloud (VPC) 可以有一個相關聯的端點。如需端點配額的詳細資訊，請參閱 S3 on Outposts 網路需求。您必須建立端點，才能存取您的 Outposts 儲存貯體並執行物件操作。如需詳細資訊，請參閱 端點。

這些範例說明如何使用 AWS CLI 和 SDK for Java 來建立端點。如需建立和管理端點所需許可的詳細資訊，請參閱 適用於 S3 on Outposts 端點的許可。

AWS CLI

Example

下列 AWS CLI 範例使用 VPC 資源存取類型為 Outpost 建立端點。VPC 衍生自子網路。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1
```

下列 AWS CLI 範例會使用客戶擁有的 IP 位址集區 (CoIP 集區) 存取類型，為 Outpost 建立端點。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1 --access-type CustomerOwnedIp --
customer-owned-ipv4-pool ipv4pool-coip-12345678901234567
```

SDK for Java

Example

下列適用於 Java 的開發套件範例建立 Outposts 的端點。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.CreateEndpointRequest;
import com.amazonaws.services.s3outposts.model.CreateEndpointResult;

public void createEndpoint() {
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    CreateEndpointRequest createEndpointRequest = new CreateEndpointRequest()
                .withOutpostId("op-0d79779cef3c30a40")
                .withSubnetId("subnet-8c7a57c5")
                .withSecurityGroupId("sg-ab19e0d1")
                .withAccessType("CustomerOwnedIp")
                .withCustomerOwnedIpv4Pool("ipv4pool-coip-12345678901234567");
    // Use .withAccessType and .withCustomerOwnedIpv4Pool only when the access type
 is
```

```
    // customer-owned IP address pool (CoIP pool)
    CreateEndpointResult createEndpointResult =
 s3OutpostsClient.createEndpoint(createEndpointRequest);
    System.out.println("Endpoint is created and its ARN is " +
 createEndpointResult.getEndpointArn());
}
```

步驟 4：將物件上傳至 S3 on Outposts 儲存貯體

若要上傳物件，請參閱 將對象上傳到 Outposts 存儲桶上的 S3。

適用於 S3 on Outposts 的網路

您可以使用 Amazon S3 on Outposts，針對需要本機資料存取、資料處理和資料駐留的應用程式，來存放和擷取內部部署物件。此部分說明存取 S3 on Outposts 的網路要求。

主題

- 選擇您的網路存取類型
- 存取 S3 on Outposts 儲存貯體和物件
- 跨帳戶彈性網路界面

選擇您的網路存取類型

您可以從 VPC 內或內部部署網路存取 S3 on Outposts。您可以使用存取點和透過端點連線，與您的 Outposts 儲存貯體進行通訊。此連線會在 AWS 網路內確保 VPC 與 S3 on Outposts 儲存貯體之間的流量。建立端點後，您必須將端點存取類型指定為 `Private` (適用於 VPC 路由) 或 `CustomerOwnedIp` (適用於客戶自訂 IP 地址集區 (CoIP 集區))。

- `Private` (針對 VPC 路由路由) - 若不指定存取類型，則預設 S3 on Outposts 會使用 `Private`。使用 `Private`存取類型，則 VPC 中的執行個體不需要公有 IP 地址，即可與 Outposts 中的資源通訊。您可以使用在 VPC 內使用 S3 on Outposts。這類端點無法透過直接 VPC 路由從內部部署網路存取。如需詳細資訊，請參閱《AWS Outposts 使用者指南》中的本機閘道路油表。
- `CustomerOwnedIp` (針對 CoIP 集區) - 如果您預設不使用 `Private` 存取類型，然後選擇 `CustomerOwnedIp`，您必須指定 IP 地址範圍。您可以使用此存取類型在內部部署網路和 VPC 內使用 S3 on Outposts。在 VPC 內存取 S3 on Outposts 時，您的流量將受限於本機閘道的頻寬。

存取 S3 on Outposts 儲存貯體和物件

若要存取 S3 on Outposts 儲存貯體和物件，必須具備下列項目：

- VPC 的存取點。
- 相同 VPC 的端點。
- Outpost 與 AWS 區域 之間的作用中連線。如需有關如何將 Outpost 連接到某個區域的詳細資訊，請參閱《AWS Outposts 使用者指南》中的AWS 區域的 Outpost 連線。

如需有關存取 S3 on Outposts 中儲存貯體和物件的詳細資訊，請參閱 使用 S3 on Outposts 儲存貯體 和 使用 S3 on Outposts 物件。

跨帳戶彈性網路界面

S3 on Outposts 端點是使用 Amazon 資源名稱 (ARN) 命名的資源。建立這些端點後，AWS Outposts 會設定多個跨帳戶彈性網路界面。S3 on Outposts 跨帳戶彈性網路界面與其他網路界面類型，但存在以下例外狀況：S3 on Outposts 會將跨帳戶彈性網路界面與 Amazon EC2 執行個體關聯。

S3 on Outposts 網域名稱系統 (DNS) 負載會透過跨帳戶彈性網路界面來平衡您的請求。S3 on Outposts 會在您的 AWS 帳戶中建立跨帳戶彈性網路界面，其可在 Amazon EC2 主控台的網路界面窗格中看到。

對於使用 CoIP 集區存取類型的端點，S3 on Outposts 會從設定的 CoIP 集區配置 IP 地址，並將其與跨帳戶彈性網路界面建立關聯。

使用 S3 on Outposts 儲存貯體

透過 Amazon S3 on Outposts，您可以在 AWS Outposts 上建立 S3 儲存貯體，並針對需要本機資料存取權限、本機資料處理和資料駐留的應用程式，在內部部署輕鬆存放和擷取物件。S3 on Outposts 提供一個全新的儲存類別，即 S3 Outposts (OUTPOSTS)，它使用 Amazon S3 API，目的是在您的 AWS Outposts Outposts 上的多個裝置和伺服器上以持久、備援的方式存放資料。就像在 Amazon S3 一樣，您在 Outpost 儲存貯體上可以使用同樣的 API 和功能，包括存取政策、加密和標記。如需詳細資訊，請參閱 什麼是 Amazon S3 on Outposts？

您可以使用存取點和透過 Virtual Private Cloud (VPC) 的端點連線，與您的 Outposts 儲存貯體進行通訊。若要存取 S3 on Outposts 儲存貯體和物件，必須具備 VPC 的存取點，並且必須具備相同 VPC 的端點。如需更多詳細資訊，請參閱 適用於 S3 on Outposts 的網路。

儲存貯體

在 S3 on Outposts 中，儲存貯體名稱對 Outposts 是唯一的名稱，並且需要 Outpost所在區域的 AWS 區域 代碼、AWS 帳戶 ID、Outpost ID 和用於識別的儲存貯體名稱。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/bucket/bucket-name
```

如需更多詳細資訊，請參閱 適用於 S3 on Outposts 的資源 ARN。

存取點

Amazon S3 on Outposts 支援僅限 Virtual Private Cloud (VPC) 的存取點，作為存取您的 Outpost 儲存貯體的唯一方法。

存取點針對 Amazon S3 中的共用資料集，簡化管理大規模的資料存取。存取點為連接到儲存貯體的指定網路端點，您可以使用這些端點來執行 Amazon S3 物件操作，例如 `GetObject` 和 `PutObject`。使用 S3 on Outposts，您必須使用存取點來存取 Outposts 儲存貯體中的任何物件。存取點僅支援虛擬主機樣式定址。

下列範例顯示了 S3 on Outposts 存取點使用的 ARN 格式。存取點 ARN 包括 Outpost 所在區域的 AWS 區域 程式碼、AWS 帳戶 ID、Outpost ID 和存取點名稱。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

端點

若要將請求路由至 S3 on Outpost 存取點，您必須建立和設定 S3 on Outposts 端點。對於 S3 on Outposts 端點，您可以將 VPC 私密地連線至 Outposts 儲存貯體。S3 on Outposts 端點是 Outposts 儲存貯體 S3 進入點的虛擬統一資源識別符 (URI)。這些端點是水平擴展、冗餘且高度可用的 VPC 元件。

Outpost 上的每個 Virtual Private Cloud (VPC) 可以有一個相關聯的端點，而且每個 Outpost 最多可以有 100 端點。您必須建立這些端點，才能存取您的 Outpost 儲存貯體並執行物件操作。建立這些端點也會讓相同的操作在 S3 和 S3 on Outposts 中運作，使 API 模型和行為相同。

適用於 S3 on Outposts 的 API

若要管理 Outpost 儲存貯體 API 操作，S3 on Outposts 會託管一個與 Amazon S3 端點不同的獨立端點。此端點為 `s3-outposts.region.amazonaws.com`。

若要使用 Amazon S3 API 操作，必須使用正確的 ARN 格式簽署儲存貯體和物件。您必須將 API 操作傳遞給 ARN，以便 Amazon S3 確定請求是針對Amazon S3 (`s3-control.region.amazonaws.com`) 還是 S3 on Outposts (`s3-outposts.region.amazonaws.com`)。根據 ARN 格式，S3 隨後可以適當地簽署和路由請求。

每當將請求傳送至 Amazon S3 控制平面時，開發套件會從 ARN 中擷取元件，並包含一個額外的標頭 `x-amz-outpost-id`，其中包含從 ARN 擷取的 *outpost-id* 值。來自 ARN 的服務名稱用來在路由傳送到 S3 on Outposts 端點之前簽署請求。該行為適用於所有由 `s3control` 用戶端處理的 API 操作。

下表列出了 Amazon S3 on Outposts 的擴展 API 操作，及其相對於 Amazon S3 的變更。

API	S3 on Outposts 參數值
`CreateBucket`	作為 ARN 的儲存貯體名稱，Outpost ID
`ListRegionalBuckets`	Outpost ID
`DeleteBucket`	作為 ARN 的儲存貯體名稱
`DeleteBucketLifecycleConfiguration`	作為 ARN 的儲存貯體名稱
`GetBucketLifecycleConfiguration`	作為 ARN 的儲存貯體名稱
`PutBucketLifecycleConfiguration`	作為 ARN 的儲存貯體名稱
`GetBucketPolicy`	作為 ARN 的儲存貯體名稱
`PutBucketPolicy`	作為 ARN 的儲存貯體名稱
`DeleteBucketPolicy`	作為 ARN 的儲存貯體名稱
`GetBucketTagging`	作為 ARN 的儲存貯體名稱
`PutBucketTagging`	作為 ARN 的儲存貯體名稱

API	S3 on Outposts 參數值
DeleteBucketTagging	作為 ARN 的儲存貯體名稱
CreateAccessPoint	作為 ARN 的存取點名稱
DeleteAccessPoint	作為 ARN 的存取點名稱
GetAccessPoint	作為 ARN 的存取點名稱
GetAccessPoint	作為 ARN 的存取點名稱
ListAccessPoints	作為 ARN 的存取點名稱
PutAccessPointPolicy	作為 ARN 的存取點名稱
GetAccessPointPolicy	作為 ARN 的存取點名稱
DeleteAccessPointPolicy	作為 ARN 的存取點名稱

建立和管理 S3 on Outposts 儲存貯體

如需有關建立和管理 S3 on Outposts 儲存貯體的詳細資訊，請參閱下列主題。

主題

- 建立 S3 on Outposts 儲存貯體
- 新增 S3 on Outposts 儲存貯體的標籤
- 使用儲存貯體政策管理 Amazon S3 on Outposts 儲存貯體的存取
- 列出 Amazon S3 on Outposts 儲存貯體
- 使用 AWS CLI 和適用於 Java 的開發套件取得 S3 on Outposts 儲存貯體
- 刪除 Amazon S3 on Outposts 儲存貯體
- 使用 Amazon S3 on Outposts 存取點
- 使用 Amazon S3 on Outposts 端點

建立 S3 on Outposts 儲存貯體

透過 Amazon S3 on Outposts，您可以在 AWS Outposts 上建立 S3 儲存貯體，並針對需要本機資料存取權限、本機資料處理和資料駐留的應用程式，在內部部署輕鬆存放和擷取物件。S3 on Outposts 提供一個全新的儲存類別，即 S3 Outposts (OUTPOSTS)，它使用 Amazon S3 API，目的是在您的 AWS Outposts Outposts 上的多個裝置和伺服器上以持久、備援的方式存放資料。您可以使用存取點和透過 Virtual Private Cloud (VPC) 的端點連線，與您的 Outpost 儲存貯體進行通訊。就像在 Amazon S3 儲存貯體一樣，您在 Outpost 儲存貯體上可以使用同樣的 API 和功能，包括存取政策、加密和標記。您可以透過 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。如需詳細資訊，請參閱 什麼是 Amazon S3 on Outposts？。

> **Note**
>
> 建立儲存貯體的 AWS 帳戶 擁有該儲存貯體，且是唯一可以向其提交動作的帳戶。儲存貯體具有組態屬性，例如 Outpost、標籤、預設加密和存取點設定。存取點設定包含用於存取儲存貯體中物件的虛擬私有雲端 (VPC) 和存取點政策，以及其他中繼資料。如需更多詳細資訊，請參閱 S3 on Outposts 規格。
>
> 如果您想要建立一個儲存貯體，讓它透過您的虛擬私有雲端 (VPC) 中的介面 VPC 端點使用 AWS PrivateLink 提供儲存貯體和端點管理存取權，請參閱適用 S3 on Outposts 的 AWS PrivateLink。

下列範例示範如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 建立 S3 on Outposts 儲存貯體。

使用 S3 主控台

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇 Create Outposts bucket (建立 Outposts 儲存貯體)。
4. 在 Bucket name (儲存貯體名稱) 中，為儲存貯體輸入符合網域名稱系統 (DNS) 規範的名稱。

   儲存貯體名稱必須；

   - 在 AWS 帳戶、Outpost，以及 Outpost 所在 AWS 區域之內都是獨一無二的。
   - 長度必須介於 3–63 個字元之間。
   - 不含大寫字元。

- 以小寫字母或數字開頭。

  建立儲存貯體後，便無法變更其名稱。如需命名儲存貯體的資訊，請參閱「儲存貯體命名規則」。

  > **Important**
  >
  > 避免在儲存貯體名稱中包含敏感資訊，例如帳戶號碼。在指向儲存貯體中之物件的 URL 中，會顯示儲存貯體名稱。

5. 在 Outpost 中，選擇您要儲存貯體駐留的 Outpost。
6. 在 Bucket Versioning (儲存貯體版本控制) 下，將 S3 on Outposts 儲存貯體的 S3 版本控制狀態設定為下列其中一個選項：

   - Disable (停用) (預設) - 儲存貯體會保留未版本控制的狀態。
   - Enable (啟用) - 針對儲存貯體中的物件啟用 S3 版本控制。所有新增至儲存貯體的物件都會收到唯一的版本 ID。

   如需 S3 版本控制的詳細資訊，請參閱「針對您的 S3 on Outposts 儲存貯體管理 S3 版本控制」。
7. (選擇性) 新增要與 Outposts 儲存貯體建立關聯的任何 optional tags (選用標籤)。您可以使用標籤來追蹤個別專案或一組專案的條件，或者使用成本分配標籤來標示儲存貯體。

   存放在 Outposts 儲存貯體中的所有物件預設為使用伺服器端加密與 Amazon S3 受管加密金鑰 (SSE-S3) 進行儲存。您也可以明確選擇使用伺服器端加密與客戶提供的加密金鑰 (SSE-C) 來存放物件。要更改加密類型，必須使用 REST API、AWS Command Line Interface (AWS CLI) 或 AWS 開發套件。
8. 在 Outposts access point settings (Outposts 存取點設定) 區段中，輸入存取點名稱。

   S3 on Outposts 存取點針對 S3 on Outposts 中的共用資料集，簡化了對大規模資料存取的管理。存取點為連接到 Outposts 儲存貯體的具名網路端點，您可以使用這些端點來執行 S3 物件操作。如需更多詳細資訊，請參閱 存取點。

   存取點名稱在此區域和 Outpost 的帳戶中必須是唯一的，並且符合 存取點的法規與限制。
9. 對此 Amazon S3 on Outposts 存取點選擇 VPC。

   如果您沒有 VPC，請選擇 Create VPC (建立 VPC)。如需更多詳細資訊，請參閱 建立受限於 Virtual Private Cloud 的存取點。

Virtual Private Cloud (VPC) 可讓您將 AWS 資源啟動到您定義的虛擬網路。這個虛擬網路與您在資料中心中操作的傳統網路非常相似，且具備使用 可擴展基礎設施的優勢AWS

10. (對於現有 VPC 可選) 請為您的端點選擇Endpoint subnet (端點子網路)。

    子網路是您的 VPC 中的 IP 地址範圍。如果您沒有想要的子網路，請選擇 Create subnet (建立子網路)。如需更多詳細資訊，請參閱 適用於 S3 on Outposts 的網路。

11. (對於現有 VPC 可選) 請為您的端點選擇Endpoint security group (端點安全群組)。

    安全群組做為虛擬防火牆以控制傳入及傳出流量。

12. (對於現有 VPC 可選) 請選擇 Endpoint access type (端點存取類型)：

    - 私有 - 與 VPC 一起使用。
    - 客戶擁有的 IP - 與內部部署網路內的客戶擁有的 IP 地址集區 (CoIP 集區) 一起使用。

13. (選用) 指定Outpost 存取點政策。主控台會自動顯示存取點的 Amazon Resource Name (ARN)，您可以在政策中使用該名稱。

14. 選擇 Create Outposts bucket (建立 Outposts 儲存貯體)。

> **Note**
>
> 建立您的 Outpost 端點以及您的儲存貯體準備就緒可以使用，可能需要 5 分鐘。若要設定其他儲存貯體設定，請選擇 View details (檢視詳細資訊)

使用 AWS CLI

Example

下列範例使用 AWS CLI 建立 S3 on Outposts 儲存貯體 (`s3-outposts:CreateBucket`)。若要執行此命令，請以您自己的資訊取代 *`user input placeholders`*。

```
aws s3control create-bucket --bucket example-outposts-bucket --outpost-
id op-01ac5d28a6a232904
```

使用適用於 Java 的 AWS SDK

Example

下列範例使用適用於 Java 的開發套件建立 S3 on Outposts 儲存貯體 (`s3-outposts:CreateBucket`)。

```
import com.amazonaws.services.s3control.model.*;

public String createBucket(String bucketName) {

    CreateBucketRequest reqCreateBucket = new CreateBucketRequest()
            .withBucket(bucketName)
            .withOutpostId(OutpostId)
            .withCreateBucketConfiguration(new CreateBucketConfiguration());

    CreateBucketResult respCreateBucket =
 s3ControlClient.createBucket(reqCreateBucket);
    System.out.printf("CreateBucket Response: %s%n", respCreateBucket.toString());

    return respCreateBucket.getBucketArn();

}
```

新增 S3 on Outposts 儲存貯體的標籤

您可以新增 S3 on Outposts 儲存貯體標籤，以追蹤個別專案或專案群組的儲存成本和其他條件。

> **Note**
>
> 建立儲存貯體的 AWS 帳戶 擁有該儲存貯體，且是唯一可以變更其標籤的帳戶。

使用 S3 主控台

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇包含您要編輯其標籤的 Outposts 儲存貯體。
4. 選擇 Properties (屬性) 標籤。

5. 在 Tags (標籤) 下方，選擇 Edit (編輯)。
6. 選擇 Add new tag (新增標籤)，然後輸入 金鑰和選項值。

   新增要與 Outposts 儲存貯體關聯的任何標籤，以追蹤個別專案或專案群組的條件。
7. 選擇 Save changes (儲存變更)。

使用 AWS CLI

下列 AWS CLI 範例使用目前檔案夾中指定標籤 (*tagging.json*) 的 JSON 文件，將標籤組態套用於 S3 on Outposts 儲存貯體上。若要使用此範例，請以您自己的資訊取代每個 *user input placeholder*。

```
aws s3control put-bucket-tagging --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --tagging file://tagging.json

tagging.json

{
   "TagSet": [
     {
       "Key": "organization",
       "Value": "marketing"
     }
   ]
}
```

下列 AWS CLI 範例直接從命令列將標籤組態應用於 S3 on Outposts 儲存貯體上。

```
aws s3control put-bucket-tagging --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --tagging 'TagSet=[{Key=organization,Value=marketing}]'
```

如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 put-bucket-tagging。

使用儲存貯體政策管理 Amazon S3 on Outposts 儲存貯體的存取

儲存貯體政策是以資源為基礎的 AWS Identity and Access Management (IAM) 政策，您可以使用這些政策來將存取許可授予儲存貯體及其中物件。只有儲存貯體擁有者可建立政策與儲存貯體的關聯。連接

到儲存貯體的許可會套用至儲存貯體擁有者帳戶擁有的所有儲存貯體物件。儲存貯體政策的大小限制為 20 KB。如需詳細資訊，請參閱 儲存貯體政策。

您可以更新儲存貯體政策以管理對 Amazon S3 on Outposts 儲存貯體的存取。如需詳細資訊，請參閱下列主題。

主題

- 新增或編輯 Amazon S3 on Outposts 儲存貯體的儲存貯體政策
- 檢視 Amazon S3 on Outposts 儲存貯體的儲存貯體政策
- 刪除 Amazon S3 on Outposts 儲存貯體的儲存貯體政策
- 儲存貯體政策範例

新增或編輯 Amazon S3 on Outposts 儲存貯體的儲存貯體政策

儲存貯體政策是以資源為基礎的 AWS Identity and Access Management (IAM) 政策，您可以使用這些政策來將存取許可授予儲存貯體及其中物件。只有儲存貯體擁有者可建立政策與儲存貯體的關聯。連接到儲存貯體的許可會套用至儲存貯體擁有者帳戶擁有的所有儲存貯體物件。儲存貯體政策的大小限制為 20 KB。如需詳細資訊，請參閱 儲存貯體政策。

下列主題示範如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 或 AWS SDK for Java 更新 Amazon S3 on Outposts 儲存貯體政策。

使用 S3 主控台

建立或編輯儲存貯體政策

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇您要編輯其儲存貯體政策的 Outposts 儲存貯體。
4. 選擇 Permissions (許可) 標籤。
5. 在 Outposts 儲存貯體政策部分，若要建立或編輯新政策，請選擇Edit (編輯)。

   現在您可以新增或編輯 S3 on Outposts 儲存貯體政策。如需詳細資訊，請參閱 使用 S3 on Outposts 設定 IAM。

使用 AWS CLI

下列 AWS CLI 範例將政策放置在 Outposts 儲存貯體上。

1. 將以下儲存貯體政策儲存到 JSON 檔案中。在此範例中，檔案命名為 policy1.json。以您自己的資訊取代 *user input placeholders*。

```
{
   "Version":"2012-10-17",
   "Id":"testBucketPolicy",
   "Statement":[
      {
         "Sid":"st1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"123456789012"
         },
         "Action":"s3-outposts:*",
         "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket"
      }
   ]
}
```

2. 提交 JSON 檔案以做為 put-bucket-policy CLI 命令的一部分。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control put-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --policy file://policy1.json
```

使用適用於 Java 的 AWS SDK

下列適用於 Java 的開發套件範例在 Outposts 儲存貯體上放置政策。

```
import com.amazonaws.services.s3control.model.*;

public void putBucketPolicy(String bucketArn) {
```

```
    String policy = "{\"Version\":\"2012-10-17\",\"Id\":\"testBucketPolicy\",
\"Statement\":[{\"Sid\":\"st1\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"" +
 AccountId+ "\"},\"Action\":\"s3-outposts:*\",\"Resource\":\"" + bucketArn + "\"}]}";

    PutBucketPolicyRequest reqPutBucketPolicy = new PutBucketPolicyRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withPolicy(policy);

    PutBucketPolicyResult respPutBucketPolicy =
 s3ControlClient.putBucketPolicy(reqPutBucketPolicy);
    System.out.printf("PutBucketPolicy Response: %s%n",
 respPutBucketPolicy.toString());

}
```

檢視 Amazon S3 on Outposts 儲存貯體的儲存貯體政策

儲存貯體政策是以資源為基礎的 AWS Identity and Access Management (IAM) 政策，您可以使用這些政策來將存取許可授予儲存貯體及其中物件。只有儲存貯體擁有者可建立政策與儲存貯體的關聯。連接到儲存貯體的許可會套用至儲存貯體擁有者帳戶擁有的所有儲存貯體物件。儲存貯體政策的大小限制為 20 KB。如需詳細資訊，請參閱 儲存貯體政策。

下列主題示範如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 或 AWS SDK for Java 來檢視 Amazon S3 on Outposts 儲存貯體政策。

使用 S3 主控台

建立或編輯儲存貯體政策

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇您要編輯其許可的 Outposts 儲存貯體。
4. 選擇 Permissions (許可) 標籤。
5. 在 Outposts 儲存貯體政策中，您可以檢閱現有的儲存貯體政策。如需詳細資訊，請參閱 使用 S3 on Outposts 設定 IAM。

使用 AWS CLI

下列 AWS CLI 範例取得 Outposts 儲存貯體的政策。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control get-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

使用適用於 Java 的 AWS SDK

下列適用於 Java 的開發套件範例取得 Outposts 儲存貯體的政策。

```
import com.amazonaws.services.s3control.model.*;

public void getBucketPolicy(String bucketArn) {

    GetBucketPolicyRequest reqGetBucketPolicy = new GetBucketPolicyRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    GetBucketPolicyResult respGetBucketPolicy =
 s3ControlClient.getBucketPolicy(reqGetBucketPolicy);
    System.out.printf("GetBucketPolicy Response: %s%n",
 respGetBucketPolicy.toString());

}
```

刪除 Amazon S3 on Outposts 儲存貯體的儲存貯體政策

儲存貯體政策是以資源為基礎的 AWS Identity and Access Management (IAM) 政策，您可以使用這些政策來將存取許可授予儲存貯體及其中物件。只有儲存貯體擁有者可建立政策與儲存貯體的關聯。連接到儲存貯體的許可會套用至儲存貯體擁有者帳戶擁有的所有儲存貯體物件。儲存貯體政策的大小限制為 20 KB。如需詳細資訊，請參閱 儲存貯體政策。

下列主題示範如何使用 AWS Management Console 或 AWS Command Line Interface (AWS CLI) 來檢視 Amazon S3 on Outposts 儲存貯體政策。

使用 S3 主控台

若要刪除儲存貯體政策

1. 請在 https://console.aws.amazon.com/s3/ 開啟 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇您要編輯其許可的 Outposts 儲存貯體。
4. 選擇 Permissions (許可) 標籤。
5. 在 Outposts bucket policy (Outposts 儲存貯體政策) 區段中，選擇 Delete (刪除)。
6. 確認刪除。

使用 AWS CLI

下列範例示範使用 AWS CLI 刪除 S3 on Outposts 儲存貯體 (`s3-outposts:DeleteBucket`) 的儲存貯體政策。若要執行此命令，請以您自己的資訊取代 *`user input placeholders`*。

```
aws s3control delete-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket
```

儲存貯體政策範例

使用 S3 on Outposts 儲存貯體政策，您可以在 Outposts 儲存貯體上安全地存取 S3 中的物件，以便只有具有適當許可的使用者才能存取它們。您甚至可以防止沒有適當許可的經過身份驗證的使用者在 Outposts 資源上存取 S3。

本節介紹了 Outposts 存儲桶政策上 S3 的典型使用案例的示例。若要測試這些政策，請將 *`user input placeholders`* 取代為您自己的資訊 (例如儲存貯體名稱)。

若要授予或拒絕一組物件的許可，您可以在 Amazon Resource Name (ARN) 和其他值上使用萬用字元 (*)。例如，您可以控制對以常用字首或以給定的擴展名結束，例如 `.html`。

如需 AWS Identity and Access Management 政策語言的詳細資訊，請參閱 使用 S3 on Outposts 設定 IAM。

> **Note**
>
> 使用 Amazon S3 主控台測試s3outposts許可時，您必須授與主控台所需的其他許可 `s3outposts:createendpoints3outposts:listendpoints`，例如、等。

建立儲存貯體政策的其他資源

- 如需在 Outposts 儲存貯體政策建立 S3 時可使用的 IAM 政策動作、資源和條件金鑰清單，請參閱 Outposts 上 Amazon S3 的動作、資源和條件金鑰。
- 如需在 Outposts 上建立 S3 政策的指引，請參閱新增或編輯 Amazon S3 on Outposts 儲存貯體的儲存貯體政策。

主題

- 根據特定 IP 地址在 Outposts 儲存貯體上管理對 Amazon S3 的存取

根據特定 IP 地址在 Outposts 儲存貯體上管理對 Amazon S3 的存取

儲存貯體政策是以資源為基礎的 AWS Identity and Access Management (IAM) 政策，您可以使用這些政策來將存取許可授予儲存貯體及其中物件。只有儲存貯體擁有者可建立政策與儲存貯體的關聯。連接到儲存貯體的許可會套用至儲存貯體擁有者帳戶擁有的所有儲存貯體物件。儲存貯體政策的大小限制為 20 KB。如需詳細資訊，請參閱 儲存貯體政策。

限制特定 IP 地址的存取

下列範例會拒絕所有使用者對指定儲存貯體中的物件執行任何 S3 的 Outposts 作業，除非要求來自指定的 IP 位址範圍。

> **Note**
>
> 限制對特定 IP 位址的存取時，請確定您也指定哪些 VPC 端點、VPC 來源 IP 位址或外部 IP 位址可以存取 Outposts 儲存貯體上的 S3。否則，如果您的政策拒絕所有使用者對 Outposts 儲存貯體中 S3 中的物件執行任何s3outposts操作，則可能會失去儲存貯體的存取權限。

此原則的`Condition`聲明會識別*`192.0.2.0/24`*為允許的 IP 第 4 版 (IPv4) IP 位址的範圍。

Condition 會封鎖使用 NotIpAddress 條件和 aws:SourceIp 條件金鑰，其為 AWS 通用條件金鑰。aws:SourceIp 條件鍵僅可用於公有 IP 地址範圍。如需有關這些條件金鑰的詳細資訊，請參閱 Outposts 上 S3 的動作、資源和條件金鑰。aws:SourceIp IPv4 值會使用標準 CIDR 表示法。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素參考資料。

> **Warning**
>
> 在 Outposts 政策上使用此 S3 之前，請先將此範例中的 *192.0.2.0/24*IP 位址範圍取代為適合您使用案例的適當值。否則，您將失去存取儲存貯體的能力。

```
{
    "Version": "2012-10-17",
    "Id": "S3OutpostsPolicyId1",
    "Statement": [
        {
            "Sid": "IPAllow",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3outposts:*",
            "Resource": [
                "arn:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
accesspoint/EXAMPLE-ACCESS-POINT-NAME"
                "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
bucket/DOC-EXAMPLE-BUCKET"
            ],
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "192.0.2.0/24"
                }
            }
        }
    ]
}
```

同時允許 IPv4 和 IPv6 地址

當您開始使用 IPv6 地址時，建議除了現有 IPv4 範圍之外，還使用 IPv6 地址範圍來更新組織的所有政策。這樣做有助於確保原則在您轉換到 IPv6 時繼續運作。

以下 Outposts 上的 S3 範例儲存貯體政策示範如何混合使用 IPv4 和 IPv6 位址範圍，以涵蓋組織的所有有效 IP 位址。政策範例允許存取 IP 地址範例 *192.0.2.1* 與 *2001:DB8:1234:5678::1*，並且拒絕存取地址 *203.0.113.1* 與 *2001:DB8:1234:5678:ABCD::1*。

aws:SourceIp 條件鍵僅可用於公有 IP 地址範圍。aws:SourceIp 的 IPv6 值必須為標準 CIDR 格式。針對 IPv6，我們支援使用 :: 代表 0 的範圍 (例如，2001:DB8:1234:5678::/64)。如需詳細資訊，請參閱《IAM 使用者指南》中的「IP 地址條件運算子」。

> **Warning**
>
> 在 Outposts 政策上使用此 S3 之前，請將此範例中的 IP 位址範圍取代為您使用案例的適當值。否則，您可能會失去存取儲存貯體的能力。

```
{
    "Id": "S3OutpostsPolicyId2",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowIPmix",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3outposts:*",
            "Resource": [
                "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
bucket/DOC-EXAMPLE-BUCKET",
                    "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-
ID/bucket/DOC-EXAMPLE-BUCKET/*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
                        "2001:DB8:1234:5678::/64"
                    ]
                },
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "203.0.113.0/24",
                        "2001:DB8:1234:5678:ABCD::/80"
                    ]
                }
```

```
            }
        }
    ]
}
```

列出 Amazon S3 on Outposts 儲存貯體

透過 Amazon S3 on Outposts，您可以在 AWS Outposts 上建立 S3 儲存貯體，並針對需要本機資料存取權限、本機資料處理和資料駐留的應用程式，在內部部署輕鬆存放和擷取物件。S3 on Outposts 提供一個全新的儲存類別，即 S3 Outposts (OUTPOSTS)，它使用 Amazon S3 API，目的是在您的 AWS Outposts Outposts 上的多個裝置和伺服器上以持久、備援的方式存放資料。您可以使用存取點和透過 Virtual Private Cloud (VPC) 的端點連線，與您的 Outpost 儲存貯體進行通訊。就像在 Amazon S3 儲存貯體一樣，您在 Outpost 儲存貯體上可以使用同樣的 API 和功能，包括存取政策、加密和標記。您可以透過 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。如需詳細資訊，請參閱 什麼是 Amazon S3 on Outposts？。

如需使用 S3 on Outposts 儲存貯體的詳細資訊，請參閱 使用 S3 on Outposts 儲存貯體。

下列範例示範如何使用 AWS Management Console、AWS CLI 和 AWS SDK for Java 返回 S3 on Outposts 儲存貯體清單。

使用 S3 主控台

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 在 Outposts 儲存貯體下，檢視 S3 on Outposts 儲存貯體清單。

使用 AWS CLI

下列 AWS CLI 範例會取得 Outpost 中儲存貯體的清單。若要執行此命令，請以您自己的資訊取代每個 *user input placeholder*。如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 list-regional-buckets。

```
aws s3control list-regional-buckets --account-id 123456789012 --outpost-
id op-01ac5d28a6a232904
```

使用適用於 Java 的 AWS SDK

下列適用於 Java 的開發套件範例取得 Outpost 中的儲存貯體清單。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的 ListRegionalBuckets。

```
import com.amazonaws.services.s3control.model.*;

public void listRegionalBuckets() {

    ListRegionalBucketsRequest reqListBuckets = new ListRegionalBucketsRequest()
            .withAccountId(AccountId)
            .withOutpostId(OutpostId);

    ListRegionalBucketsResult respListBuckets =
 s3ControlClient.listRegionalBuckets(reqListBuckets);
    System.out.printf("ListRegionalBuckets Response: %s%n",
 respListBuckets.toString());

}
```

使用 AWS CLI 和適用於 Java 的開發套件取得 S3 on Outposts 儲存貯體

透過 Amazon S3 on Outposts，您可以在 AWS Outposts 上建立 S3 儲存貯體，並針對需要本機資料存取權限、本機資料處理和資料駐留的應用程式，在內部部署輕鬆存放和擷取物件。S3 on Outposts 提供一個全新的儲存類別，即 S3 Outposts (0UTP0STS)，它使用 Amazon S3 API，目的是在您的 AWS Outposts Outposts 上的多個裝置和伺服器上以持久、備援的方式存放資料。您可以使用存取點和透過 Virtual Private Cloud (VPC) 的端點連線，與您的 Outpost 儲存貯體進行通訊。就像在 Amazon S3 儲存貯體一樣，您在 Outpost 儲存貯體上可以使用同樣的 API 和功能，包括存取政策、加密和標記。您可以透過 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。如需詳細資訊，請參閱 什麼是 Amazon S3 on Outposts？。

下列範例示範如何使用 AWS CLI 和 AWS SDK for Java 取得 S3 on Outposts 儲存貯體。

> **Note**
>
> 透過 AWS CLI 或 AWS SDK 使用 Amazon S3 on Outposts 使用時，您可以提供 Outposts 的存取點 ARN，以取代儲存貯體名稱。存取點 ARN 採用以下形式，其中 *region* 是 Outpost 所在區域的 AWS 區域 代碼：
> `arn:aws:s3-outposts:`*region*`:`*123456789012*`:outpost/`*op-01ac5d28a6a232904*`/accesspoint/`*example-outposts-access-point*
> 如需 S3 on Outposts ARN 的詳細資訊，請參閱 適用於 S3 on Outposts 的資源 ARN。

使用 AWS CLI

下列 S3 on Outposts 範例使用 AWS CLI 取得儲存貯體。若要執行此命令，請以您自己的資訊取代每個 *user input placeholder*。如需此命令詳細資訊，請參閱 AWS CLI 參考中的 get-bucket。

```
aws s3control get-bucket --account-id 123456789012 --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket"
```

使用適用於 Java 的 AWS SDK

下列 S3 on Outposts 範例使用適用於 Java 的開發套件，取得一個儲存貯體。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的 GetBucket。

```
import com.amazonaws.services.s3control.model.*;

public void getBucket(String bucketArn) {

    GetBucketRequest reqGetBucket = new GetBucketRequest()
            .withBucket(bucketArn)
            .withAccountId(AccountId);

    GetBucketResult respGetBucket = s3ControlClient.getBucket(reqGetBucket);
    System.out.printf("GetBucket Response: %s%n", respGetBucket.toString());

}
```

刪除 Amazon S3 on Outposts 儲存貯體

透過 Amazon S3 on Outposts，您可以在 AWS Outposts 上建立 S3 儲存貯體，並針對需要本機資料存取權限、本機資料處理和資料駐留的應用程式，在內部部署輕鬆存放和擷取物件。S3 on Outposts 提供一個全新的儲存類別，即 S3 Outposts (0UTP0STS)，它使用 Amazon S3 API，目的是在您的 AWS Outposts Outposts 上的多個裝置和伺服器上以持久、備援的方式存放資料。您可以使用存取點和透過 Virtual Private Cloud (VPC) 的端點連線，與您的 Outpost 儲存貯體進行通訊。就像在 Amazon S3 儲存貯體一樣，您在 Outpost 儲存貯體上可以使用同樣的 API 和功能，包括存取政策、加密和標記。您可以透過 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。如需詳細資訊，請參閱 什麼是 Amazon S3 on Outposts？。

如需使用 S3 on Outposts 儲存貯體的詳細資訊，請參閱 使用 S3 on Outposts 儲存貯體。

建立儲存貯體的 AWS 帳戶 擁有該儲存貯體，且是唯一可以將其刪除的帳戶。

> **Note**
>
> - Outposts 儲存貯體在刪除之前必須先清空。
>
> Amazon S3 主控台不支援 S3 on Outposts 物件動作。若要刪除 S3 on Outposts 儲存貯體中的物件，必須使用 REST API、AWS CLI 或 AWS SDK。
> - 您必須先刪除儲存貯體的所有 Outposts 存取點，才能刪除 Outposts 儲存貯體。如需詳細資訊，請參閱 刪除存取點。
> - 您無法恢復刪除後的儲存貯體。

下列範例示範如何使用 AWS Management Console 和 AWS Command Line Interface (AWS CLI) 刪除 S3 on Outposts 儲存貯體。

使用 S3 主控台

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇您要刪除的儲存貯體，然後選擇 Delete (刪除)。
4. 確認刪除。

使用 AWS CLI

下列範例示範使用 AWS CLI 刪除 S3 on Outposts 儲存貯體 (`s3-outposts:DeleteBucket`)。若要執行此命令，請以您自己的資訊取代 *`user input placeholders`*。

```
aws s3control delete-bucket --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

使用 Amazon S3 on Outposts 存取點

若要存取 Amazon S3 on Outposts 儲存貯體，您必須建立和設定存取點。

存取點針對 Amazon S3 中的共用資料集，簡化管理大規模的資料存取。存取點為連接到儲存貯體的指定網路端點，您可以使用這些端點來執行 Amazon S3 物件操作，例如 `GetObject` 和 `PutObject`。

使用 S3 on Outposts，您必須使用存取點來存取 Outposts 儲存貯體中的任何物件。存取點僅支援虛擬主機樣式定址。

> **Note**
>
> 建立 Outposts 儲存貯體的 AWS 帳戶 擁有該儲存貯體，且是唯一可以為其指派存取點的帳戶。

下列各節說明了如何建立和管理 S3 on Outposts 儲存貯體存取點。

主題

建立 S3 on Outposts 存取點

若要存取 Amazon S3 on Outposts 儲存貯體，您必須建立和設定存取點。

存取點針對 Amazon S3 中的共用資料集，簡化管理大規模的資料存取。存取點為連接到儲存貯體的指定網路端點，您可以使用這些端點來執行 Amazon S3 物件操作，例如 `GetObject` 和 `PutObject`。使用 S3 on Outposts，您必須使用存取點來存取 Outposts 儲存貯體中的任何物件。存取點僅支援虛擬主機樣式定址。

下列範例示範如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 建立 S3 on Outposts 存取點。

> **Note**
>
> 建立 Outposts 儲存貯體的 AWS 帳戶 擁有該儲存貯體，且是唯一可以為其指派存取點的帳戶。

使用 S3 主控台

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇您要為其建立 Outposts 存取點的 Outposts 儲存貯體。
4. 選擇 Outposts access points (Outposts 存取點) 索引標籤。
5. 在 Outposts access points (Outposts 存取點) 區段中，選擇 Create Outposts access point (建立 Outposts 存取點)。
6. 在 Outposts access point settings (Outposts 存取點設定) 中，輸入存取點的名稱，然後選擇存取點的 Virtual Private Cloud (VPC)。
7. 如果您想要新增存取點的政策，可以在 Outposts access point policy (Outposts 存取點政策) 區段輸入政策。

   如需詳細資訊，請參閱 使用 S3 on Outposts 設定 IAM。

使用 AWS CLI

Example

下列 AWS CLI 範例建立 Outposts 儲存貯體的存取點。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control create-access-point --account-id 123456789012
 --name example-outposts-access-point --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket" --vpc-configuration VpcId=example-vpc-12345
```

使用適用於 Java 的 AWS SDK

Example

下列適用於 Java 的開發套件範例建立 Outposts 儲存貯體的存取點。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
import com.amazonaws.services.s3control.model.*;

public String createAccessPoint(String bucketArn, String accessPointName) {
```

```
    CreateAccessPointRequest reqCreateAP = new CreateAccessPointRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withName(accessPointName)
            .withVpcConfiguration(new VpcConfiguration().withVpcId("vpc-12345"));

    CreateAccessPointResult respCreateAP =
 s3ControlClient.createAccessPoint(reqCreateAP);
    System.out.printf("CreateAccessPoint Response: %s%n", respCreateAP.toString());

    return respCreateAP.getAccessPointArn();

}
```

針對您的 S3 on Outposts 儲存貯體存取點使用儲存貯體樣式別名

使用 S3 on Outposts，您必須使用存取點來存取 Outposts 儲存貯體中的任何物件。每次建立儲存貯體的存取點時，S3 on Outposts 都會自動產生存取點別名。您可以針對任何資料平面操作使用此存取點別名，而不是存取點 Amazon Resource Name (ARN)。例如，您可以使用存取點別名來執行物件層級操作，例如 PUT、GET、LIST 等等。如需這些操作的清單，請參閱 適用於管理物件的 Amazon S3 API 操作。

下列範例顯示了名稱為 *my-access-point* 之存取點的 ARN 和存取點別名。

- 存取點 ARN – `arn:aws:s3-outposts:`*region*`:`*123456789012*`:outpost/`*op-01ac5d28a6a232904*`/accesspoint/`*my-access-point*
- 存取點別名 – *my-access-po-o01ac5d28a6a232904e8xz5w8ijx1qzlbp3i3kuse10*`--op-s3`

如需 ARN 的詳細資訊，請參閱《AWS 一般參考》中的 Amazon Resource Name (ARN)。

如需存取點別名的詳細資訊，請參閱下列主題。

主題

- 存取點別名
- 在 S3 on Outposts 物件操作中使用存取點別名
- 限制

存取點別名

存取點別名是在與 S3 on Outposts 儲存貯體相同的命名空間內建立的。當您建立存取點時，S3 on Outposts 會自動產生無法變更的存取點別名。存取點別名符合有效 S3 on Outposts 儲存貯體名稱的所有要求，並由下列部分組成：

access point name prefix-*metadata*--op-s3

> **Note**
>
> --op-s3 尾碼保留給存取點別名，因此建議不要將其用於儲存貯體或存取點名稱。如需 S3 on Outposts 儲存貯體命名規則的詳細資訊，請參閱 使用 S3 on Outposts 儲存貯體。

尋找存取點別名

下列範例向您展示如何使用 Amazon S3 主控台和 AWS CLI 尋找存取點別名。

Example ：在 Amazon S3 主控台中尋找並複製存取點別名

在主控台中建立存取點之後，您可以從 Access Points (存取點) 清單中的 Access Point alias (存取點別名) 欄取得存取點別名。

複製存取點別名

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts access points (Outposts 存取點)。
3. 若要複製存取點別名，請執行下列其中一項動作：
   - 在 Access Points (存取點) 清單中，選取存取點名稱旁邊的選項按鈕，然後選擇 Copy Access Point alias (複製存取點別名)。
   - 選擇存取點名稱。然後，在 Outposts access point overview (Outposts 存取點概觀) 下，複製存取點別名。

Example ：使用 AWS CLI 建立存取點，並在回應中尋找存取點別名

create-access-point 命令的下列 AWS CLI 範例會建立存取點，並傳回自動產生的存取點別名。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control create-access-point --bucket example-outposts-bucket --name example-
outposts-access-point --account-id 123456789012

{
    "AccessPointArn":
    "arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/
accesspoint/example-outposts-access-point",
    "Alias": "example-outp-o01ac5d28a6a232904e8xz5w8ijx1qzlbp3i3kuse10--op-s3"
}
```

Example ：使用 AWS CLI 取得存取點別名

get-access-point 命令的下列 AWS CLI 範例會傳回所指定存取點的相關資訊。此資訊包括存取點別名。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control get-access-point --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --name example-outposts-access-point --account-id 123456789012

{
    "Name": "example-outposts-access-point",
    "Bucket": "example-outposts-bucket",
    "NetworkOrigin": "Vpc",
    "VpcConfiguration": {
        "VpcId": "vpc-01234567890abcdef"
    },
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2022-09-18T17:49:15.584000+00:00",
    "Alias": "example-outp-o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3"
}
```

Example ：使用 AWS CLI 列出存取點以尋找存取點別名

list-access-points 命令的下列 AWS CLI 範例會列出所指定存取點的相關資訊。此資訊包括存取點別名。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control list-access-points --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket

{
    "AccessPointList": [
        {
            "Name": "example-outposts-access-point",
            "NetworkOrigin": "Vpc",
            "VpcConfiguration": {
                "VpcId": "vpc-01234567890abcdef"
            },
            "Bucket": "example-outposts-bucket",
            "AccessPointArn": "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point",
            "Alias": "example-outp-o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3"
        }
    ]
}
```

在 S3 on Outposts 物件操作中使用存取點別名

採用存取點時，您可以使用存取點別名，而不需要進行大量的程式碼變更。

此 AWS CLI 範例顯示了 S3 on Outposts 儲存貯體的 `get-object` 操作。此範例會使用存取點別名作為 `--bucket` 的值，而非完整存取點 ARN。

```
aws s3api get-object --bucket my-access-po-
o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3 --key  testkey  sample-object.rtf

{
    "AcceptRanges": "bytes",
    "LastModified": "2020-01-08T22:16:28+00:00",
    "ContentLength": 910,
    "ETag": "\"00751974dc146b76404bb7290f8f51bb\"",
    "VersionId": "null",
    "ContentType": "text/rtf",
    "Metadata": {}
}
```

限制

- 客戶無法設定別名。
- 存取點上的別名無法刪除、修改或停用。
- 您無法將存取點別名用於 S3 on Outposts 控制平面操作。如需 S3 on Outposts 控制平面操作清單，請參閱適用於管理儲存貯體的 Amazon S3 Control API 操作。
- AWS Identity and Access Management (IAM) 政策中無法使用別名。

檢視存取點組態的相關資訊

存取點針對 Amazon S3 中的共用資料集，簡化管理大規模的資料存取。存取點為連接到儲存貯體的指定網路端點，您可以使用這些端點來執行 Amazon S3 物件操作，例如 `GetObject` 和 `PutObject`。使用 S3 on Outposts，您必須使用存取點來存取 Outposts 儲存貯體中的任何物件。存取點僅支援虛擬主機樣式定址。

下列主題示範如何使用 AWS Management Console、AWS Command Line Interface(AWS CLI) 和 AWS SDK for Java 傳回 S3 on Outposts 存取點的組態資訊。

使用 S3 主控台

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts access points (Outposts 存取點)。
3. 選擇您要檢視組態詳細資訊的 Outposts 存取點。
4. 在 Outposts 存取點概觀下，檢閱存取點的組態詳細資訊。

使用 AWS CLI

下列 AWS CLI 範例取得 Outposts 儲存貯體的存取點。以您的資訊取代 *user input placeholders*。

```
aws s3control get-access-point --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

使用適用於 Java 的 AWS SDK

下列適用於 Java 的開發套件範例取得 Outposts 儲存貯體的存取點。

```
import com.amazonaws.services.s3control.model.*;

public void getAccessPoint(String accessPointArn) {

    GetAccessPointRequest reqGetAP = new GetAccessPointRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn);

    GetAccessPointResult respGetAP = s3ControlClient.getAccessPoint(reqGetAP);
    System.out.printf("GetAccessPoint Response: %s%n", respGetAP.toString());

}
```

檢視 Amazon S3 Outposts 存取點清單

存取點針對 Amazon S3 中的共用資料集，簡化管理大規模的資料存取。存取點為連接到儲存貯體的指定網路端點，您可以使用這些端點來執行 Amazon S3 物件操作，例如 `GetObject` 和 `PutObject`。使用 S3 on Outposts，您必須使用存取點來存取 Outposts 儲存貯體中的任何物件。存取點僅支援虛擬主機樣式定址。

下列主題示範如何使用AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 傳回 S3 Outposts 存取點清單。

使用 S3 主控台

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts access points (Outposts 存取點)。
3. 在 Outposts 存取點下，檢閱 S3 on Outposts 存取點。

使用 AWS CLI

下列 AWS CLI 範例列出了 Outposts 儲存貯體的存取點。若要執行此命令，請以您自己的資訊取代 *`user input placeholders`*。

```
aws s3control list-access-points --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

使用適用於 Java 的 AWS SDK

下列適用於 Java 的開發套件範例列出了 Outposts 儲存貯體的存取點。

```
import com.amazonaws.services.s3control.model.*;

public void listAccessPoints(String bucketArn) {

    ListAccessPointsRequest reqListAPs = new ListAccessPointsRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    ListAccessPointsResult respListAPs = s3ControlClient.listAccessPoints(reqListAPs);
    System.out.printf("ListAccessPoints Response: %s%n", respListAPs.toString());

}
```

刪除存取點

存取點針對 Amazon S3 中的共用資料集，簡化管理大規模的資料存取。存取點為連接到儲存貯體的指定網路端點，您可以使用這些端點來執行 Amazon S3 物件操作，例如 `GetObject` 和 `PutObject`。使用 S3 on Outposts，您必須使用存取點來存取 Outposts 儲存貯體中的任何物件。存取點僅支援虛擬主機樣式定址。

下列範例示範如何使用 AWS Management Console 和 AWS Command Line Interface (AWS CLI) 刪除存取點。

使用 S3 主控台

1. 請在 https://console.aws.amazon.com/s3/ 開啟 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts access points (Outposts 存取點)。
3. 在 Outposts access points (Outposts 存取點) 區段中，選擇您要刪除的 Outposts 存取點。
4. 選擇 Delete (刪除)。
5. 確認刪除。

使用 AWS CLI

下列 AWS CLI 範例刪除 Outposts 存取點。若要執行此命令，請以您自己的資訊取代 *`user input placeholders`*。

```
aws s3control delete-access-point --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

新增或編輯存取點政策

存取點有 Amazon S3 on Outposts 對於透過該存取點進行的任何請求所套用的不同許可和網路控制。每個存取點都會強制執行自訂的存取點政策，該政策可結合附加至基礎儲存貯體的儲存貯體政策運作。如需詳細資訊，請參閱 存取點。

下列主題介紹如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 新增或編輯 S3 on Outposts 存取點政策。

使用 S3 主控台

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇您要編輯存取點政策的 Outposts 儲存貯體。
4. 選擇 Outposts access points (Outposts 存取點) 索引標籤。
5. 在 Outposts access points (Outposts 存取點) 區段中，選取您要編輯其政策的存取點，然後選擇 Edit policy (編輯政策)。
6. 在 Outposts access point policy (Outposts 存取點政策) 區段中新增或編輯政策。如需詳細資訊，請參閱 使用 S3 on Outposts 設定 IAM。

使用 AWS CLI

下列 AWS CLI 範例將政策放置在 Outposts 存取點上。

1. 將以下存取點政策儲存至 JSON 檔案。在此範例中，檔案命名為 `appolicy1.json`。以您的資訊取代 *`user input placeholders`*。

```
{
   "Version":"2012-10-17",
   "Id":"exampleAccessPointPolicy",
   "Statement":[
      {
         "Sid":"st1",
         "Effect":"Allow",
         "Principal":{
```

```
            "AWS":"123456789012"
         },
         "Action":"s3-outposts:*",
         "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point
      }
   ]
}
```

2. 提交 JSON 檔案以做為 put-access-point-policy CLI 命令的一部分。以您的資訊取代 *user input placeholders*。

```
aws s3control put-access-point-policy --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --policy file://appolicy1.json
```

使用適用於 Java 的 AWS SDK

下列適用於 Java 的開發套件範例在 Outposts 存取點上放置政策。

```
import com.amazonaws.services.s3control.model.*;

public void putAccessPointPolicy(String accessPointArn) {

    String policy = "{\"Version\":\"2012-10-17\",\"Id\":\"testAccessPointPolicy\",
\"Statement\":[{\"Sid\":\"st1\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"" +
 AccountId + "\"},\"Action\":\"s3-outposts:*\",\"Resource\":\"" + accessPointArn +
 "\"}]}";

    PutAccessPointPolicyRequest reqPutAccessPointPolicy = new
 PutAccessPointPolicyRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn)
            .withPolicy(policy);

    PutAccessPointPolicyResult respPutAccessPointPolicy =
 s3ControlClient.putAccessPointPolicy(reqPutAccessPointPolicy);
    System.out.printf("PutAccessPointPolicy Response: %s%n",
 respPutAccessPointPolicy.toString());
    printWriter.printf("PutAccessPointPolicy Response: %s%n",
 respPutAccessPointPolicy.toString());
```

```
}
```

查看 S3 on Outposts 存取點政策

存取點有 Amazon S3 on Outposts 對於透過該存取點進行的任何請求所套用的不同許可和網路控制。每個存取點都會強制執行自訂的存取點政策，該政策可結合附加至基礎儲存貯體的儲存貯體政策運作。如需詳細資訊，請參閱 存取點。

如需使用 S3 on Outposts 存取點的詳細資訊，請參閱 使用 S3 on Outposts 儲存貯體。

下列主題示範如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 檢視 S3 on Outposts 存取點政策。

使用 S3 主控台

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts access points (Outposts 存取點)。
3. 選擇您要編輯存取點政策的 Outposts 存取點。
4. 在Permissions (許可)標籤上，檢閱 S3 on Outposts 存取點政策。
5. 若要編輯存取點政策，請參閱 新增或編輯存取點政策。

使用 AWS CLI

下列 AWS CLI 範例取得 Outposts 存取點的政策。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control get-access-point-policy --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

使用適用於 Java 的 AWS SDK

下列適用於 Java 的開發套件範例取得 Outposts 存取點的政策。

```
import com.amazonaws.services.s3control.model.*;

public void getAccessPointPolicy(String accessPointArn) {
```

```
    GetAccessPointPolicyRequest reqGetAccessPointPolicy = new
 GetAccessPointPolicyRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn);

    GetAccessPointPolicyResult respGetAccessPointPolicy =
 s3ControlClient.getAccessPointPolicy(reqGetAccessPointPolicy);
    System.out.printf("GetAccessPointPolicy Response: %s%n",
 respGetAccessPointPolicy.toString());
    printWriter.printf("GetAccessPointPolicy Response: %s%n",
 respGetAccessPointPolicy.toString());

}
```

使用 Amazon S3 on Outposts 端點

若要將請求路由至 Amazon S3 on Outpost 存取點，您必須建立和設定 S3 on Outposts 端點。若要建立端點，您需要與 Outposts 主要區域的服務連結有效連接。Outpost 上的每個 Virtual Private Cloud (VPC) 可以有一個相關聯的端點。如需端點配額的詳細資訊，請參閱 S3 on Outposts 網路需求。您必須建立端點，才能存取您的 Outposts 儲存貯體並執行物件操作。如需更多詳細資訊，請參閱 端點。

建立端點後，您可以使用「狀態」欄位來了解端點狀態。若您的 Outposts 處於離線狀態，系統將回傳 CREATE_FAILED。您可以檢查服務連結的連線、刪除端點，然後在連線恢復後重試建立操作。如需其他錯誤代碼列表，請參閱下方內容。如需更多詳細資訊，請參閱 端點。

API	狀態	失敗原因錯誤代碼	訊息 - 失敗原因
`CreateEndpoint`	Create_Failed	OutpostNotReachable	因為與 Outposts 主要區域的服務連結連線失敗，所以無法建立端點。請檢查您的連線狀態、刪除端點，然後再試一次。
`CreateEndpoint`	Create_Failed	InternalError	由於內部錯誤，無法建立端點。請刪除端點然後重新建立一次。
`DeleteEndpoint`	Delete_Failed	OutpostNotReachable	因為與 Outposts 主要區域的服務連結連線失敗，所以無法刪除端點。請檢查您的連線狀態，然後再試一次。

API	狀態	失敗原因錯誤代碼	訊息 - 失敗原因
DeleteEndpoint	Delete_Failed	InternalError	由於內部錯誤，無法刪除端點。請再試一次。

如需在 S3 on Outposts 上使用儲存貯體的詳細資訊，請參閱 使用 S3 on Outposts 儲存貯體。

以下幾節描述如何建立和管理 S3 on Outposts 的端點。

主題

- 在 Outpost 上建立端點
- 檢視 Amazon S3 on Outposts 端點上的清單
- 刪除 Amazon S3 on Outposts 端點

在 Outpost 上建立端點

若要將請求路由至 Amazon S3 on Outpost 存取點，您必須建立和設定 S3 on Outposts 端點。若要建立端點，您需要與 Outposts 主要區域的服務連結有效連接。Outpost 上的每個 Virtual Private Cloud (VPC) 可以有一個相關聯的端點。如需端點配額的詳細資訊，請參閱 S3 on Outposts 網路需求。您必須建立端點，才能存取您的 Outposts 儲存貯體並執行物件操作。如需更多詳細資訊，請參閱 端點。

許可

如需建立端點所需許可的詳細資訊，請參閱 適用於 S3 on Outposts 端點的許可。

當您建立端點時，S3 on Outposts 也會在您的 AWS 帳戶 中建立服務連結角色。如需更多詳細資訊，請參閱 針對 Amazon S3 on Outposts 使用服務連結角色。

以下範例示範如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 建立 S3 on Outposts 端點。

使用 S3 主控台

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts access points (Outposts 存取點)。
3. 選擇 Outposts endpoints (Outposts 端點) 索引標籤。
4. 選擇 Create Outposts endpoint (建立 Outposts 端點)。

5. 在 Outpost 下，選擇 Outpost 以建立此端點。
6. 在 VPC 下，選擇還沒有端點且符合 Ourposts 端點規則的 VPC。

   Virtual Private Cloud (VPC) 可讓您將 AWS 資源啟動到您定義的虛擬網路。這個虛擬網路與您在資料中心中操作的傳統網路非常相似，且具備使用 可擴展基礎設施的優勢AWS

   如果您沒有 VPC，請選擇 Create VPC (建立 VPC)。如需更多詳細資訊，請參閱 建立受限於 Virtual Private Cloud 的存取點。
7. 選擇 Create Outposts endpoint (建立 Outposts 端點)。

使用 AWS CLI

Example

下列 AWS CLI 範例使用 VPC 資源存取類型建立 Outpost 的端點。VPC 衍生自子網路。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1
```

下列 AWS CLI 範例使用客戶擁有的 IP 地址集區 (CoIP 集區) 存取類型為 Outpost 建立端點。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1 --access-type CustomerOwnedIp --
customer-owned-ipv4-pool ipv4pool-coip-12345678901234567
```

使用適用於 Java 的 AWS SDK

Example

下列適用於 Java 的開發套件範例建立 Outposts 的端點。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.CreateEndpointRequest;
import com.amazonaws.services.s3outposts.model.CreateEndpointResult;
```

```
public void createEndpoint() {
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    CreateEndpointRequest createEndpointRequest = new CreateEndpointRequest()
                .withOutpostId("op-0d79779cef3c30a40")
                .withSubnetId("subnet-8c7a57c5")
                .withSecurityGroupId("sg-ab19e0d1")
                .withAccessType("CustomerOwnedIp")
                .withCustomerOwnedIpv4Pool("ipv4pool-coip-12345678901234567");
    // Use .withAccessType and .withCustomerOwnedIpv4Pool only when the access type is
    // customer-owned IP address pool (CoIP pool)
    CreateEndpointResult createEndpointResult =
 s3OutpostsClient.createEndpoint(createEndpointRequest);
    System.out.println("Endpoint is created and its ARN is " +
 createEndpointResult.getEndpointArn());
}
```

檢視 Amazon S3 on Outposts 端點上的清單

若要將請求路由至 Amazon S3 on Outpost 存取點，您必須建立和設定 S3 on Outposts 端點。若要建立端點，您需要與 Outposts 主要區域的服務連結有效連接。Outpost 上的每個 Virtual Private Cloud (VPC) 可以有一個相關聯的端點。如需端點配額的詳細資訊，請參閱 S3 on Outposts 網路需求。您必須建立端點，才能存取您的 Outposts 儲存貯體並執行物件操作。如需更多詳細資訊，請參閱 端點。

下列範例示範如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 返回 S3 on Outposts 端點的清單。

使用 S3 主控台

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts access points (Outposts 存取點)。
3. 在 Outposts access points (Outposts 存取點)頁面上，選擇 Outposts endpoints (Outposts 端點)標籤。
4. 在 Outposts endpoints (Outposts 端點)中，您可以查看 S3 on Outposts 端點清單。

使用 AWS CLI

下列 AWS CLI 範例列出與您帳戶關聯的 AWS Outposts 資源其端點。如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 list-endpoints。

```
aws s3outposts list-endpoints
```

使用適用於 Java 的 AWS SDK

下列適用於 Java 的開發套件範例列出了 Outpost 的端點。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的 ListEndpoints。

```
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.ListEndpointsRequest;
import com.amazonaws.services.s3outposts.model.ListEndpointsResult;

public void listEndpoints() {
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    ListEndpointsRequest listEndpointsRequest = new ListEndpointsRequest();
    ListEndpointsResult listEndpointsResult =
 s3OutpostsClient.listEndpoints(listEndpointsRequest);
    System.out.println("List endpoints result is " + listEndpointsResult);
}
```

刪除 Amazon S3 on Outposts 端點

若要將請求路由至 Amazon S3 on Outpost 存取點，您必須建立和設定 S3 on Outposts 端點。若要建立端點，您需要與 Outposts 主要區域的服務連結有效連接。Outpost 上的每個 Virtual Private Cloud (VPC) 可以有一個相關聯的端點。如需端點配額的詳細資訊，請參閱 S3 on Outposts 網路需求。您必須建立端點，才能存取您的 Outposts 儲存貯體並執行物件操作。如需更多詳細資訊，請參閱 端點。

下列範例示範如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 刪除 S3 on Outposts 端點。

使用 S3 主控台

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts access points (Outposts 存取點)。
3. 在 Outposts access points (Outposts 存取點)頁面上，選擇 Outposts endpoints (Outposts 端點)標籤。
4. 在 Outposts 端點下，選擇您要刪除的端點，然後選擇Delete (刪除)。

使用 AWS CLI

下列 AWS CLI 範例刪除了 Outpost 的端點。若要執行此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3outposts delete-endpoint --endpoint-id example-endpoint-id --outpost-
id op-01ac5d28a6a232904
```

使用適用於 Java 的 AWS SDK

下列適用於 Java 的開發套件範例刪除 Outpost 的端點。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
import com.amazonaws.arn.Arn;
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.DeleteEndpointRequest;

public void deleteEndpoint(String endpointArnInput) {
    String outpostId = "op-01ac5d28a6a232904";
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    Arn endpointArn = Arn.fromString(endpointArnInput);
    String[] resourceParts = endpointArn.getResource().getResource().split("/");
    String endpointId = resourceParts[resourceParts.length - 1];
    DeleteEndpointRequest deleteEndpointRequest = new DeleteEndpointRequest()
                .withEndpointId(endpointId)
                .withOutpostId(outpostId);
    s3OutpostsClient.deleteEndpoint(deleteEndpointRequest);
    System.out.println("Endpoint with id " + endpointId + " is deleted.");
}
```

使用 S3 on Outposts 物件

使用 Outposts 上的 Amazon S3，您可以在 Outposts 上建立 S3 儲存貯體，並輕鬆地在現場部署存放和擷取物件，以供需要本機資料存取、本機資料處理和資料存放區的應用程式使用。 AWS Outposts 上的 S3 提供新的儲存類別 S3 Outposts (0UTP0STS)，該類別使用 Amazon S3 API，其設計用於在您的多個裝置和伺服器上持久且冗餘地存放資料。 AWS Outposts您可以使用存取點和透過 Virtual Private Cloud (VPC) 的端點連線，與您的 Outpost 儲存貯體進行通訊。就像在 Amazon S3 儲存貯體

一樣，您在 Outpost 儲存貯體上可以使用同樣的 API 和功能，包括存取政策、加密和標記。您可以透過 AWS Management Console、 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 REST API 在 Outposts 上使用 S3。

物件是存放在 Amazon S3 on Outposts 中的基本實體。每個物件都包含在儲存貯體中。您必須使用存取點來存取 Outpost 儲存貯體中的任何物件。針對物件操作指定儲存貯體時，您可以使用存取點 Amazon Resource Name (ARN) 或存取點別名。如需存取點別名的詳細資訊，請參閱 針對您的 S3 on Outposts 儲存貯體存取點使用儲存貯體樣式別名。

下列範例顯示 Outposts 存取點上 S3 的 ARN 格式，其中包括前哨所在地區的程式 AWS 區域碼、ID、前哨 AWS 帳戶 ID 和存取點名稱：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

如需 S3 on Outposts ARN 的詳細資訊，請參閱 適用於 S3 on Outposts 的資源 ARN。

物件 ARN 使用下列格式，其中包括 Outpost AWS 區域 所屬的目 AWS 帳戶 標、ID、前哨識別碼、值區名稱和物件金鑰：

```
arn:aws:s3-outposts:us-west-2:123456789012:outpost/ op-01ac5d28a6a232904/
bucket/example-s3-bucket1/object/myobject
```

對於 Amazon S3 on Outposts，物件資料始終存放在 Outpost 上。 AWS 安裝 Outpost 機架時，您的資料會保留在 Outpost 的本機，以符合資料駐留需求。您的物件永遠不會離開您的 Outpost，也不會在 AWS 區域中。由於在區域內託管，因 AWS Management Console 此您無法使用控制台上傳或管理 Outpost 中的物件。但是，您可以使用 REST API、 AWS Command Line Interface (AWS CLI) 和 AWS SDK，透過存取點上傳和管理您的物件。

主題

- 將對象上傳到 Outposts 存儲桶上的 S3
- 使用 AWS SDK for Java 在 Amazon S3 on Outposts 儲存貯體中複製物件
- 從 Amason S3 on Outposts 儲存貯體取得物件
- 列出 Amazon S3 on Outposts 儲存貯體中的物件
- 刪除 Amazon S3 on Outposts 儲存貯體中的物件
- 使用 HeadBucket 判斷 S3 on Outposts 儲存貯體是否存在，並且您是否擁有存取許可
- 使用適用於 Java 的開發套件執行和管理分段上傳

- 使用適用於 S3 on OutOutposts 的預先簽章 URL
- Outposts 上的 Amazon S3 與 Outposts 上的本地 Amazon EMR
- 授權和身份驗證緩存

將對象上傳到 Outposts 存儲桶上的 S3

物件是存放在 Amazon S3 on Outposts 中的基本實體。每個物件都包含在儲存貯體中。您必須使用存取點來存取 Outpost 儲存貯體中的任何物件。針對物件操作指定儲存貯體時，您可以使用存取點 Amazon Resource Name (ARN) 或存取點別名。如需存取點別名的詳細資訊，請參閱 針對您的 S3 on Outposts 儲存貯體存取點使用儲存貯體樣式別名。

下列範例顯示 Outposts 存取點上 S3 的 ARN 格式，其中包括前哨所在地區的程式 AWS 區域碼、ID、前哨 AWS 帳戶 ID 和存取點名稱：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

如需 S3 on Outposts ARN 的詳細資訊，請參閱 適用於 S3 on Outposts 的資源 ARN。

對於 Amazon S3 on Outposts，物件資料始終存放在 Outpost 上。 AWS 安裝 Outpost 機架時，您的資料會保留在 Outpost 的本機，以符合資料駐留需求。您的物件永遠不會離開您的 Outpost，也不會在 AWS 區域中。由於在區域內託管，因 AWS Management Console 此您無法使用控制台上傳或管理 Outpost 中的物件。但是，您可以使用 REST API、 AWS Command Line Interface (AWS CLI) 和 AWS SDK，透過存取點上傳和管理物件。

以下 AWS CLI 和 AWS SDK for Java 範例說明如何使用存取點將物件上傳至 Outposts 儲存貯體上的 S3。

AWS CLI

Example

下列範例使用 AWS CLI將名稱為 `sample-object.xml` 的物件放置在 S3 on Outposts 儲存貯體 (`s3-outposts:PutObject`) 上。若要執行此命令，請以您自己的資訊取代每個 *`user input placeholder`*。如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 put-object。

```
aws s3api put-object --bucket arn:aws:s3-
outposts:Region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --key sample-object.xml --body sample-object.xml
```

SDK for Java

Example

下列範例使用適用於 Java 的開發套件，將物件放置在 S3 on Outposts 儲存貯體。若要使用此範例，請以您自己的資訊取代每個 *user input placeholder*。如需更多詳細資訊，請參閱 上傳物件。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;

import java.io.File;

public class PutObject {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String stringObjKeyName = "*** String object key name ***";
        String fileObjKeyName = "*** File object key name ***";
        String fileName = "*** Path to file to upload ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Upload a text string as a new object.
            s3Client.putObject(accessPointArn, stringObjKeyName, "Uploaded String
 Object");

            // Upload a file as a new object with ContentType and title specified.
            PutObjectRequest request = new PutObjectRequest(accessPointArn,
 fileObjKeyName, new File(fileName));
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentType("plain/text");
            metadata.addUserMetadata("title", "someTitle");
            request.setMetadata(metadata);
```

```
            s3Client.putObject(request);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用 AWS SDK for Java 在 Amazon S3 on Outposts 儲存貯體中複製物件

物件是存放在 Amazon S3 on Outposts 中的基本實體。每個物件都包含在儲存貯體中。您必須使用存取點來存取 Outpost 儲存貯體中的任何物件。針對物件操作指定儲存貯體時，您可以使用存取點 Amazon Resource Name (ARN) 或存取點別名。如需存取點別名的詳細資訊，請參閱 針對您的 S3 on Outposts 儲存貯體存取點使用儲存貯體樣式別名。

以下範例顯示了 S3 on Outposts 存取點的 ARN 格式，其中包括 Outpost 所在區域的 AWS 區域代碼、AWS 帳戶 ID、Outposts ID，以及存取點名稱：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

如需 S3 on Outposts ARN 的詳細資訊，請參閱 適用於 S3 on Outposts 的資源 ARN。

對於 Amazon S3 on Outposts，物件資料始終存放在 Outpost 上。當 AWS 安裝 Outpost 機架時，您的資料將保留在本機 Outpost 上，以滿足資料駐留的要求。您的物件永遠不會離開您的 Outpost，也不會在 AWS 區域 中。由於 AWS Management Console 託管在區域內，您無法使用主控台上傳或管理 Outpost 中的物件。然而，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 以及 AWS SDK 透過存取點上傳和管理您的物件。

下列範例示範如何使用 AWS SDK for Java 複製 S3 on Outposts 儲存貯體中的物件。

使用適用於 Java 的 AWS SDK

下列 S3 on Outposts 範例使用適用於 Java 的開發套件，將物件複製到同一儲存貯體中的新物件。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;

public class CopyObject {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String sourceKey = "*** Source object key ***";
        String destinationKey = "*** Destination object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Copy the object into a new object in the same bucket.
            CopyObjectRequest copyObjectRequest = new CopyObjectRequest(accessPointArn,
 sourceKey, accessPointArn, destinationKey);
            s3Client.copyObject(copyObjectRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

從 Amason S3 on Outposts 儲存貯體取得物件

物件是存放在 Amazon S3 on Outposts 中的基本實體。每個物件都包含在儲存貯體中。您必須使用存取點來存取 Outpost 儲存貯體中的任何物件。針對物件操作指定儲存貯體時，您可以使用存取點 Amazon Resource Name (ARN) 或存取點別名。如需存取點別名的詳細資訊，請參閱 針對您的 S3 on Outposts 儲存貯體存取點使用儲存貯體樣式別名。

以下範例顯示了 S3 on Outposts 存取點的 ARN 格式，其中包括 Outpost 所在區域的 AWS 區域代碼、AWS 帳戶 ID、Outposts ID，以及存取點名稱：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

如需 S3 on Outposts ARN 的詳細資訊，請參閱 適用於 S3 on Outposts 的資源 ARN。

對於 Amazon S3 on Outposts，物件資料始終存放在 Outpost 上。當 AWS 安裝 Outpost 機架時，您的資料將保留在本機 Outpost 上，以滿足資料駐留的要求。您的物件永遠不會離開您的 Outpost，也不會在 AWS 區域 中。由於 AWS Management Console 託管在區域內，您無法使用主控台上傳或管理 Outpost 中的物件。然而，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 以及 AWS SDK 透過存取點上傳和管理您的物件。

下列範例示範如何使用 AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 下載 (取得) 物件。

使用 AWS CLI

下列範例使用 AWS CLI 從 S3 on Outposts 儲存貯體 (`s3-outposts:GetObject`) 取得名稱為 `sample-object.xml` 的物件。若要執行此命令，請以您自己的資訊取代每個 *`user input placeholder`*。如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 get-object。

```
aws s3api get-object --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point --key testkey sample-object.xml
```

使用適用於 Java 的 AWS SDK

下列 S3 on Outposts 範例使用適用於 Java 的開發套件取得物件。若要使用此範例，請以您自己的資訊取代每個 *`user input placeholder`*。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 GetObject。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.ResponseHeaderOverrides;
import com.amazonaws.services.s3.model.S3Object;

import java.io.BufferedReader;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class GetObject {
    public static void main(String[] args) throws IOException {
        String accessPointArn = "*** access point ARN ***";
        String key = "*** Object key ***";

        S3Object fullObject = null, objectPortion = null, headerOverrideObject = null;
        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Get an object and print its contents.
            System.out.println("Downloading an object");
            fullObject = s3Client.getObject(new GetObjectRequest(accessPointArn, key));
            System.out.println("Content-Type: " +
 fullObject.getObjectMetadata().getContentType());
            System.out.println("Content: ");
            displayTextInputStream(fullObject.getObjectContent());

            // Get a range of bytes from an object and print the bytes.
            GetObjectRequest rangeObjectRequest = new GetObjectRequest(accessPointArn,
 key)
                    .withRange(0, 9);
            objectPortion = s3Client.getObject(rangeObjectRequest);
            System.out.println("Printing bytes retrieved.");
            displayTextInputStream(objectPortion.getObjectContent());

            // Get an entire object, overriding the specified response headers, and
 print the object's content.
            ResponseHeaderOverrides headerOverrides = new ResponseHeaderOverrides()
                    .withCacheControl("No-cache")
                    .withContentDisposition("attachment; filename=example.txt");
            GetObjectRequest getObjectRequestHeaderOverride = new
 GetObjectRequest(accessPointArn, key)
                    .withResponseHeaders(headerOverrides);
            headerOverrideObject = s3Client.getObject(getObjectRequestHeaderOverride);
            displayTextInputStream(headerOverrideObject.getObjectContent());
```

```
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        } finally {
            // To ensure that the network connection doesn't remain open, close any
 open input streams.
            if (fullObject != null) {
                fullObject.close();
            }
            if (objectPortion != null) {
                objectPortion.close();
            }
            if (headerOverrideObject != null) {
                headerOverrideObject.close();
            }
        }
    }

    private static void displayTextInputStream(InputStream input) throws IOException {
        // Read the text input stream one line at a time and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        String line = null;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
}
```

列出 Amazon S3 on Outposts 儲存貯體中的物件

物件是存放在 Amazon S3 on Outposts 中的基本實體。每個物件都包含在儲存貯體中。您必須使用存取點來存取 Outpost 儲存貯體中的任何物件。針對物件操作指定儲存貯體時，您可以使用存取點 Amazon Resource Name (ARN) 或存取點別名。如需存取點別名的詳細資訊，請參閱 [針對您的 S3 on Outposts 儲存貯體存取點使用儲存貯體樣式別名]。

以下範例顯示了 S3 on Outposts 存取點的 ARN 格式，其中包括 Outpost 所在區域的 AWS 區域代碼、AWS 帳戶 ID、Outposts ID，以及存取點名稱：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

如需 S3 on Outposts ARN 的詳細資訊，請參閱 適用於 S3 on Outposts 的資源 ARN。

> **Note**
>
> 對於 Amazon S3 on Outposts，物件資料始終存放在 Outpost 上。當 AWS 安裝 Outpost 機架時，您的資料將保留在本機 Outpost 上，以滿足資料駐留的要求。您的物件永遠不會離開您的 Outpost，也不會在 AWS 區域 中。由於 AWS Management Console 託管在區域內，您無法使用主控台上傳或管理 Outpost 中的物件。然而，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 以及 AWS SDK 透過存取點上傳和管理您的物件。

下列範例示範如何使用 AWS CLI 和 AWS SDK for Java 列出 S3 on Outposts 儲存貯體中的物件。

使用 AWS CLI

下列範例使用 AWS CLI 列出 S3 on Outposts 儲存貯體 (`s3-outposts:ListObjectsV2`) 中的物件。若要執行此命令，請以您自己的資訊取代每個 *`user input placeholder`*。如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 list-objects-v2。

```
aws s3api list-objects-v2 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

> **Note**
>
> 透過 AWS SDK 將此動作與 Amazon S3 on Outposts 搭配使用時，您可以提供 Outposts 存取點 ARN 取代儲存貯體名稱，格式如下：`arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-Outposts-Access-Point`。如需 S3 on Outposts ARN 的詳細資訊，請參閱 適用於 S3 on Outposts 的資源 ARN。

使用適用於 Java 的 AWS SDK

下列 S3 on Outposts 範例使用適用於 Java 的開發套件，在儲存貯體中列出物件。若要使用此範例，請以您自己的資訊取代每個 *`user input placeholder`*。

> **Important**
>
> 此範例使用 ListObjectsV2，這是 ListObjects API 操作的最新修訂版。建議您使用此修訂版本後的 API 操作進行應用程式進行開發。為了回溯相容性，Amazon S3 會繼續支援此 API 操作的舊版本。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsV2Request;
import com.amazonaws.services.s3.model.ListObjectsV2Result;
import com.amazonaws.services.s3.model.S3ObjectSummary;

public class ListObjectsV2 {

    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            System.out.println("Listing objects");

            // maxKeys is set to 2 to demonstrate the use of
            // ListObjectsV2Result.getNextContinuationToken()
            ListObjectsV2Request req = new
 ListObjectsV2Request().withBucketName(accessPointArn).withMaxKeys(2);
            ListObjectsV2Result result;

            do {
                result = s3Client.listObjectsV2(req);

                for (S3ObjectSummary objectSummary : result.getObjectSummaries()) {
                    System.out.printf(" - %s (size: %d)\n", objectSummary.getKey(),
 objectSummary.getSize());
```

```
                }
                // If there are more than maxKeys keys in the bucket, get a
 continuation token
                // and list the next objects.
                String token = result.getNextContinuationToken();
                System.out.println("Next Continuation Token: " + token);
                req.setContinuationToken(token);
            } while (result.isTruncated());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

刪除 Amazon S3 on Outposts 儲存貯體中的物件

物件是存放在 Amazon S3 on Outposts 中的基本實體。每個物件都包含在儲存貯體中。您必須使用存取點來存取 Outpost 儲存貯體中的任何物件。針對物件操作指定儲存貯體時，您可以使用存取點 Amazon Resource Name (ARN) 或存取點別名。如需存取點別名的詳細資訊，請參閱 針對您的 S3 on Outposts 儲存貯體存取點使用儲存貯體樣式別名。

以下範例顯示了 S3 on Outposts 存取點的 ARN 格式，其中包括 Outpost 所在區域的 AWS 區域代碼、AWS 帳戶 ID、Outposts ID，以及存取點名稱：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

如需 S3 on Outposts ARN 的詳細資訊，請參閱 適用於 S3 on Outposts 的資源 ARN。

對於 Amazon S3 on Outposts，物件資料始終存放在 Outpost 上。當 AWS 安裝 Outpost 機架時，您的資料將保留在本機 Outpost 上，以滿足資料駐留的要求。您的物件永遠不會離開您的 Outpost，也不會在 AWS 區域 中。由於 AWS Management Console 託管在區域內，您無法使用主控台上傳或管理 Outpost 中的物件。然而，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 以及 AWS SDK 透過存取點上傳和管理您的物件。

下列範例示範如何透過使用 AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 來刪除 S3 on Outposts 儲存貯體中的單個物件或多個物件。

使用 AWS CLI

下列範例示範如何從 S3 on Outposts 儲存貯體中刪除單個物件或多個物件。

delete-object

下列範例使用 AWS CLI 從 S3 on Outposts 儲存貯體 (`s3-outposts:DeleteObject`) 刪除名稱為 `sample-object.xml` 的物件。若要執行此命令，請以您自己的資訊取代每個 *`user input placeholder`*。如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 delete-object。

```
aws s3api delete-object --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --key sample-object.xml
```

delete-objects

下列範例使用 AWS CLI 從 S3 on Outposts 儲存貯體 (`s3-outposts:DeleteObject`) 刪除名稱為 `sample-object.xml` 和 `test1.text` 的兩個物件。若要執行此命令，請以您自己的資訊取代每個 *`user input placeholder`*。如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 delete-objects。

```
aws s3api delete-objects --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --delete file://delete.json

delete.json
{
  "Objects": [
    {
      "Key": "test1.txt"
    },
    {
      "Key": "sample-object.xml"
    }
  ],
  "Quiet": false
```

```
}
```

使用適用於 Java 的 AWS SDK

下列範例示範如何從 S3 on Outposts 儲存貯體中刪除單個物件或多個物件。

DeleteObject

下列 S3 on Outposts 範例使用適用於 Java 的開發套件，在儲存貯體中刪除物件。若要使用此範例，請為 Outpost 指定存取點 ARN，並為您要刪除的物件指定金鑰名稱。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的DeleteObject。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectRequest;

public class DeleteObject {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String keyName = "*** key name ****";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            s3Client.deleteObject(new DeleteObjectRequest(accessPointArn, keyName));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
```

```
}
```

DeleteObjects

下列 S3 on Outposts 範例使用適用於 Java 的開發套件，在儲存貯體中上傳，然後刪除物件。若要使用此範例，請為 Outpost 指定存取點 ARN。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的 DeleteObjects。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectsRequest;
import com.amazonaws.services.s3.model.DeleteObjectsRequest.KeyVersion;
import com.amazonaws.services.s3.model.DeleteObjectsResult;

import java.util.ArrayList;

public class DeleteObjects {

    public static void main(String[] args) {
       String accessPointArn = "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point";


        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Upload three sample objects.
            ArrayList<KeyVersion> keys = new ArrayList<KeyVersion>();
            for (int i = 0; i < 3; i++) {
                String keyName = "delete object example " + i;
                s3Client.putObject(accessPointArn, keyName, "Object number " + i + "
 to be deleted.");
                keys.add(new KeyVersion(keyName));
            }
            System.out.println(keys.size() + " objects successfully created.");
```

```
            // Delete the sample objects.
            DeleteObjectsRequest multiObjectDeleteRequest = new
 DeleteObjectsRequest(accessPointArn)
                    .withKeys(keys)
                    .withQuiet(false);

            // Verify that the objects were deleted successfully.
            DeleteObjectsResult delObjRes =
 s3Client.deleteObjects(multiObjectDeleteRequest);
            int successfulDeletes = delObjRes.getDeletedObjects().size();
            System.out.println(successfulDeletes + " objects successfully
 deleted.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用 HeadBucket 判斷 S3 on Outposts 儲存貯體是否存在，並且您是否擁有存取許可

物件是存放在 Amazon S3 on Outposts 中的基本實體。每個物件都包含在儲存貯體中。您必須使用存取點來存取 Outpost 儲存貯體中的任何物件。針對物件操作指定儲存貯體時，您可以使用存取點 Amazon Resource Name (ARN) 或存取點別名。如需存取點別名的詳細資訊，請參閱 針對您的 S3 on Outposts 儲存貯體存取點使用儲存貯體樣式別名。

以下範例顯示了 S3 on Outposts 存取點的 ARN 格式，其中包括 Outpost 所在區域的 AWS 區域代碼、AWS 帳戶 ID、Outposts ID，以及存取點名稱：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

如需 S3 on Outposts ARN 的詳細資訊，請參閱 適用於 S3 on Outposts 的資源 ARN。

> **Note**
>
> 對於 Amazon S3 on Outposts，物件資料始終存放在 Outpost 上。當 AWS 安裝 Outpost 機架時，您的資料將保留在本機 Outpost 上，以滿足資料駐留的要求。您的物件永遠不會離開您的 Outpost，也不會在 AWS 區域 中。由於 AWS Management Console 託管在區域內，您無法使用主控台上傳或管理 Outpost 中的物件。然而，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 以及 AWS SDK 透過存取點上傳和管理您的物件。

下列 AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 範例顯示如何使用 HeadBucket API 操作來判斷 Amazon S3 on Outposts 儲存貯體是否存在，以及您是否有存取許可。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的 HeadBucket。

使用 AWS CLI

下列 S3 on Outposts AWS CLI 範例使用 `head-bucket` 命令判斷儲存貯體是否存在，以及您是否擁有存取許可。若要執行此命令，請以您自己的資訊取代每個 *`user input placeholder`*。如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 head-bucket。

```
aws s3api head-bucket --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

使用適用於 Java 的 AWS SDK

下列 S3 on Outposts 範例顯示如何判斷儲存貯體是否存在，以及您是否有存取許可。若要使用此範例，請為 Outpost 指定存取點 ARN。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的 HeadBucket。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.HeadBucketRequest;

public class HeadBucket {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";

        try {
```

```
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            s3Client.headBucket(new HeadBucketRequest(accessPointArn));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用適用於 Java 的開發套件執行和管理分段上傳

透過 Amazon S3 on Outposts，您可以在 AWS Outposts 資源上建立 S3 儲存貯體，並針對需要本機資料存取權限、本機資料處理和資料駐留的應用程式，在內部部署存放和擷取物件。您可以透過 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。如需詳細資訊，請參閱 什麼是 Amazon S3 on Outposts？。

下列範例示範如何使用 S3 on Outposts 搭配 AWS SDK for Java 執行和管理分段上傳。

主題

- 在 S3 on Outposts 儲存貯體中執行物件的分段上傳
- 使用分段上傳在 S3 on Outposts 儲存貯體中複製大型物件
- 列出 S3 on Outposts 儲存貯體中物件的片段
- 擷取 S3 on Outposts 儲存貯體中進行中的分段上傳清單

在 S3 on Outposts 儲存貯體中執行物件的分段上傳

下列 S3 on Outposts 範例使用適用於 Java 的開發套件，在 Outposts 儲存貯體啟動、上傳和完成物件的分段上傳。若要使用此範例，請以您自己的資訊取代每個 *`user input placeholder`*。如需詳細資訊，請參閱 使用分段上傳來上傳物件。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.ArrayList;
import java.util.List;

public class MultipartUploadCopy {
    public static void main(String[] args) {
        String accessPointArn = "*** Source access point ARN ***";
        String sourceObjectKey = "*** Source object key ***";
        String destObjectKey = "*** Target object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(accessPointArn, destObjectKey);
            InitiateMultipartUploadResult initResult =
 s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest(accessPointArn, sourceObjectKey);
            ObjectMetadata metadataResult =
 s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();

            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make sure
```

```
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(accessPointArn)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(accessPointArn)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and make
the copied object available.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
                    accessPointArn,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
```

```
        return etags;
    }
```

使用分段上傳在 S3 on Outposts 儲存貯體中複製大型物件

下列 S3 on Outposts 範例使用適用於 Java 的開發套件在儲存貯體中複製物件。若要使用此範例，請以您自己的資訊取代每個 *user input placeholder*。這個範例改編自 使用分段上傳來複製物件。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.ArrayList;
import java.util.List;

public class MultipartUploadCopy {
    public static void main(String[] args) {
        String accessPointArn = "*** Source access point ARN ***";
        String sourceObjectKey = "*** Source object key ***";
        String destObjectKey = "*** Target object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(accessPointArn, destObjectKey);
            InitiateMultipartUploadResult initResult =
 s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest(accessPointArn, sourceObjectKey);
            ObjectMetadata metadataResult =
 s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();
```

```
            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make sure
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(accessPointArn)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(accessPointArn)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and make
the copied object available.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
                    accessPointArn,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
```

```
    }

    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
        return etags;
    }
}
```

列出 S3 on Outposts 儲存貯體中物件的片段

下列 S3 on Outposts 範例使用適用於 Java 的開發套件，在儲存貯體中列出物件的片段。若要使用此範例，請以您自己的資訊取代每個 *user input placeholder*。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.List;

public class ListParts {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String keyName = "*** Key name ***";
        String uploadId = "*** Upload ID ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            ListPartsRequest listPartsRequest = new ListPartsRequest(accessPointArn,
 keyName, uploadId);
            PartListing partListing = s3Client.listParts(listPartsRequest);
            List<PartSummary> partSummaries = partListing.getParts();
```

```
            System.out.println(partSummaries.size() + " multipart upload parts");
            for (PartSummary p : partSummaries) {
                System.out.println("Upload part: Part number = \"" + p.getPartNumber()
 + "\", ETag = " + p.getETag());
            }

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

擷取 S3 on Outposts 儲存貯體中進行中的分段上傳清單

下列 S3 on Outposts 範例顯示如何使用適用於 Java 的開發套件，從 Outposts 儲存貯體擷取進行中的分段上傳清單。若要使用此範例，請以您自己的資訊取代每個 *user input placeholder*。以下是從 Amazon S3 的 列出分段上傳 範例修改的範例。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListMultipartUploadsRequest;
import com.amazonaws.services.s3.model.MultipartUpload;
import com.amazonaws.services.s3.model.MultipartUploadListing;

import java.util.List;

public class ListMultipartUploads {
    public static void main(String[] args) {
                String accessPointArn = "*** access point ARN ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
```

```
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Retrieve a list of all in-progress multipart uploads.
            ListMultipartUploadsRequest allMultipartUploadsRequest = new
 ListMultipartUploadsRequest(accessPointArn);
            MultipartUploadListing multipartUploadListing =
 s3Client.listMultipartUploads(allMultipartUploadsRequest);
            List<MultipartUpload> uploads =
 multipartUploadListing.getMultipartUploads();

            // Display information about all in-progress multipart uploads.
            System.out.println(uploads.size() + " multipart upload(s) in progress.");
            for (MultipartUpload u : uploads) {
                System.out.println("Upload in progress: Key = \"" + u.getKey() + "\",
 id = " + u.getUploadId());
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用適用於 S3 on OutOutposts 的預先簽章 URL

若要授予對存放在本機 Outpost 上物件的有限時間存取權限，而不會更新儲存貯體政策，您可以使用預先簽章 URL。使用預先簽章 URL，身為儲存貯體擁有者的您可以與虛擬私有雲端 (VPC) 中的個人共享物件，或授予他們上傳或刪除物件的能力。

當您使用 AWS SDK 或 AWS Command Line Interface (AWS CLI) 建立預先簽章 URL 時，會建立 URL 與特定動作的關聯。您也可以選擇自訂到期時間 (最低 1 秒，最高 7 天) 來授予預先簽章 URL 有限時間的存取權。當您共用預先簽章 URL 時，VPC 中的個人可以執行內嵌在 URL 中的動作，如同原始簽章使用者一樣。當 URL 到達到期時間時，該 URL 就會過期且再也無法運作。

限制預先簽章的 URL 功能

預先簽章的 URL 的功能，受到建立它的使用者許可所限制。實質上，預先簽章的 URL 是一種承載符記，可為擁有這些網址的客戶授與存取權。因此，我們建議您妥善保護它們。

AWS 第 4 版簽署程序 (SigV4)

若要在使用 AWS 第 4 版簽署程序 (SigV4) 驗證預先簽章 URL 請求時強制執行特定行為，您可以在儲存貯體政策和存取點政策中使用條件金鑰。例如，您可以建立儲存貯體政策，使用 `s3-outposts:signatureAge` 條件來拒絕任何 `example-outpost-bucket` 儲存貯體中物件上的 Amazon S3 on Outposts 預先簽章 URL 請求 (如果簽章超過 10 分鐘)。若要使用此範例，請以您自己的資訊取代 *`user input placeholders`*。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Deny a presigned URL request if the signature is more than 10
 minutes old",
            "Effect": "Deny",
            "Principal": {"AWS":"444455556666"},
            "Action": "s3-outposts:*",
            "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
            "Condition": {
                "NumericGreaterThan": {"s3-outposts:signatureAge": 600000},
                "StringEquals": {"s3-outposts:authType": "REST-QUERY-STRING"}
            }
        }
    ]
}
```

如需取得可用來強制執行特定行為 (在使用第 4 版簽署程序驗證預先簽章的 URL 請求時) 的條件金鑰和其他政策範例清單，請參閱 AWS 第 4 版簽署程序 (SigV4) 驗證特定的政策索引鍵。

網路路徑限制

如果您想要限制使用預先簽章 URL 和對特定網路路徑的所有 S3 on Outposts 存取權，您可以撰寫需要特定網路路徑的政策。若要針對進行呼叫的 IAM 主體設定限制，您可以使用身分型 AWS Identity and Access Management (IAM) 政策 (例如使用者、群組或角色政策)。若要在 S3 on Outposts 資源上設定的限制，您可以使以資源型政策 (例如儲存貯體和存取點政策)。

IAM 主體的網路路徑限制需要這些憑證的使用者從指定的網路發出請求。儲存貯體或存取點上的限制要求所有對該資源的請求都來自指定網路。這些限制也適用於預先簽章的 URL 案例之外。

您使用的 IAM 全域條件取決於端點類型。如果您正在使用 S3 on Outposts 的公有端點，請使用 `aws:SourceIp`。如果您正在使用 S3 on Outposts 的 VPC 端點，請使用 `aws:SourceVpc` 或 `aws:SourceVpce`。

下列 IAM 政策聲明要求委託人只能從指定的網路範圍存取 AWS。由於此政策聲明，所有存取均必須源自該範圍。這包含某人使用 S3 on Outposts 預先簽章 URL 的情況。若要使用此範例，請以您自己的資訊取代 *`user input placeholders`*。

```
{
    "Sid": "NetworkRestrictionForIAMPrincipal",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "NotIpAddressIfExists": {"aws:SourceIp": "IP-address-range"},
        "BoolIfExists": {"aws:ViaAWSService": "false"}
    }
}
```

如需使用 `aws:SourceIP` AWS 全域條件金鑰來將 S3 on Outposts 儲存貯體之存取權限限制在特定網路範圍內的儲存貯體政策範例，請參閱 使用 S3 on Outposts 設定 IAM。

誰可以建立預先簽章的 URL

任何具備有效安全憑證的使用者，均可建立預先簽章的 URL。但為了讓 VPC 中的使用者能順利存取物件，預先簽章的 URL 必須由有權執行預先簽章的 URL 做為基礎之操作的人員來建立。

您可以使用下列憑證來建立預先簽章 URL：

- IAM 執行個體設定檔 - 有效期限最長 6 小時。
- AWS Security Token Service - 以 AWS 帳戶 根使用者或 IAM 使用者憑證等永久憑證簽章時，有效期限最長 36 小時。
- IAM 使用者 - 使用 AWS 第 4 版簽署程序時，有效期限最長 7 天。

  若要建立有效期限最長 7 天的預先簽章 URL，請先將 IAM 使用者憑證 (存取金鑰和私密金鑰) 委派給您在使用的 SDK。然後，使用 AWS 第 4 版簽署程序來產生預先簽章的 URL。

> **Note**
>
> - 如果使用暫時字符建立了預先簽章的 URL，則 URL 會在字符過期時過期，即使您使用較晚的過期時間建立 URL 亦然。
> - 由於預先簽章 URL 會將 S3 on Outposts 儲存貯體的存取權授予擁有 URL 的任何人，因此我們建議您妥善保護這些 URL。如需保護預先簽章的 URL 的詳細資訊，請參閱限制預先簽章的 URL 功能。

S3 on Outposts 何時檢查預先簽章的 URL 中的到期日期和時間？

S3 on Outposts 會在發出 HTTP 請求時，檢查已簽署 URL 的到期日期和時間。例如，如果用戶端在到期前一刻才開始下載大型檔案，則即使在下載期間過期了，下載也會繼續。然而，如果連線中斷並且用戶端在到期時間過後嘗試重新啟動下載，則下載會失敗。

如需使用預先簽章 URL 來共享或上傳物件的詳細資訊，請參閱下列主題。

主題

- 使用預先簽章的 URL 來共用物件
- 產生預先簽章 URL 以將物件上傳至 S3 on Outposts 儲存貯體

使用預先簽章的 URL 來共用物件

若要授予對存放在本機 Outpost 上物件的有限時間存取權限，而不會更新儲存貯體政策，您可以使用預先簽章 URL。使用預先簽章 URL，身為儲存貯體擁有者的您可以與虛擬私有雲端 (VPC) 中的個人共享物件，或授予他們上傳或刪除物件的能力。

當您使用 AWS SDK 或 AWS Command Line Interface (AWS CLI) 建立預先簽署的 URL 時，您可以將 URL 與特定動作建立關聯。您也可以選擇自訂到期時間 (最低 1 秒，最高 7 天) 來授予預先簽章 URL 有限時間的存取權。當您共用預先簽章 URL 時，VPC 中的個人可以執行內嵌在 URL 中的動作，如同原始簽章使用者一樣。當 URL 到達到期時間時，該 URL 就會過期且再也無法運作。

當您建立預先簽章 URL 時，必須提供安全憑證，然後指定下列項目：

- 適用於 Amazon S3 on Outposts 儲存貯體的存取點 Amazon Resource Name (ARN)
- 物件索引鍵
- HTTP 方法 (GET 用於下載物件)

- 過期日期和時間

預先簽章 URL 僅在指定的期間內有效。也就是說，您必須在到期日期和時間之前開始 URL 所允許的操作。您可以多次使用預先簽章 URL，直到到期日期和時間為止。如果使用暫時字符建立了預先簽章的 URL，那麼 URL 會在字符過期時過期，即使您使用較晚的過期時間建立 URL 亦然。

虛擬私有雲端 (VPC) 中可存取預先簽章 URL 的使用者可以存取物件。例如，若儲存貯體中有一段影片且儲存貯體與物件皆為私有，即可透過產生預先簽章的 URL 來與其他人分享這段影片。由於預先簽章 URL 會將 S3 on Outposts 儲存貯體的存取權授予擁有 URL 的任何人，因此我們建議您妥善保護這些 URL。如需有關保護預先簽署 URL 的詳細資訊，請參閱限制預先簽章的 URL 功能。

任何具備有效安全憑證的使用者，均可建立預先簽章的 URL。然而，只有具備許可執行作為預先簽章 URL 基礎操作的人員，才能建立預先簽章 URL。如需詳細資訊，請參閱 誰可以建立預先簽章的 URL。

您可以使用 AWS SDK 與 AWS CLI來產生預先簽章 URL，以在 S3 on Outposts 儲存貯體中共享物件。如需詳細資訊，請參閱下列範例。

使用 AWS 軟體開發套件

您可以使用 AWS SDK 產生可提供給其他人的預先簽署 URL，以便他們擷取物件。

> **Note**
>
> 當您使用 AWS SDK 產生預先簽署的 URL 時，預先簽署 URL 的到期時間上限為建立後 7 天。

Java

Example

以下範例會產生預先簽章的 URL，您可以將其提供給其他人，讓他們可以從 S3 on Outposts 儲存貯體擷取物件。如需詳細資訊，請參閱 使用適用於 S3 on OutOutposts 的預先簽章 URL。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

如需建立和測試工作範例的指示，請參閱開 AWS SDK for Java 發人員指南中的入門指南。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.HttpMethod;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
```

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GeneratePresignedUrlRequest;

import java.io.IOException;
import java.net.URL;
import java.time.Instant;

public class GeneratePresignedURL {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String accessPointArn = "*** access point ARN ***";
        String objectKey = "*** object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Set the presigned URL to expire after one hour.
            java.util.Date expiration = new java.util.Date();
            long expTimeMillis = Instant.now().toEpochMilli();
            expTimeMillis += 1000 * 60 * 60;
            expiration.setTime(expTimeMillis);

            // Generate the presigned URL.
            System.out.println("Generating pre-signed URL.");
            GeneratePresignedUrlRequest generatePresignedUrlRequest =
                    new GeneratePresignedUrlRequest(accessPointArn, objectKey)
                            .withMethod(HttpMethod.GET)
                            .withExpiration(expiration);
            URL url = s3Client.generatePresignedUrl(generatePresignedUrlRequest);

            System.out.println("Pre-Signed URL: " + url.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't
 process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
```

```
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

Example

以下範例會產生預先簽章的 URL，您可以將其提供給其他人，讓他們可以從 S3 on Outposts 儲存貯體擷取物件。如需詳細資訊，請參閱 使用適用於 S3 on OutOutposts 的預先簽章 URL。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

如需有關設定和執行程式碼範例的詳細資訊，請參閱 .NET 開發人員指南中的AWS SDK for .NET 入門。 AWS

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;

namespace Amazon.DocSamples.S3
{
    class GenPresignedURLTest
    {
        private const string accessPointArn = "*** access point ARN ***";
        private const string objectKey = "*** object key ***";
        // Specify how long the presigned URL lasts, in hours.
        private const double timeoutDuration = 12;
        // Specify your bucket Region (an example Region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            string urlString = GeneratePreSignedURL(timeoutDuration);
        }
        static string GeneratePreSignedURL(double duration)
        {
```

```
            string urlString = "";
            try
            {
                GetPreSignedUrlRequest request1 = new GetPreSignedUrlRequest
                {
                    BucketName = accessPointArn,
                    Key = objectKey,
                    Expires = DateTime.UtcNow.AddHours(duration)
                };
                urlString = s3Client.GetPreSignedURL(request1);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            return urlString;
        }
    }
}
```

Python

下列範例使用了 SDK for Python (Boto3) 來產生預先簽章的 URL 以共用物件。例如，使用 Boto3 用戶端和 `generate_presigned_url` 函數來產生允許您 `GET` 物件的預先簽章的 URL。

```
import boto3
    url = boto3.client('s3').generate_presigned_url(
    ClientMethod='get_object',
    Params={'Bucket': 'ACCESS_POINT_ARN', 'Key': 'OBJECT_KEY'},
    ExpiresIn=3600)
```

如需有關使用 SDK for Python (Boto3) 產生預先簽章的 URL 的詳細資訊，請參閱《AWS SDK for Python (Boto) API 參考》中的「Python」。

使用 AWS CLI

下列範例 AWS CLI 命令會為 Outposts 儲存貯體上的 S3 產生預先簽署的 URL。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

> **Note**
>
> 使用產生預先簽署的 URL 時，預先簽署 URL 的到期時間上限為建立後 7 天。 AWS CLI

```
aws s3 presign s3://arn:aws:s3-outposts:us-east-1:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/example-outpost-access-point/mydoc.txt --expires-in 604800
```

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 presign。

產生預先簽章 URL 以將物件上傳至 S3 on Outposts 儲存貯體

若要授予對存放在本機 Outpost 上物件的有限時間存取權限，而不會更新儲存貯體政策，您可以使用預先簽章 URL。使用預先簽章 URL，身為儲存貯體擁有者的您可以與虛擬私有雲端 (VPC) 中的個人共享物件，或授予他們上傳或刪除物件的能力。

當您使用 AWS SDK 或 AWS Command Line Interface (AWS CLI) 建立預先簽章 URL 時，會建立 URL 與特定動作的關聯。您也可以選擇自訂到期時間 (最低 1 秒，最高 7 天) 來授予預先簽章 URL 有限時間的存取權。當您共用預先簽章 URL 時，VPC 中的個人可以執行內嵌在 URL 中的動作，如同原始簽章使用者一樣。當 URL 到達到期時間時，該 URL 就會過期且再也無法運作。

當您建立預先簽章 URL 時，必須提供安全憑證，然後指定下列項目：

- 適用於 Amazon S3 on Outposts 儲存貯體的存取點 Amazon Resource Name (ARN)
- 物件索引鍵
- HTTP 方法 (PUT 用於上傳物件)
- 過期日期和時間

預先簽章 URL 僅在指定的期間內有效。也就是說，您必須在到期日期和時間之前開始 URL 所允許的操作。您可以多次使用預先簽章 URL，直到到期日期和時間為止。如果使用暫時字符建立了預先簽章的 URL，那麼 URL 會在字符過期時過期，即使您使用較晚的過期時間建立 URL 亦然。

如果預先簽章的 URL 所允許的動作包含多個步驟 (例如分段上傳)，則您必須在到期之前開始所有步驟。如果 S3 on Outposts 嘗試以過期 URL 開始步驟時，您會收到錯誤。

虛擬私有雲端 (VPC) 中可存取預先簽章 URL 的使用者可以上傳物件。例如，VPC 中具有可存取預先簽章 URL 的使用者可以將物件上傳到您的儲存貯體。由於預先簽章 URL 會將 S3 on Outposts 儲存貯體的存取權授予 VPC 中擁有預先簽章 URL 存取權的任何使用者，因此我們建議您妥善保護這些 URL。如需有關保護預先簽章 URL 的詳細資訊，請參閱限制預先簽章的 URL 功能。

任何具備有效安全憑證的使用者，均可建立預先簽章的 URL。然而，只有具備許可執行作為預先簽章 URL 基礎操作的人員，才能建立預先簽章 URL。如需詳細資訊，請參閱 誰可以建立預先簽章的 URL。

使用 AWS SDK 為 S3 on Outposts 物件操作產生預先簽章 URL。

Java

SDK for Java 2.x

此範例顯示如何產生可以於限定時間內用來將物件上傳至 S3 on Outposts 儲存貯體的預先簽章 URL。如需詳細資訊，請參閱 使用適用於 S3 on OutOutposts 的預先簽章 URL。

```
   public static void signBucket(S3Presigner presigner, String
outpostAccessPointArn, String keyName) {

       try {
           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(accessPointArn)
                   .key(keyName)
                   .contentType("text/plain")
                   .build();

           PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))
                   .putObjectRequest(objectRequest)
                   .build();

           PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);


           String myURL = presignedRequest.url().toString();
           System.out.println("Presigned URL to upload a file to: " +myURL);
```

```
            System.out.println("Which HTTP method must be used when uploading a
file: " +
                    presignedRequest.httpRequest().method());

            // Upload content to the S3 on Outposts bucket by using this URL.
            URL url = presignedRequest.url();

            // Create the connection and use it to upload the new object by using
the presigned URL.
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setDoOutput(true);
            connection.setRequestProperty("Content-Type","text/plain");
            connection.setRequestMethod("PUT");
            OutputStreamWriter out = new
OutputStreamWriter(connection.getOutputStream());
            out.write("This text was uploaded as an object by using a presigned
URL.");
            out.close();

            connection.getResponseCode();
            System.out.println("HTTP response code is " +
connection.getResponseCode());

        } catch (S3Exception e) {
            e.getStackTrace();
        } catch (IOException e) {
            e.getStackTrace();
        }
    }
```

Python

SDK for Python (Boto3)

此範例顯示如何產生可於限定時間內執行 S3 on Outposts 動作的預先簽章 URL。如需詳細資訊，請參閱 使用適用於 S3 on OutOutposts 的預先簽章 URL。若要使用 URL 提出請求，請使用 `Requests` 套件。

```
import argparse
import logging
import boto3
```

```
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned S3 on Outposts URL that can be used to perform an
 action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds that the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method,
            Params=method_parameters,
            ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.",
 client_method)
        raise
    return url


def usage_demo():
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('-'*88)
    print("Welcome to the Amazon S3 on Outposts presigned URL demo.")
    print('-'*88)

    parser = argparse.ArgumentParser()
    parser.add_argument('accessPointArn', help="The name of the S3 on Outposts
 access point ARN.")
    parser.add_argument(
```

```
        'key', help="For a GET operation, the key of the object in S3 on
 Outposts. For a "
                    "PUT operation, the name of a file to upload.")
    parser.add_argument(
        'action', choices=('get', 'put'), help="The action to perform.")
    args = parser.parse_args()

    s3_client = boto3.client('s3')
    client_action = 'get_object' if args.action == 'get' else 'put_object'
    url = generate_presigned_url(
        s3_client, client_action, {'Bucket': args.accessPointArn, 'Key':
 args.key}, 1000)

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == 'get':
        response = requests.get(url)
    elif args.action == 'put':
        print("Putting data to the URL.")
        try:
            with open(args.key, 'r') as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(f"Couldn't find {args.key}. For a PUT operation, the key must
 be the "
                  f"name of a file that exists on your computer.")

    if response is not None:
        print("Got response:")
        print(f"Status: {response.status_code}")
        print(response.text)

    print('-'*88)


if __name__ == '__main__':
    usage_demo()
```

Outposts 上的 Amazon S3 與 Outposts 上的本地 Amazon EMR

Amazon EMR 是一種受管叢集平台，可簡化執行大數據架構 (例如Apache Hadoop和)Apache Spark，AWS以處理和分析大量資料。透過使用這些架構和相關的開放原始碼專案，您可以針對分析目的和商業智慧工作負載處理資料。Amazon EMR 還可協助您將大量資料轉換和移出其他資料存放區和AWS資料庫，並在 Outposts 上支援 Amazon S3。有關 Amazon EMR 的更多信息，請參閱 Amazon EMR 管理指南中的 Outposts 上的 Amazon EM R。

對於 Outposts 上的 Amazon S3，Amazon EMR 開始在 7.0.0 版中支持 Apache Hadoop S3A 連接器。舊版的 Amazon EMR 不支援 Outposts 上的本機 S3，且不支援 EMR 檔案系統 (EMRFS)。

支援的應用程式

Amazon EMR 與 Amazon S3 在 Outposts 支持以下應用程序：

- Hadoop
- Spark
- Hue
- Hive
- Sqoop
- Pig
- Hudi
- Flink

如需詳細資訊，請參閱 Amazon EMR 版本指南。

在 Outposts 存儲桶上創建和配置 Amazon S3

Amazon EMR 使用AWS SDK for Java與 Outposts 的 Amazon S3 來存儲輸入數據和輸出數據。您的 Amazon EMR 日誌檔存放在您選取的區域 Amazon S3 位置，而不會存放在前哨的本機位置。如需詳細資訊，請參閱 Amazon EMR 管理指南中的 Amazon EMR 日誌。

為了符合 Amazon S3 和 DNS 要求，Outposts 儲存貯體上的 S3 有命名限制和限制。如需詳細資訊，請參閱 建立 S3 on Outposts 儲存貯體。

使用 Amazon EMR 7.0.0 版及更新版本，您可以在 Outposts 和 S3A 檔案系統上搭配 S3 使用 Amazon EMR。

事前準備

Outposts 上的 S3 許可 — 當您建立 Amazon EMR 執行個體設定檔時，您的角色必須包含 Outposts 上 S3 的 AWS Identity and Access Management (IAM) 命名空間。Outposts 上的 S3 有自己的命名空間，`s3-outposts*`. 如需使用此命名空間的範例原則，請參閱使用 S3 on Outposts 設定 IAM。

S3A 連接器 — 若要將 EMR 叢集設定為從 Outposts 儲存貯體上的 Amazon S3 存取資料，您必須使用 S3A 連接器Apache Hadoop。若要使用連接器，請確保所有 S3 URI 都使用`s3a`配置。如果沒有，您可以設定用於 EMR 叢集的檔案系統實作，以便 S3 URI 與 S3A 連接器搭配使用。

若要設定檔案系統實作以搭配 S3A 連接器使用，您可以使用 EMR 叢集的`fs.`*`file_scheme`*`.impl`和`fs.AbstractFileSystem.`*`file_scheme`*`.impl`組態屬性，其中與您擁有的 S3 URI 類型*`file_scheme`*相對應。若要使用下列範例，請以您自己的資訊取代 *`user input placeholders`*。例如，若要變更使用配置之 S3 URI 的檔案系統實作，請指定下列叢集組態屬性：`s3`

```
[
  {
"Classification": "core-site",
    "Properties": {
    "fs.s3.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
    "fs.AbstractFileSystem.s3.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

若要使用 S3A，請將`fs.`*`file_scheme`*`.impl`組態屬性設定為`org.apache.hadoop.fs.s3a.S3AFileSystem`，然後將`fs.AbstractFileSystem.`*`file_scheme`*`.impl`屬性設定為`org.apache.hadoop.fs.s3a.S3A`。

例如，如果您要存取路徑`s3a://`*`bucket`*`/...`，請將`fs.s3a.impl`屬性設定為`org.apache.hadoop.fs.s3a.S3AFileSystem`，然後將`fs.AbstractFileSystem.s3a.impl`屬性設定為`org.apache.hadoop.fs.s3a.S3A`。

在 Outposts 上開始使用 Amazon EMR 與 Amazon S3

下列主題說明如何在 Outposts 上開始使用 Amazon EMR 搭配 Amazon S3。

主題

- 建立許可政策

- 建立和設定叢集
- 組態概觀
- 考量事項

建立許可政策

在 Outposts 上建立使用 Amazon S3 的 EMR 叢集之前，您必須先建立 IAM 政策以連接到叢集的 Amazon EC2 執行個體設定檔。該政策必須具有訪問 Outposts 接入點 Amazon 資源名稱（ARN）上的 S3 的許可。如需在 Outposts 上為 S3 建立 IAM 政策的詳細資訊，請參閱使用 S3 on Outposts 設定 IAM。

下列範例原則顯示如何授與所需權限。建立政策後，將政策連接至用於建立 EMR 叢集的執行個體設定檔角色，如 the section called “建立和設定叢集” 一節中所述。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

```
{
"Version":"2012-10-17",
  "Statement": [
        {
  "Effect": "Allow",
            "Resource": "arn:aws:s3-outposts:us-
west-2:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/access-point-name,
            "Action": [
                "s3-outposts:*"
            ]
        }
    ]

 }
```

建立和設定叢集

若要建立在 Outposts 上使用 S3 執行 Spark 的叢集，請在主控台中完成以下步驟。

若要建立在 Outposts 上Spark使用 S3 執行的叢集

1. 請在 https://console.aws.amazon.com/elasticmapreduce/ 開啟 Amazon EMR 主控台。
2. 在左側導覽窗格中選擇 Clusters (叢集)。
3. 選擇 建立叢集。

4. 對於 Amazon EMR 發行版本，請選擇emr-7.0.0或更新版本。
5. 針對應用程式套件，選擇 Spark 互動式。然後選取您要包含在叢集中的任何其他受支援的應用程式。
6. 若要在 Outposts 上啟用 Amazon S3，請輸入您的組態設定。

   範例組態設定

   若要使用下列範例組態設定，請以您自己*user input placeholders*的資訊取代。

```
[
 {
   "Classification": "core-site",
   "Properties": {
     "fs.s3a.bucket.DOC-EXAMPLE-BUCKET.accesspoint.arn": "arn:aws:s3-outposts:us-
west-2:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/access-point-name"
     "fs.s3a.committer.name": "magic",
     "fs.s3a.select.enabled": "false"
    }
  },
  {
    "Classification": "hadoop-env",
    "Configurations": [
      {
        "Classification": "export",
        "Properties": {
          "JAVA_HOME": "/usr/lib/jvm/java-11-amazon-corretto.x86_64"
          }
       }
     ],
     "Properties": {}
   },
   {
     "Classification": "spark-env",
     "Configurations": [
       {
         "Classification": "export",
         "Properties": {
           "JAVA_HOME": "/usr/lib/jvm/java-11-amazon-corretto.x86_64"
         }
       }
      ],
      "Properties": {}
```

```
    },
    {
     "Classification": "spark-defaults",
     "Properties": {
       "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-11-amazon-
corretto.x86_64",
       "spark.sql.sources.fastS3PartitionDiscovery.enabled": "false"
     }
    }
  ]
```

7. 在 [網路] 區段中，選擇AWS Outposts機架上的虛擬私有雲 (VPC) 和子網路。如需 Outposts 上的 Amazon EMR 的詳細資訊，請參閱《Amazon EMR 管理指南》AWS Outposts中的〈EM R 叢集〉。
8. 在 Amazon EMR 適用的 EC2 執行個體設定檔區段中，選擇具有您先前建立的許可政策附加的 IAM 角色。
9. 設定剩餘的叢集設定，然後選擇 [建立叢集]。

組態概觀

下表說明 S3A 和Spark組態，以及當您設定使用具有 Amazon EMR 的 Outposts 上使用 S3 的叢集時，要為其參數指定的值。

S3A 配置

參數	預設值	Outposts S3 的所需值	說明
`fs.s3a.aws.credentials.provider`	如果未指定，S3A 會在具有 Outposts 儲存貯體名稱的區域儲存貯體中尋找 S3。	Outposts 儲存貯體上 S3 的存取點 ARN	Amazon S3 on Outposts 支援僅限 Virtual Private Cloud (VPC) 的存取點，作為存取您的 Outpost 儲存貯體的唯一方法。
`fs.s3a.committer.name`	`file`	`magic`	魔法提交者是 Outposts 上 S3 唯一支持的提交者。

參數	預設值	Outposts S3 的所需值	說明
fs.s3a.select.enabled	TRUE	FALSE	Outposts 不支持 S3 選擇。
JAVA_HOME	/usr/lib/jvm/java-8	/usr/lib/jvm/java-11-amazon-corretto.x86_64	S3A 上的 Outposts 上的 S3 需要Java版本 11。

星火配置

參數	預設值	Outposts S3 的所需值	說明
spark.sql.sources.fastS3PartitionDiscovery.enabled	TRUE	FALSE	Outposts 上的 S3 不支持快速分區。
spark.executorEnv.JAVA_HOME	/usr/lib/jvm/java-8	/usr/lib/jvm/java-11-amazon-corretto.x86_64	S3A 上的 Outposts 上的 S3 需要 Java 版本 11。

考量事項

在 Outposts 儲存貯體上整合 Amazon EMR 與 S3 時，請考慮下列事項：

- Outposts 上的 Amazon S3 支持 Amazon EMR 版本 7.0.0 及更高版本。
- 需要 S3A 連接器才能在配備 Amazon EMR 的 Outposts 上使用 S3。只有 S3A 具有在 Outposts 存儲桶上與 S3 交互所需的功能。如需 S3A 連接器設定的資訊，請參閱先決條件。
- Outposts 上的 Amazon S3 僅支援使用 Amazon S3 受管金鑰 (SSE-S3) 與 Amazon EMR 進行伺服器端加密。如需詳細資訊，請參閱 the section called “資料加密”。

- Outposts 上的 Amazon S3 不支持使用 S FileOutputCommitter 3A 寫入。在 Outposts 儲存貯體 FileOutputCommitter 上使用 S3 上的 S3A 寫入會導致下列錯誤：InvalidStorageClass: 您指定的儲存類別無效。
- Outposts 上的 Amazon S3 不支援 Amazon EMR 無伺服器或 EKS 上的 Amazon EMR。
- 亞馬遜 EMR 日誌存放在您選擇的區域 Amazon S3 位置，而不是本地存放在 Outposts 儲存貯體的 S3 中。

授權和身份驗證緩存

Outposts 上的 S3 可以在 Outposts 機架上安全地快取本地身份驗證和授權資料。緩存刪除每個請求到父級 AWS 區域 的往返。這消除了網絡往返引入的變化性。使用 Outposts 上 S3 中的身份驗證和授權緩存，您可以獲得一致的延遲，這些延遲與 Outposts 和. AWS 區域

當您在 Outposts 上發出 S3 API 請求時，身份驗證和授權數據將被安全地緩存。然後，快取的資料會用於驗證後續 S3 物件 API 請求。Outposts 上的 S3 只會在使用簽名版本 4A (SigV4a) 簽署請求時快取身份驗證和授權資料。緩存存儲在前哨服務 S3 內的 Outposts 本地。當您提出 S3 API 請求時，它會以非同步方式重新整理。緩存被加密，並沒有明文加密密鑰存儲在 Outposts 上。

當前哨連接到時，緩存有效期最多為 10 分鐘。 AWS 區域當您在 Outposts 上發出 S3 API 請求時，它會以非同步方式重新整理，以確保使用最新的政策。如果前哨與斷開連接 AWS 區域，則緩存將有效期長達 12 小時。

設定授權和驗證快取

Outposts 上的 S3 會針對使用 SigV4a 演算法簽署的請求自動快取身份驗證和授權資料。如需詳細資訊，請參閱AWS Identity and Access Management 使用者指南中的簽署 AWS API 要求。SigV4a 演算法可在最新版本的開發套件中使用。 AWS 您可以通過對通AWS 用運行時（CRT）庫的依賴來獲取它。

您需要使用最新版本的 AWS SDK 並安裝最新版本的 CRT。例如，您可以運行`pip install awscrt`以獲取 Boto3 的最新版本的 CRT。

Outposts 上的 S3 不會為使用 SIGv4 演算法簽署的請求快取身份驗證和授權資料。

驗證簽署

您可以使用 AWS CloudTrail 來驗證請求是否已使用 SigV4a 簽署。如需在 Outposts 上設 CloudTrail 定 S3 的詳細資訊，請參閱使用 AWS CloudTrail 日誌監控 S3 on Outposts。

設定完成後 CloudTrail，您可以在 CloudTrail 記錄檔`SignatureVersion`欄位中驗證要求的簽署方式。使用 SigV4a 簽署的請求將設定為`SignatureVersion`。`AWS 4-ECDSA-P256-SHA256`使用 Sigv4 簽署的要求將`SignatureVersion`設定為`AWS 4-HMAC-SHA256`。

S3 on Outposts 中的安全性

雲端安全是 AWS 最重視的一環。身為 AWS 的客戶，您將能從資料中心和網路架構中獲益，這些都是專為最重視安全的組織而設計的。

安全是 AWS 與您共同肩負的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端本身的安全 – AWS 負責保護在 AWS 雲端 中執行 AWS 服務 的基礎設施。AWS 也提供您可安全使用的服務。第三方稽核人員會定期測試和驗證我們安全性的有效性，作為 AWS 合規計劃的一部分。若要了解適用於 Amazon S3 on Outposts 的合規計畫，請參閱 AWS 合規計畫的服務範圍 。
- 雲端內部的安全：您的責任取決於所使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件有助於您了解如何在使用 S3 on Outposts 時套用共同的責任模型。下列主題說明如何將 S3 on Outposts 設定為達到您的安全及合規目標。您也將了解如何使用其他 AWS 服務，幫助您監控並保護 S3 on Outposts 資源。

主題

- S3 on Outposts 中的資料加密
- AWS PrivateLink 對於 Outposts 的 S3
- AWS 第 4 版簽署程序 (SigV4) 驗證特定的政策索引鍵
- 適用於 Amazon S3 on Outposts 的 AWS 受管政策
- 針對 Amazon S3 on Outposts 使用服務連結角色

S3 on Outposts 中的資料加密

依預設，所有存放在 Amazon S3 on Outposts 中的資料均會使用伺服器端加密與 Amazon S3 受管加密金鑰 (SSE-S3) 進行加密。如需詳細資訊，請參閱 使用 Amazon S3 受管金鑰 (SSE-S3) 進行伺服器端加密。

您可以選擇性以客戶提供的加密金鑰 (SSE-C) 使用伺服器端加密。若要使用 SSE-C，請指定加密金鑰做為物件 API 請求的一部分。伺服器端加密只會加密物件資料，非物件中繼資料。如需詳細資訊，請參閱 搭配客戶提供的金鑰 (SSE-C) 使用伺服器端加密。

> **Note**
>
> S3 on Outposts 不支援使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 進行伺服器端加密。

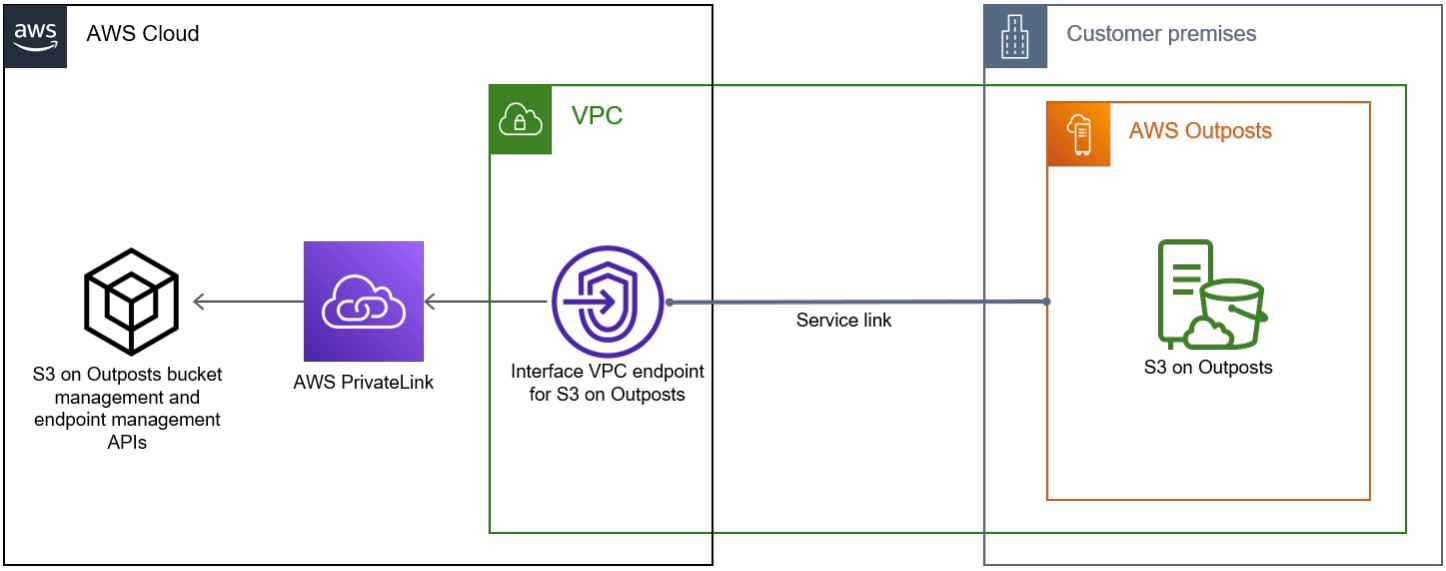
AWS PrivateLink 對於 Outposts 的 S3

Outposts 上的 S3 支援 AWS PrivateLink，可透過虛擬私有網路內的私有端點直接管理存取 Outposts 上的 S3 儲存。如此您就能使用自己的虛擬私有雲端 (VPC) 中的私有 IP 地址，進而簡化內部網路架構並在您的 Outposts 物件儲存上執行管理操作。使用 AWS PrivateLink 無需使用公用 IP 位址或 Proxy 伺服器。

使 AWS PrivateLink 用 Outposts 上的 Amazon S3，您可以在虛擬私有雲端 (VPC) 中佈建介面虛擬私有雲端節點，以便在 Outposts 儲存貯體管理和端點管理 API 上存取 S3。您可透過虛擬私有網路 (VPN) 或 AWS Direct Connect，直接從部署在 VPC 中或內部部署中的應用程式存取介面 VPC 端點。您可以透過以下方式存取值區和端點管理 API AWS PrivateLink。 AWS PrivateLink 不支援資料傳輸 API 作業，例如 GET、PUT 和類似的 API。這些操作已透過 S3 on Outposts 端點和存取點組態私下傳輸。如需詳細資訊，請參閱 適用於 S3 on Outposts 的網路。

介面端點由一個或多個彈性網路介面 (ENI) 來表示，這些是在 VPC 的子網路中指派的私有 IP 地址。對 S3 on Outposts 介面端點發出的請求會自動路由至 AWS 網路上的 S3 on Outposts 儲存貯體和端點管理 API。您也可以透過 AWS Direct Connect 或 AWS Virtual Private Network (AWS VPN) 從內部部署應用程式存取 VPC 中的介面端點。如需有關如何將 VPC 與內部部署網路連線的詳細資訊，請參閱《AWS Direct Connect 使用者指南》和《AWS Site-to-Site VPN 使用者指南》。

介面端點透過 AWS 網路路由 Outposts 儲存貯體和端點管理 API 上 S3 的請求 AWS PrivateLink，如下圖所示。



如需有關介面端點的一般資訊，請參閱《AWS PrivateLink指南》中的介面 VPC 端點 (AWS PrivateLink)。

主題

- 法規與限制
- 存取 S3 on Outposts 介面端點
- 更新內部部署 DNS 組態
- 為 S3 on Outposts 建立 VPC 端點政策
- 為 S3 on Outposts 建立儲存貯體政策與 VPC 端點政策

法規與限制

當您透過 Outposts 儲存貯體和端點管理 API 存取 S3 時 AWS PrivateLink，會套用 VPC 限制。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的介面端點屬性和限制以及AWS PrivateLink 配額。

此外， AWS PrivateLink 不支援下列項目：

- 聯邦資訊處理標準 (FIPS) 端點
- S3 on Outposts 資料傳輸 API，例如 GET、PUT 和類似的物件 API 操作。
- 私有 DNS

存取 S3 on Outposts 介面端點

若要使用存取 Outposts 儲存貯體和端點管理 API 上的 S3 AWS PrivateLink，您必須更新應用程式以使用端點特定的 DNS 名稱。建立介面端點時， AWS PrivateLink 會在 Outposts 名稱上產生兩種端點特定的 S3 類型：區域和區域。

- 地區 DNS 名稱 — 包括唯一的 VPC 端點識別碼、服務識別碼 AWS 區域`vpce.amazonaws.com`、和，*`vpce-1a2b3c4d-5e6f`*`.s3-outposts.us-east-1.vpce.amazonaws.com`例如。
- 區域 DNS 名稱 — 包括唯一的 VPC 端點識別碼、可用區域、服務識別碼 AWS 區域`vpce.amazonaws.com`、和 (例如)。*`vpce-1a2b3c4d-5e6f`*`-us-east-1a.s3-outposts.us-east-1.vpce.amazonaws.com`如果您的架構可隔離可用區域，則可以使用此選項。例如，您可以將地區 DNS 名稱用於故障遏止或降低區域資料傳輸成本。

> **Important**
>
> S3 on Outposts 介面端點是從公有 DNS 網域解析而得。S3 on Outposts 不支援私有 DNS。針對所有儲存貯體與端點管理 API 使用 `--endpoint-url` 參數。

AWS CLI 例子

使用 `--region` 和 `--endpoint-url` 參數透過 S3 on Outposts 介面端點存取儲存貯體管理與端點管理 API。

Example ：使用端點 URL 列出具有 S3 控制項 API 的儲存貯體

在下列範例中，將區域 *`us-east-1`*、VPC 端點 URL *`vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com`* 及帳戶 ID *`111122223333`* 取代為適當的資訊。

```
aws s3control list-regional-buckets --region us-east-1 --endpoint-url
 https://vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com --account-
id 111122223333
```

AWS SDK 範例

將 SDK 更新至最新版本，並設定您的用戶端使用端點 URL，以存取 S3 on Outposts 介面端點的 S3 控制 API。如需詳細資訊，請參閱 AWS PrivateLink的AWS SDK 範例。

SDK for Python (Boto3)

Example ：使用端點 URL 存取 S3 控制 API

在下列範例中，將區域 *us-east-1* 和 VPC 端點 URL *vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com* 取代為適當的資訊。

```
control_client = session.client(
service_name='s3control',
region_name='us-east-1',
endpoint_url='https://vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com'
)
```

如需詳細資訊，請參閱《Boto3 開發人員指南》中的 AWS PrivateLink for Amazon S3。

SDK for Java 2.x

Example ：使用端點 URL 存取 S3 控制 API

在下列範例中，將 VPC 端點 URL *vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com* 和區域 *Region.US_EAST_1* 取代為適當的資訊。

```
// control client
Region region = Region.US_EAST_1;
s3ControlClient = S3ControlClient.builder().region(region)

 .endpointOverride(URI.create("https://vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com"))
                                  .build()
```

如需詳細資訊，請參閱 AWS SDK for Java API 參考中的 S3ControlClient。

更新內部部署 DNS 組態

使用端點特定 DNS 名稱來存取 S3 on Outposts 儲存貯體管理與端點管理 API 的介面端點時，您不必更新內部部署 DNS 解析器。您可以使用公有 S3 on Outposts DNS 網域中介面端點的私有 IP 地址，來解析端點特定的 DNS 名稱。

為 S3 on Outposts 建立 VPC 端點政策

若要為 S3 on Outposts 建立 VPC 介面端點，請參閱 AWS PrivateLink 指南中的建立 VPC 端點。

為 S3 on Outposts 建立儲存貯體政策與 VPC 端點政策

您可以將端點政策連接至控制 S3 on Outposts 存取權的 VPC 端點。您還可以使用 S3 on Outposts 儲存貯體政策的 `aws:sourceVpce` 條件，來限制特定 VPC 端點對特定儲存貯體的存取。透過 VPC 端點政策，您可以控制存取 S3 on Outposts 儲存貯體管理 API 與端點管理 API。透過儲存貯體政策，您可以控制存取 S3 on Outposts 儲存貯體管理 API。然而，您無法使用 `aws:sourceVpce` 管理對 S3 on Outposts 其物件動作的存取。

S3 on Outposts 的存取政策指定下列資訊：

- 允許或拒絕其動作的 AWS Identity and Access Management (IAM) 主體。
- 遭允許或拒絕的 S3 控制項動作。
- 遭允許或拒絕其動作的 S3 on Outposts 資源。

下列範例顯示了限制儲存貯體或端點存取權的政策。如需 VPC 連線的詳細資訊，請參閱 AWS 白皮書 Amazon Virtual Pri vate Cloud 連線選項中的網路到 VPC 連線選項。

> **Important**
>
> - 當套用本節所述的 VPC 端點範例政策時，您可能會在無意間封鎖您對儲存貯體的存取。會限制儲存貯體存取源自您 VPC 端點之連線的儲存貯體許可，可能會封鎖所有對儲存貯體的連線。如需有關如何修復此問題的資訊，請參閱我的儲存貯體政策有錯誤的 VPC 或 VPC 端點 ID。我該如何修復政策，讓我可以存取儲存貯體？(位於 AWS Support 知識中心)。
> - 使用下列範例儲存貯體政策之前，請以適合您使用案例的適當值取代 VPC 端點 ID。否則，您將無法存取儲存貯體。
> - 如果您的政策僅允許從特定 VPC 端點存取 S3 on Outposts 儲存貯體，則它會停用對該儲存貯體的主控台存取權，因為主控台請求不是源自指定的 VPC 端點。

主題

範例：限制從 VPC 端點對特定儲存貯體的存取

您可以建立端點政策，以限制只存取特定 S3 on Outposts 儲存貯體。下列原則僅限於 GetBucketPolicy 動作的存取。*example-outpost-bucket*若要使用這個政策，請使用您的值來取代範例值。

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909151",
  "Statement": [
    { "Sid": "Access-to-specific-bucket-only",
      "Principal": {"AWS":"111122223333"},
      "Action": "s3-outposts:GetBucketPolicy",
      "Effect": "Allow",
      "Resource": "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outpost-
bucket"
    }
  ]
}
```

範例：在 S3 on Outposts 儲存貯體政策中拒絕從特定 VPC 端點存取

Outposts 儲存貯體政策上的以下 S3 拒絕透過 *vpce-1a2b3c4d* VPC 端點存取儲存*example-outpost-bucket*貯體。 GetBucketPolicy

`aws:sourceVpce` 條件會指定端點，且不需要 VPC 端點資源的 Amazon Resource Name (ARN)，只需要端點 ID。若要使用這個政策，請使用您的值來取代範例值。

```
{
    "Version": "2012-10-17",
    "Id": "Policy1415115909152",
    "Statement": [
        {
            "Sid": "Deny-access-to-specific-VPCE",
            "Principal": {"AWS":"111122223333"},
            "Action": "s3-outposts:GetBucketPolicy",
            "Effect": "Deny",
            "Resource": "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outpost-
bucket",
            "Condition": {
```

```
                "StringEquals": {"aws:sourceVpce": "vpce-1a2b3c4d"}
            }
        }
    ]
}
```

AWS 第 4 版簽署程序 (SigV4) 驗證特定的政策索引鍵

下表顯示與 AWS 第 4 版簽署程序 (SigV4) 身分驗證相關且可搭配 Amazon S3 on Outposts 使用的條件金鑰。在儲存貯體政策中，您可以新增這些條件，以便在使用第 4 版簽署程序來驗證請求時強制執行特定行為。如需取得範例政策，請參閱 使用第 4 版簽署程序相關條件金鑰的儲存貯體政策範例。如需使用 Signature 第 4 版身分驗證請求的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的「身分驗證請求 (AWS 第 4 版簽署程序)」

`s3-outposts:*` 動作或任何 S3 on Outposts 動作的適用金鑰

適用金鑰	Description (描述)
`s3-outposts:authType`	S3 on Outposts 支援多種不同的身分驗證方法。若要限制傳入請求使用特定身分驗證方法，您可以使用此可選條件金鑰。例如，您可以使用此條件金鑰，僅允許 HTTP `Authorization` 標頭用在請求身分驗證中。 有效值： `REST-HEADER` `REST-QUERY-STRING`
`s3-outposts:signatureAge`	簽章在已驗證請求中有效的時間長度 (以毫秒為單位)。 此條件僅適用於預先簽章 URL。 在第 4 版簽署程序中，簽署金鑰的有效期限最長七天。因此，簽章的有效期限也是最長七天。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的「簽署請求簡介」。您可以使用此條件來進一步限制簽章存留期。 範例值：`600000`

適用金鑰	Description (描述)
s3-outposts:x-amz-content-sha256	您可以使用此條件金鑰以不允許在儲存貯體中未簽署的內容。 當您使用第 4 版簽署程序時，針對使用 Authorization 標頭的請求，會在簽署計算中新增 x-amz-content-sha256 標頭，然後將其值設定為雜湊承載。 您可以在儲存貯體政策中使用此條件金鑰，拒絕任何尚未簽署承載的上傳項目。例如： • 拒絕使用了 Authorization 標頭來驗證請求但並未簽署承載的上傳項目。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的在單個區塊中傳輸承載。 • 拒絕使用預先簽章 URL 的上傳。預先簽章 URL 一律有 UNSIGNED_PAYLOAD 。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的身分驗證請求和身分驗證方法。 有效值：UNSIGNED-PAYLOAD

使用第 4 版簽署程序相關條件金鑰的儲存貯體政策範例

若要使用下列範例，請以您自己的資訊取代 *user input placeholders*。

Example : **s3-outposts:signatureAge**

下列儲存貯體政策會拒絕 example-outpost-bucket 中物件上的任何 S3 on Outposts 預先簽章 URL 請求 (如果簽章已超過 10 分鐘之久)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Deny a presigned URL request if the signature is more than 10
 minutes old",
            "Effect": "Deny",
            "Principal": {"AWS":"444455556666"},
            "Action": "s3-outposts:*",
```

```
            "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
            "Condition": {
                "NumericGreaterThan": {"s3-outposts:signatureAge": 600000},
                "StringEquals": {"s3-outposts:authType": "REST-QUERY-STRING"}
            }
        }
    ]
}
```

Example : **s3-outposts:authType**

下列儲存貯體政策僅允許使用 Authorization 標頭以供請求驗證的任何請求。任何預先簽章 URL 請求都會遭到拒絕，因為預先簽章 URL 會使用查詢參數來提供請求和驗證資訊。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的「身分驗證方法」。

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
               "Sid": "Allow only requests that use the Authorization header for
 request authentication. Deny presigned URL requests.",
               "Effect": "Deny",
               "Principal": {"AWS":"111122223333"},
               "Action": "s3-outposts:*",
               "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
               "Condition": {
                     "StringNotEquals": {
                           "s3-outposts:authType": "REST-HEADER"
                     }
               }
         }
   ]
}
```

Example : **s3-outposts:x-amz-content-sha256**

下列儲存貯體政策會拒絕任何具有未簽署承載的上傳項目，例如使用預先簽章 URL 的上傳。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的身分驗證請求和身分驗證方法。

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
               "Sid": "Deny uploads with unsigned payloads.",
               "Effect": "Deny",
               "Principal": {"AWS":"111122223333"},
               "Action": "s3-outposts:*",
               "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
               "Condition": {
                     "StringEquals": {
                           "s3-outposts:x-amz-content-sha256": "UNSIGNED-PAYLOAD"
                     }
               }
         }
   ]
}
```

適用於 Amazon S3 on Outposts 的 AWS 受管政策

AWS 管理的政策是由 AWS 建立和管理的獨立政策。AWS 管理的政策的設計在於為許多常見使用案例提供許可，如此您就可以開始將許可指派給使用者、群組和角色。

請記住，AWS 管理的政策可能不會授予您特定使用案例的最低權限許可，因為它們可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法更改 AWS 管理的政策中定義的許可。如果 AWS 更新 AWS 管理的政策中定義的許可，更新會影響政策連接的所有主體身分 (使用者、群組和角色)。在推出新的 AWS 服務 或有新的 API 操作可供現有服務使用時，AWS 很可能會更新 AWS 管理的政策。

如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html#aws-managed-policies中的 AWS 受管政策。

AWS 受管政策：AWSS3OnOutpostsServiceRolePolicy

作為服務連結角色 `AWSServiceRoleForS3OnOutposts` 的一部分，協助您管理網路資源。

若要檢視此政策的許可，請參閱 AWSS3OnOutpostsServiceRolePolicy。

AWS 受管政策的 S3 on Outposts 更新

檢視自此服務開始追蹤 S3 on Outposts 的 AWS 受管政策更新以來，所有變更的詳細資訊。

變更	描述	日期
S3 on Outposts 新增了 `AWSS3OnOutpostsServiceRolePolicy`	S3 on Outposts 新增了 `AWSS3OnOutpostsServiceRolePolicy` 作為服務連結角色 `AWSServiceRoleForS3OnOutposts` 的一部分，可協助您管理網路資源。	2023 年 10 月 3 日
S3 on Outposts 開始追蹤變更	S3 on Outposts 開始追蹤其 AWS 受管政策的變更。	2023 年 10 月 3 日

針對 Amazon S3 on Outposts 使用服務連結角色

Amazon S3 on Outposts 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 S3 on Outposts 的一種特殊 IAM 角色類型。服務連結角色是由 S3 on Outposts 預先定義，且包含該服務代表您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓設定 S3 on Outposts 更為容易，因為您不必手動新增必要的許可。S3 on Outposts 會定義其服務連結角色的許可，除非另有定義，否則只有 S3 on Outposts 可以擔任其角色。定義的許可包括信任政策和許可政策，並且該許可政策不能連接到任何其他 IAM 實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。如此可保護您 S3 on Outposts 資源，避免您不小心移除存取資源的許可。

如需有關支援服務連結角色之其他服務的資訊，請參閱可搭配 IAM 運作的 AWS 服務，並尋找 Service-linked roles (服務連結角色) 資料行中顯示為 Yes (是) 的服務。選擇具有連結的 Yes (是)，以檢視該服務的服務連結角色文件。

S3 on Outposts 的服務連結角色許可

S3 on Outposts 使用名為 AWSServiceRoleForS3OnOutposts 的服務連結角色來協助您管理網路資源。

AWSServiceRoleForS3OnOutposts 服務連結角色信任下列服務以擔任角色：

- s3-outposts.amazonaws.com

名為 AWSS3OnOutpostsServiceRolePolicy 的角色許可政策允許 S3 on Outposts 對指定的資源完成下列動作：

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DescribeCoipPools",
                "ec2:GetCoipPoolUsage",
                "ec2:DescribeAddresses",
                "ec2:DescribeLocalGatewayRouteTableVpcAssociations"
            ],
            "Resource": "*",
            "Sid": "DescribeVpcResources"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ],
            "Sid": "CreateNetworkInterface"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ],
```

```
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "CreateTagsForCreateNetworkInterface"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AllocateAddress"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:ipv4pool-ec2/*"
            ],
            "Sid": "AllocateIpAddress"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AllocateAddress"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:elastic-ip/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "CreateTagsForAllocateIpAddress"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DisassociateAddress",
                "ec2:ReleaseAddress",
                "ec2:AssociateAddress"
            ],
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "ReleaseVpcResources"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": [
                        "CreateNetworkInterface",
                        "AllocateAddress"
                    ],
                    "aws:RequestTag/CreatedBy": [
                        "S3 On Outposts"
                    ]
                }
            },
            "Sid": "CreateTags"
        }
    ]
}
```

您必須設定許可，IAM 實體 (如角色) 才能建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

為 S3 on Outposts 建立服務連結角色

您不需要手動建立一個服務連結角色。當您在 AWS Management Console、AWS CLI 或是 AWS API 中建立 S3 on Outposts 端點時，S3 on Outposts 會為您建立服務連結角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您建立 S3 on Outposts 端點時，S3 on Outposts 會再次為您建立服務連結角色。

您也可以使用 IAM 主控台，參考 S3 on Outposts 使用案例來建立服務連結角色。在 AWS CLI CLI 或 AWS API 中，建立一個服務名稱為 `s3-outposts.amazonaws.com` 的服務連結角色。如需詳細資

訊，請參閱《IAM 使用者指南》中的建立服務連結角色。如果您刪除此服務連結角色，您可以使用此相同的程序以再次建立該角色。

編輯 S3 on Outposts 的服務連結角色

S3 on Outposts 不允許您編輯 AWSServiceRoleForS3OnOutposts 服務連結角色。包括角色的名稱也不可編輯，因為可能有各種不同的實體參考角色。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱《IAM 使用者指南》中的編輯服務連結角色。

刪除 S3 on Outposts 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。

> **Note**
>
> 若 S3 on Outposts 服務在您試圖刪除資源時正在使用該角色，刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

刪除 AWSServiceRoleForS3OnOutposts 角色使用的 S3 on Outposts 資源

1. 在所有 AWS 區域 於您的 AWS 帳戶 中，刪除 S3 on Outposts 端點。
2. 使用 IAM 刪除服務連結角色。

   使用 IAM 主控台、AWS CLI 或 AWS API 來刪除 AWSServiceRoleForS3OnOutposts 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

S3 on Outposts 服務連結角色的支援區域

S3 on Outposts 支援在所有提供服務的 AWS 區域 中，使用服務連結角色。如需詳細資訊，請參閱 S3 on Outposts 區域和端點。

管理 S3 on Outposts 儲存貯體

透過 Amazon S3 on Outposts，您可以在 AWS Outposts 上建立 S3 儲存貯體，並針對需要本機資料存取權限、本機資料處理和資料駐留的應用程式，在內部部署輕鬆存放和擷取物件。S3 on Outposts 提供一個全新的儲存類別，即 S3 Outposts (OUTPOSTS)，它使用 Amazon S3 API，目的是在您的 AWS

Outposts Outposts 上的多個裝置和伺服器上以持久、備援的方式存放資料。您可以使用存取點和透過 Virtual Private Cloud (VPC) 的端點連線，與您的 Outpost 儲存貯體進行通訊。就像在 Amazon S3 儲存貯體一樣，您在 Outpost 儲存貯體上可以使用同樣的 API 和功能，包括存取政策、加密和標記。您可以透過 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。如需詳細資訊，請參閱 什麼是 Amazon S3 on Outposts？

如需管理和共享 Amazon S3 on Outposts 儲存貯體容量的詳細資訊，請參閱下列主題。

主題

- 針對您的 S3 on Outposts 儲存貯體管理 S3 版本控制
- 建立和管理 Amazon S3 on Outposts 儲存貯體的生命週期組態
- 複寫 S3 on Outposts 的物件
- 通過使用在 Outposts 上共享 S3 AWS RAM
- 其他使用 S3 on Outposts 的 AWS 服務

針對您的 S3 on Outposts 儲存貯體管理 S3 版本控制

啟用時，S3 版本控制會在相同的儲存貯體中儲存物件的多個不同複本。您可以使用 S3 版本控制，保留、擷取和還原在 Outposts 儲存貯體中所存放每個物件的各個版本。S3 版本控制可協助您從意外的使用者動作和應用程式失敗中復原。

Amazon S3 on Outposts 儲存貯體具有三種版本控制狀態：

- Unversioned (未版本控制) - 如果您從未在儲存貯體上啟用或暫停 S3 版本控制，則表示未版本控制，並且不會傳回任何 S3 版本控制狀態。如需 S3 版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。
- Enabled (已啟用) - 針對儲存貯體中的物件啟用 S3 版本控制。所有新增至儲存貯體的物件都會收到唯一的版本 ID。啟用版本控制時已存在於儲存貯體中的物件的版本 ID 為 `null`。如果您使用其他操作修改這些 (或任何其他) 物件，例如 PutObject，新物件會取得唯一的版本 ID。
- Suspended (已暫停) - 針對儲存貯體中的物件暫停 S3 版本控制。所有在版本控制暫停之後新增至儲存貯體的物件都會收到版本 ID `null`。如需詳細資訊，請參閱 將物件新增至暫停版本控制的儲存貯體。

在您針對 S3 on Outposts 儲存貯體啟用 S3 版本控制之後，此儲存貯體永遠無法回復為未使用版本控制狀態。不過，您可以暫停版本控制。如需 S3 版本控制的詳細資訊，請參閱「在 S3 儲存貯體中使用版本控制」。

對於儲存貯體中的每個物件，您都有一個目前版本，以及零或多個非目前的版本。若要降低儲存成本，您可以將儲存貯體 S3 生命週期規則設定為在指定的時段之後使非目前版本過期。如需詳細資訊，請參閱 建立和管理 Amazon S3 on Outposts 儲存貯體的生命週期組態。

下列範例向您展示如何使用 AWS Management Console 和 AWS Command Line Interface (AWS CLI) 啟用現有 S3 on Outposts 儲存貯體的版本控制。若要建立已啟用版本控制的 S3 儲存貯體，請參閱 建立 S3 on Outposts 儲存貯體。

> **Note**
>
> 建立儲存貯體的 AWS 帳戶 擁有該儲存貯體，且是唯一可以向其提交動作的帳戶。儲存貯體具有組態屬性，例如 Outpost、標籤、預設加密和存取點設定。存取點設定包含用於存取儲存貯體中物件的虛擬私有雲端 (VPC) 和存取點政策，以及其他中繼資料。如需詳細資訊，請參閱 S3 on Outposts 規格。

使用 S3 主控台

編輯儲存貯體的 S3 版本控制設定

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇您要針對其啟用 S3 版本控制的 Outposts 儲存貯體。
4. 選擇 Properties (屬性) 標籤。
5. 在 Bucket Versioning (儲存貯體版本控制) 底下，選擇 Edit (編輯)。
6. 選擇下列其中一個選項來編輯儲存貯體的 S3 版本控制：
   - 若要暫停 S3 版本控制並停止建立新的物件版本，請選擇 Suspend (暫停)。
   - 若要啟用 S3 版本控制並儲存每個物件的多個不同複本，請選擇 Enable (啟用)。
7. 選擇 Save changes (儲存變更)。

使用 AWS CLI

若要使用 AWS CLI 啟用或暫停儲存貯體的 S3 版本控制，請使用 `put-bucket-versioning` 命令，如下列範例所示。若要使用這些範例，請以您自己的資訊取代每個 *`user input placeholder`*。

如需詳細資訊，請參閱《AWS CLI 參考》中的 put-bucket-versioning。

Example ：啟用 S3 版本控制

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket --versioning-configuration Status=Enabled
```

Example ：暫停 S3 版本控制

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket --versioning-configuration Status=Suspended
```

建立和管理 Amazon S3 on Outposts 儲存貯體的生命週期組態

您可以使用 S3 生命週期，最佳化 Amazon S3 on Outposts 的儲存容量。您可以建立生命週期規則，在物件老化或取代為較新的版本時使這些物件過期。您可以建立、啟用、停用或刪除生命週期規則。

如需 S3 生命週期的詳細資訊，請參閱 管理儲存生命週期。

> **Note**
>
> 建立儲存貯體的 AWS 帳戶 擁有該儲存貯體，且是唯一可以建立、啟用、停用或刪除生命週期規則的帳戶。

若要為您的 S3 on Outposts 儲存貯體建立和管理生命週期組態，請參閱下列主題。

主題

- 使用 AWS Management Console 建立和管理生命週期規則
- 使用適用於 Java 的 AWS CLI 和 SDK 建立和管理生命週期組態

使用 AWS Management Console 建立和管理生命週期規則

您可以使用 S3 生命週期，最佳化 Amazon S3 on Outposts 的儲存容量。您可以建立生命週期規則，在物件老化或取代為較新的版本時使這些物件過期。您可以建立、啟用、停用或刪除生命週期規則。

如需 S3 生命週期的詳細資訊，請參閱 管理儲存生命週期。

> **Note**
>
> 建立儲存貯體的 AWS 帳戶 擁有該儲存貯體，且是唯一可以建立、啟用、停用或刪除生命週期規則的帳戶。

若要使用 AWS Management Console 建立和管理 S3 on Outposts 生命週期規則，請參閱下列主題。

主題

- 建立生命週期規則
- 啟用生命週期規則
- 編輯生命週期規則
- 刪除生命週期規則

建立生命週期規則

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇您要為其建立生命週期規則的 Outposts 儲存貯體。
4. 選擇 Management (管理) 索引標籤，然後選擇 Create lifecycle rule (建立生命週期規則)。
5. 輸入 Lifecycle rule name (生命週期規則名稱) 的值。
6. 在 Rule scope (規則範圍) 下，選擇下列其中一個選項：
   - 若要將範圍限制為特定篩選條件，請選擇 Limit the scope of this rule using one or more filters (使用一或多個篩選條件限制此規則的範圍)。然後，新增字首篩選條件、標籤或物件大小。
   - 若要將規則套用至儲存貯體中的所有物件，請選擇 Apply to all objects in the bucket (套用至儲存貯體中的所有物件)。
7. 在 Lifecycle rule actions (生命週期規則動作) 下，選擇下列其中一個選項：
   - Expire current versions of objects (讓目前版本的物件過期) - 對於已啟用版本控制的儲存貯體，S3 on Outposts 會新增刪除標記，並將物件保留為非目前版本。對於未使用 S3 版本控制的儲存貯體，S3 on Outposts 會永久刪除這些物件。
   - Permanently delete noncurrent versions of objects (永久刪除非目前版本的物件) - S3 on Outposts 會永久刪除非目前版本的物件。

- Delete expired object delete markers or incomplete multipart uploads (刪除過期物件刪除標記或未完成的分段上傳) - S3 on Outposts 會永久刪除過期物件刪除標記或未完成的分段上傳。

  如果您使用物件標籤來限制生命週期規則的範圍，則無法選擇 Delete expired object delete markers (刪除過期物件刪除標記)。如果您選擇 Expire current object versions (讓目前版本的物件過期)，也無法選擇 Delete expired object delete markers (刪除過期物件刪除標記)。

> **Note**
>
> 大小型篩選條件無法與刪除標記和未完成的分段上傳搭配使用。

8. 如果您選擇 Expire current versions of objects (讓目前版本的物件過期) 或 Permanently delete noncurrent versions of objects (永久刪除非目前版本的物件)，請根據特定日期或物件的存留期來設定規則觸發條件。
9. 如果您選擇了 Delete expired object delete markers (刪除過期物件刪除標記)，為了確認您想要刪除過期物件刪除標記，請選取 Delete expired object delete markers (刪除過期物件刪除標記)。
10. 在 Timeline Summary (時間軸摘要) 下，檢閱您的生命週期規則，然後選擇 Create rule (建立規則)。

啟用生命週期規則

若要啟用或停用儲存貯體生命週期規則

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇您要啟用或停用其生命週期規則的 Outposts 儲存貯體。
4. 選擇 Management (管理) 索引標籤，然後在 Lifecycle rule (生命週期規則) 下，選擇您要啟用或停用的規則。
5. 對於 Action (動作)，選擇 Enable or disable rule (啟用或停用規則)。

編輯生命週期規則

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇您要為其編輯生命週期規則的 Outposts 儲存貯體。

4. 選擇 Management (管理) 索引標籤，然後選擇您要編輯的 Lifecycle rule (生命週期規則)。
5. (選用) 更新 Lifecycle rule name (生命週期規則名稱) 的值。
6. 在 Rule scope (規則範圍) 下，視需要編輯範圍：

   - 若要將範圍限制為特定篩選條件，請選擇 Limit the scope of this rule using one or more filters (使用一或多個篩選條件限制此規則的範圍)。然後，新增字首篩選條件、標籤或物件大小。
   - 若要將規則套用至儲存貯體中的所有物件，請選擇 Apply to all objects in the bucket (套用至儲存貯體中的所有物件)。
7. 在 Lifecycle rule actions (生命週期規則動作) 下，選擇下列其中一個選項：

   - Expire current versions of objects (讓目前版本的物件過期) - 對於已啟用版本控制的儲存貯體，S3 on Outposts 會新增刪除標記，並將物件保留為非目前版本。對於未使用 S3 版本控制的儲存貯體，S3 on Outposts 會永久刪除這些物件。
   - Permanently delete noncurrent versions of objects (永久刪除非目前版本的物件) - S3 on Outposts 會永久刪除非目前版本的物件。
   - Delete expired object delete markers or incomplete multipart uploads (刪除過期物件刪除標記或未完成的分段上傳) - S3 on Outposts 會永久刪除過期物件刪除標記或未完成的分段上傳。

     如果您使用物件標籤來限制生命週期規則的範圍，則無法選擇 Delete expired object delete markers (刪除過期物件刪除標記)。如果您選擇 Expire current object versions (讓目前版本的物件過期)，也無法選擇 Delete expired object delete markers (刪除過期物件刪除標記)。

   > **Note**
   >
   > 大小型篩選條件無法與刪除標記和未完成的分段上傳搭配使用。

8. 如果您選擇 Expire current versions of objects (讓目前版本的物件過期)或 Permanently delete noncurrent versions of objects (永久刪除非目前版本的物件)，請根據特定日期或物件存留期來設定規則觸發條件。
9. 如果您選擇了 Delete expired object delete markers (刪除過期物件刪除標記)，為了確認您想要刪除過期物件刪除標記，請選取 Delete expired object delete markers (刪除過期物件刪除標記)。
10. 選擇 Save (儲存)。

刪除生命週期規則

1. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇您要為其刪除生命週期規則的 Outposts 儲存貯體。
4. 選擇 Management (管理) 索引標籤，然後在 Lifecycle rule (生命週期規則)，選擇您要刪除的規則。
5. 選擇 Delete (刪除)。

使用適用於 Java 的 AWS CLI 和 SDK 建立和管理生命週期組態

您可以使用 S3 生命週期，最佳化 Amazon S3 on Outposts 的儲存容量。您可以建立生命週期規則，在物件老化或取代為較新的版本時使這些物件過期。您可以建立、啟用、停用或刪除生命週期規則。

如需 S3 生命週期的詳細資訊，請參閱 管理儲存生命週期。

> **Note**
>
> 建立儲存貯體的 AWS 帳戶 擁有該儲存貯體，且是唯一可以建立、啟用、停用或刪除生命週期規則的帳戶。

若要使用 AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 建立和管理 S3 on Outposts 儲存貯體生命週期組態，請參閱下列範例。

主題

- 放置生命週期組態
- 取得 S3 on Outposts 儲存貯體的生命週期組態

放置生命週期組態

AWS CLI

下列 AWS CLI 範例將生命週期組態政策放置在 Outpost 儲存貯體上。此政策指定，所有包含標記字首 (*myprefix*) 和標籤的物件會在 10 天後過期。若要使用此範例，請以您自己的資訊取代每個 *user input placeholder*。

1. 將生命週期組態原則儲存至 JSON 檔案。在此範例中，檔案命名為 `lifecycle1.json`。

```
{
    "Rules": [
```

```
        {
            "ID": "id-1",
            "Filter": {
                "And": {
                    "Prefix": "myprefix",
                    "Tags": [
                        {
                            "Value": "mytagvalue1",
                            "Key": "mytagkey1"
                        },
                        {
                            "Value": "mytagvalue2",
                            "Key": "mytagkey2"
                        }
                    ],
                    "ObjectSizeGreaterThan": 1000,
                    "ObjectSizeLessThan": 5000
                }
            },
            "Status": "Enabled",
            "Expiration": {
                "Days": 10
            }
        }
    ]
}
```

2. 提交 JSON 檔案以做為 put-bucket-lifecycle-configuration CLI 命令的一部分。若要執行此命令，請以您自己的資訊取代每個 *user input placeholder*。如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 put-bucket-lifecycle-configuration。

```
aws s3control put-bucket-lifecycle-configuration --account-id 123456789012 --
bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/
bucket/example-outposts-bucket --lifecycle-configuration file://lifecycle1.json
```

SDK for Java

下列適用於 Java 的開發套件範例將生命週期組態放置在 Outpost 儲存貯體上。此生命週期組態指定，所有包含標記字首 (*myprefix*) 和標籤的物件會在 10 天後過期。若要使用此範例，請以您自己的資訊取代每個 *user input placeholder*。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的 PutBucketLifecycleConfiguration。

```
import com.amazonaws.services.s3control.model.*;

public void putBucketLifecycleConfiguration(String bucketArn) {

    S3Tag tag1 = new S3Tag().withKey("mytagkey1").withValue("mytagkey1");
    S3Tag tag2 = new S3Tag().withKey("mytagkey2").withValue("mytagkey2");

    LifecycleRuleFilter lifecycleRuleFilter = new LifecycleRuleFilter()
            .withAnd(new LifecycleRuleAndOperator()
                    .withPrefix("myprefix")
                    .withTags(tag1, tag2))
                    .withObjectSizeGreaterThan(1000)
                    .withObjectSizeLessThan(5000);

    LifecycleExpiration lifecycleExpiration = new LifecycleExpiration()
            .withExpiredObjectDeleteMarker(false)
            .withDays(10);

    LifecycleRule lifecycleRule = new LifecycleRule()
            .withStatus("Enabled")
            .withFilter(lifecycleRuleFilter)
            .withExpiration(lifecycleExpiration)
            .withID("id-1");

    LifecycleConfiguration lifecycleConfiguration = new LifecycleConfiguration()
            .withRules(lifecycleRule);

    PutBucketLifecycleConfigurationRequest reqPutBucketLifecycle = new
 PutBucketLifecycleConfigurationRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withLifecycleConfiguration(lifecycleConfiguration);

    PutBucketLifecycleConfigurationResult respPutBucketLifecycle =
 s3ControlClient.putBucketLifecycleConfiguration(reqPutBucketLifecycle);
    System.out.printf("PutBucketLifecycleConfiguration Response: %s%n",
 respPutBucketLifecycle.toString());

}
```

取得 S3 on Outposts 儲存貯體的生命週期組態

AWS CLI

下列 AWS CLI 範例在 Outposts 儲存貯體上取得生命週期組態。若要執行此命令，請以您自己的資訊取代每個 *user input placeholder*。如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 get-bucket-lifecycle-configuration。

```
aws s3control get-bucket-lifecycle-configuration --account-id 123456789012 --bucket
 arn:aws:s3-outposts:<your-region>:123456789012:outpost/op-01ac5d28a6a232904/
bucket/example-outposts-bucket
```

SDK for Java

下列適用於 Java 的開發套件範例取得 Outpost 儲存貯體的生命週期組態。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的 GetBucketLifecycleConfiguration。

```
import com.amazonaws.services.s3control.model.*;

public void getBucketLifecycleConfiguration(String bucketArn) {

    GetBucketLifecycleConfigurationRequest reqGetBucketLifecycle = new
 GetBucketLifecycleConfigurationRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    GetBucketLifecycleConfigurationResult respGetBucketLifecycle =
 s3ControlClient.getBucketLifecycleConfiguration(reqGetBucketLifecycle);
    System.out.printf("GetBucketLifecycleConfiguration Response: %s%n",
 respGetBucketLifecycle.toString());

}
```

複寫 S3 on Outposts 的物件

使用 S3 Replication on AWS Outposts，您可以設定 Amazon S3 on Outposts，以跨不同的 Outposts 或在相同 Outpost 上的儲存貯體之間複寫 S3 物件。您可以使用 S3 Replication on Outposts，在相同或不同的 Outposts 中或跨不同的帳戶維護資料的多個複本，以協助符合資料駐留需求。S3 Replication on Outposts 有助於支援合規儲存需求，以及跨帳戶資料共用。如果需要確保複本與來源資料相同，您

可以使用 S3 Replication on Outposts 來建立保留所有中繼資料的物件複本，例如原始物件建立時間、標籤和版本 ID。

S3 Replication on Outposts 也提供詳細的指標和通知，以監控儲存貯體之間的物件複寫狀態。您可以使用 Amazon CloudWatch 監控複寫進度，方法為追蹤位元組等待複寫、操作等待複寫，以及來源與目標儲存貯體之間的複寫延遲。若要快速診斷並更正組態問題，您也可以將 Amazon EventBridge 設定為接收有關複寫物件失敗的通知。如需進一步了解，請參閱管理複寫。

主題

複寫組態

S3 on Outposts 會以 XML 的形式存放複寫組態。在複寫組態 XML 檔案中，您可以指定 AWS Identity and Access Management (IAM) 角色與一或多個規則。

```
<ReplicationConfiguration>
    <Role>IAM-role-ARN</Role>
    <Rule>
        ...
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
</ReplicationConfiguration>
```

若沒有您的許可，S3 on Outposts 無法複寫物件。您可以使用複寫組態中指定的 IAM 角色來授予 S3 on Outposts 許可。S3 on Outposts 會擔任該 IAM 角色來代您複寫物件。您必須將所需的許可授予 IAM 角色，然後才能開始複寫。如需這些 S3 on Outposts 許可的詳細資訊，請參閱 建立 IAM 角色。

針對下列情況，您可以在複寫組態中新增一個規則：

- 您想要複寫所有物件。
- 您想要複寫物件子集。您在規則中新增篩選條件，以識別物件子集。您可以在篩選條件中指定物件金鑰前綴、標籤，或這兩項的組合，以識別要套用規則的物件子集。

若您想複寫不同的物件子集，可以在複寫組態中新增多項規則。在每個規則中，您可以指定篩選條件以選取不同的物件子集。例如，您可以選擇複寫含有 `tax/` 或 `document/` 索引鍵字首的物件。要做到這一點，您需要新增兩個規則，一個指定 `tax/` 索引鍵字首篩選條件，另一個指定 `document/` 索引鍵字首。

如需 S3 on Outposts 複寫組態和複寫規則的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的ReplicationConfiguration。

S3 Replication on Outposts 的需求

複寫需求如下：

- 目的地 Outpost CIDR 範圍必須與來源 Outpost 子網路表格相關聯。如需詳細資訊，請參閱「建立複寫規則的先決條件」。
- 來源與目的地儲存貯體都必須啟用 S3 版本控制。如需版本控制的詳細資訊，請參閱「針對您的 S3 on Outposts 儲存貯體管理 S3 版本控制」。
- Amazon S3 on Outposts 必須具備許可，才能代您將物件從來源儲存貯體複寫至目的地儲存貯體。這意味著您必須建立一個服務角色，將 `GET` 和 `PUT` 許可委派給 S3 on Outposts。
  1. 在建立服務角色之前，您必須對來源儲存貯體具有 `GET` 許可，以及對目的地儲存貯體具有 `PUT` 許可。
  2. 若要建立服務角色以將許可委派給 S3 on Outposts，您必須先設定許可，以允許 IAM 實體 (使用者或角色) 執行 `iam:CreateRole` 和 `iam:PassRole` 動作。然後，允取 IAM 實體建立服務角色。若要讓 S3 on Outposts 代您擔任服務角色，並將 `GET` 和 `PUT` 許可委派給 S3 on Outposts，您必須將所需的信任和許可政策指派給該角色。如需這些 S3 on Outposts 許可的詳細資訊，請參閱 建立 IAM 角色。如需建立服務角色的詳細資訊，請參閱建立服務角色。

複寫內容為何？

依預設，S3 on Outposts 會複寫下列項目：

- 在您新增複寫組態之後建立的物件。
- 從來源物件到複本的物件中繼資料。如需如何將中繼資料從複本複寫至來源物件的相關資訊，請參閱啟用 Outposts 上的 Amazon S3 複本修改同步時的複寫狀態。

- 物件標籤 (如果有)。

刪除操作對複寫的影響

如果您從來源儲存貯體中刪除物件，預設會執行下列動作：

- 如果您發出 DELETE 請求但未指定物件版本 ID，則 S3 on Outposts 會新增刪除標記。S3 on Outposts 會如下處理刪除標記：
  - S3 on Outposts 預設不會複寫刪除標記。
  - 但是，您可以將刪除標記複寫新增至非標籤型規則。如需如何在複寫組態中啟用刪除標記複寫的詳細資訊，請參閱使用 S3 主控台。
- 如果您在 DELETE 請求中指定要刪除的物件版本 ID，S3 on Outposts 會永久刪除來源儲存貯體中的該物件版本。不過，不會在目的地儲存貯體中複寫刪除。換句話說，Amazon S3 不會從目的地儲存貯體中刪除相同的物件版本。此行為可防止資料遭到惡意刪除。

未複寫內容為何？

依預設，S3 on Outposts 不會複寫下列項目：

- 來源儲存貯體中由其他複寫規則所建立的物件複本。例如，若您設定複寫，其中儲存貯體 A 是來源，而儲存貯體 B 是目的地。現在，假設您新增另一個複寫組態，其中儲存貯體 B 是來源，而儲存貯體 C 是目的地。在此情況下，如果儲存貯體 B 中的物件是儲存貯體 A 中的物件複本，則不會複寫至儲存貯體 C。
- 來源儲存貯體中已複寫至不同目的地的物件。例如，如果您變更現有複寫組態中的目的地儲存貯體，則 S3 on Outposts 不會再次複寫這些物件。
- 在使用客戶所提供加密金鑰 (SSE-C) 進行伺服器端加密情況下建立的物件。
- 儲存貯體層級子資源的更新。

  例如，如果您變更來源儲存貯體上的生命週期組態，或將通知組態新增至來源儲存貯體，這些變更並不會套用至目的地儲存貯體。此功能可讓來源儲存貯體與目的地儲存貯體各有不同的組態。
- 生命週期組態執行的動作。

  例如，如果您只在來源儲存貯體上啟用生命週期組態，並設定到期動作，則 S3 on Outposts 會為過期物件建立刪除標記，但不會將這些標記複寫至目的地儲存貯體。如果您想要將相同的生命週期組態套用至來源與目的地儲存貯體，請在這兩個儲存貯體上啟用相同的生命週期組態。如需生命週期組態的詳細資訊，請參閱「管理儲存生命週期」。

S3 Replication on Outposts 不支援哪些項目？

S3 on Outposts 目前不支援下列 S3 複寫功能。

- S3 複寫時間控制 (S3 RTC)。不支援 S3 RTC，因為 S3 Replication on Outposts 中的物件流量會透過內部部署網路 (本機閘道) 傳輸。如需本機閘道的相關資訊，請參閱《AWS Outposts 使用者指南》中的使用本機閘道。
- 適用於批次操作的 S3 複寫。

設定複寫

> **Note**
>
> 您設定複寫之前就已存在於儲存貯體中的物件不會自動複寫。換句話說，Amazon S3 on Outposts 不會追溯複寫物件。若要複寫您在複寫組態之前建立的物件，您可使用 `CopyObject` API 操作以將其複製到相同的儲存貯體。複製物件之後，這些物件會在儲存貯體中顯示為「新」物件，且複寫組態將套用至這些物件。如需複製物件的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 使用 AWS SDK for Java 在 Amazon S3 on Outposts 儲存貯體中複製物件 和 CopyObject。

若要啟用 S3 Replication on Outposts，請將複寫規則新增至來源 Outposts 儲存貯體。複寫規則會通知 S3 on Outposts 依指定方式複寫物件。在複寫規則中，您必須提供以下項目：

- 來源 Outposts 儲存貯體存取點 — 您想要 S3 on Outposts 從其中複寫物件的存取點 Amazon Resource Name (ARN) 或存取點別名。如需使用存取點別名的詳細資訊，請參閱針對您的 S3 on Outposts 存取點使用儲存貯體樣式別名。
- 您要複寫的物件 – 您可以複寫來源 Outposts 儲存貯體中的所有物件或物件子集。您可以在組態中提供金鑰名稱前綴、一或多個物件標籤或兩者，來識別子集。

  例如，如果您將複寫規則設定為僅複寫具有金鑰名稱前綴為 `Tax/` 的物件，則 S3 on Outposts 會複寫具有 `Tax/doc1` 或 `Tax/doc2` 等金鑰的物件。但不會複寫具有 `Legal/doc3` 金鑰的物件。如果您指定字首以及一或多個標籤，則 S3 on Outposts 僅會複寫具備特定索引鍵字首和標籤的物件。
- 目的地 Outposts 儲存貯體 — 您想要 S3 on Outposts 複寫物件的儲存貯體 ARN 或存取點別名。

您可以使用 REST API、AWS SDK、AWS Command Line Interface (AWS CLI) 或 Amazon S3 主控台來設定複寫規則。

S3 on Outposts 也提供 API 操作來支援設定複寫規則。如需詳細資訊，請參閱 Amazon Simple Storage Service API 參考中的下列主題：

- PutBucketReplication
- GetBucketReplication
- DeleteBucketReplication

主題

- 建立複寫規則的先決條件
- 建立 Outposts 上的複寫規則

建立複寫規則的先決條件

主題

- 連線您的來源和目的地 Outpost 子網路
- 建立 IAM 角色

連線您的來源和目的地 Outpost 子網路

若要讓您的複寫流量透過本機閘道從來源 Outpost 傳送到目的地 Outpost，您必須新增新路由來設定網路。您必須將存取點的無類別域間路由 (CIDR) 網路範圍連接在一起。對於每對存取點，您只需要設定一次此連線。

根據與存取點相關聯 Outposts 端點的存取類型，設定連線的某些步驟會有所不同。端點的存取類型可以是私有 (AWS Outposts 的直接虛擬私有雲端 [VPC] 路由) 或客戶擁有的 IP (內部部署網路中客戶擁有的 IP 地址集區 [CoIP 集區])。

步驟 1：尋找來源 Outposts 端點的 CIDR 範圍

尋找與來源存取點相關聯來源端點的 CIDR 範圍

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 在 Outposts 儲存貯體清單中，選擇您要進行複寫的來源儲存貯體。
4. 選擇 Outposts 存取點索引標籤，然後針對複寫規則選擇來源儲存貯體的 Outposts 存取點。

5. 選擇 Outposts 端點。
6. 複製要在步驟 5 中使用的子網路 ID。
7. 您用來尋找來源 Outposts 端點 CIDR 範圍的方法取決於端點的存取類型。

   在 Outposts 端點概觀區段中，請參閱存取類型。

   - 如果存取類型為私有，請複製要在步驟 6 中使用的無類別域間路由 (CIDR) 值。
   - 如果存取類型為客戶擁有的 IP，請執行下列動作：
     1. 複製客戶擁有的 IPv4 集區值，以便稍後使用做為地址集區的 ID。
     2. 於 AWS Outpostshttps://console.aws.amazon.com/outposts/ 開啟 主控台。
     3. 在導覽窗格中，選擇本機閘道路由表。
     4. 選擇來源 Outpost 的本機閘道路由表 ID 值。
     5. 在詳細資訊窗格中，選擇 CoIP 集區索引標籤。將先前所複製 CoIP 集區 ID 的值貼到搜尋方塊中。
     6. 對於相符的 CoIP 集區，複製來源 Outposts 端點的對應 CIDR 值，以便在步驟 6 中使用。

步驟 2：尋找目的地 Outposts 端點的子網路 ID 和 CIDR 範圍

若要尋找與目的地存取點相關聯目的地端點的子網路 ID 和 CIDR 範圍，請遵循步驟 1 中的相同子步驟，並在套用這些子步驟時，將來源 Outposts 端點變更為目的地 Outposts 端點。複製目的地 Outposts 端點的子網路 ID 值，以便在步驟 6 中使用。複製目的地 Outposts 端點的 CIDR 值，以便在步驟 5 中使用。

步驟 3：尋找來源 Outpost 的本機閘道 ID

尋找來源 Outpost 的本機閘道 ID

1. 於 AWS Outpostshttps://console.aws.amazon.com/outposts/ 開啟 主控台。
2. 在左側導覽窗格中，選擇本機閘道。
3. 在本機閘道頁面上，尋找您要用於複寫來源 Outpost 的 Outpost ID。
4. 複製來源 Outpost 的本機閘道 ID 值，以便在步驟 5 中使用。

如需本機閘道的詳細資訊，請參閱《AWS Outposts 使用者指南》中的本機閘道。

步驟 4：尋找目的地 Outpost 的本機閘道 ID

若要尋找目的地 Outpost 的本機閘道 ID，請遵循步驟 3 中的相同子步驟，但尋找目的地 Outpost 的 Outpost ID 步驟除外。複製目的地 Outpost 的本機閘道 ID 值，以便在步驟 6 中使用。

步驟 5：設定從來源 Outpost 子網路到目的地 Outpost 子網路的連線

從來源 Outpost 子網路連線至目的地 Outpost 子網路

1. 登入 AWS Management Console，並在 https://console.aws.amazon.com/vpc/ 開啟 VPC 主控台。
2. 在左側導覽窗格中，選擇 Subnets (子網路)。
3. 在搜尋方塊中，輸入您在步驟 1 中所找到來源 Outposts 端點的子網路 ID。選擇具有相符子網路 ID 的子網路。
4. 對於相符的子網路項目，請選擇此子網路的路由表值。
5. 在具有所選路由表的頁面上，選擇動作，然後選擇編輯路由。
6. 在路由標籤中，選擇編輯路由。
7. 在目的地下，輸入您在步驟 2 中所找到目的地 Outposts 端點的 CIDR 範圍。
8. 在目標下，選擇 Outpost 本機閘道，然後輸入您在步驟 3 中所找到來源 Outpost 的本機閘道 ID。
9. 選擇 Save changes (儲存變更)。
10. 確定路由的狀態為作用中。

步驟 6：設定從目的地 Outpost 子網路到來源 Outpost 子網路的連線

1. 登入 AWS Management Console，並在 https://console.aws.amazon.com/vpc/ 開啟 VPC 主控台。
2. 在左側導覽窗格中，選擇 Subnets (子網路)。
3. 在搜尋方塊中，輸入您在步驟 2 中所找到目的地 Outposts 端點的子網路 ID。選擇具有相符子網路 ID 的子網路。
4. 對於相符的子網路項目，請選擇此子網路的路由表值。
5. 在具有所選路由表的頁面上，選擇動作，然後選擇編輯路由。
6. 在路由標籤中，選擇編輯路由。
7. 在目的地下，輸入您在步驟 1 中所找到來源 Outposts 端點的 CIDR 範圍。
8. 在目標下，選擇 Outpost 本機閘道，然後輸入您在步驟 4 中所找到目的地 Outpost 的本機閘道 ID。

9. 選擇 Save changes (儲存變更)。
10. 確定路由的狀態為作用中。

連接來源和目的地存取點的 CIDR 網路範圍之後，您必須建立 AWS Identity and Access Management (IAM) 角色。

建立 IAM 角色

根據預設，所有 S3 on Outposts 資源 (儲存貯體、物件與相關子資源) 皆為私有，且只有資源擁有者才可存取該資源。S3 on Outposts 需要從來源 Outposts 儲存貯體讀取和複寫物件的許可。您可建立 IAM 服務角色並在您的複寫組態中指定此角色以授予這些許可。

本節說明信任政策與最低必要許可政策。這些範例演練提供建立 IAM 角色的逐步說明。如需詳細資訊，請參閱 建立 Outposts 上的複寫規則。如需 IAM 角色的詳細資訊，請參閱 IAM 使用者指南中的 IAM 角色。

- 以下範例顯示信任政策，您可以在其中將 S3 on Outposts 識別為可擔任該角色的服務主體。

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

- 以下範例顯示存取政策，您可以在其中授予角色許可來代您執行複寫作業。當 S3 on Outposts 擔任該角色時，即具備您在此政策中指定的許可。若要使用此政策，請以您自己的資訊取代 *user input placeholders*。請務必將其取代為來源和目的地 Outposts 的 Outpost ID，以及來源和目的地 Outposts 儲存貯體的儲存貯體名稱和存取點名稱。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
```

```
         "Action":[
            "s3-outposts:GetObjectVersionForReplication",
            "s3-outposts:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/
bucket/SOURCE-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/
accesspoint/SOURCE-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:ReplicateObject",
            "s3-outposts:ReplicateDelete"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/
bucket/DESTINATION-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/
accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      }
   ]
}
```

存取政策會授予下列動作的許可：

- `s3-outposts:GetObjectVersionForReplication` — 對所有物件授予此動作的許可，以允許 S3 on Outposts 取得與每個物件相關聯的特定物件版本。

- `s3-outposts:GetObjectVersionTagging` – *SOURCE-OUTPOSTS-BUCKET* 儲存貯體 (來源儲存貯體) 中物件上這個動作的許可，其允許 S3 on Outposts 讀取要複寫的物件標籤。如需詳細資訊，請參閱 新增 S3 on Outposts 儲存貯體的標籤。如果 S3 on Outposts 不具備這些許可，則會複寫物件，但不會複寫物件標籤。

- `s3-outposts:ReplicateObject` 與 `s3-outposts:ReplicateDelete` – *DESTINATION-OUTPOSTS-BUCKET* 儲存貯體 (目的地儲存貯體) 中所有物件上這些動作的許可，其允許 S3 on Outposts 將物件或刪除標記複寫至目的地 Outposts 儲存貯體。如需刪除標記的資訊，請參閱 刪除操作對複寫的影響。

> **Note**
>
> - *DESTINATION-OUTPOSTS-BUCKET* 儲存貯體 (目的地儲存貯體) 上 `s3-outposts:ReplicateObject` 動作的許可也允許複寫物件標籤。因此，您不需要明確授予 `s3-outposts:ReplicateTags` 動作的許可。
> - 對於跨帳戶複寫，目的地 Outposts 儲存貯體的擁有者必須更新其儲存貯體政策，以授予對 *DESTINATION-OUTPOSTS-BUCKET* 的 `s3-outposts:ReplicateObject` 動作許可。`s3-outposts:ReplicateObject` 動作可讓 S3 on Outposts 將物件和物件標籤複寫到目的地 Outposts 儲存貯體。

如需 S3 on Outposts 動作的清單，請參閱 S3 on Outposts 定義的動作。

> **Important**
>
> 擁有 IAM 角色的 AWS 帳戶 必須具備授予 IAM 角色的動作許可。
> 例如，假設來源 Outposts 儲存貯體包含另一個 AWS 帳戶 所擁有的物件。物件擁有者必須透過儲存貯體政策和存取點政策，將必要的許可明確授予擁有 IAM 角色的 AWS 帳戶。否則，S3 on Outposts 就無法存取這些物件，而導致物件的複寫失敗。
> 此處描述的許可與基本複寫組態相關。如果您選擇新增額外的複寫組態，則必須將額外許可授予給 S3 on Outposts。

在來源與目的地 Outposts 儲存貯體分屬於不同 AWS 帳戶 時授予許可

當來源和目的地 Outposts 儲存貯體不屬於相同帳戶時，目的地 Outposts 儲存貯體的擁有者必須更新目的地儲存貯體的儲存貯體和存取點政策。這些政策必須對來源 Outposts 儲存貯體和 IAM 服務角色的擁有者授予執行複寫動作的許可，如同下列政策範例所示，若未授予，複寫則會失敗。在這些政策範例中，*DESTINATION-OUTPOSTS-BUCKET* 是目的地儲存貯體。若要使用這些政策範例，請以您自己的資訊取代 *user input placeholders*。

如果您要手動建立 IAM 服務角色，請將角色路徑設定為 `role/service-role/`，如下列政策範例所示。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM ARN。

```
{
   "Version":"2012-10-17",
   "Id":"PolicyForDestinationBucket",
   "Statement":[
```

```
      {
         "Sid":"Permissions on objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":[
            "s3-outposts:ReplicateDelete",
            "s3-outposts:ReplicateObject"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:DestinationBucket-account-
ID:outpost/DESTINATION-OUTPOST-ID/bucket/DESTINATION-OUTPOSTS-BUCKET/object/*"
         ]
      }

   ]
}
```

```
{
"Version":"2012-10-17",
   "Id":"PolicyForDestinationAccessPoint",
   "Statement":[
      {
         "Sid":"Permissions on objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":[
            "s3-outposts:ReplicateDelete",
            "s3-outposts:ReplicateObject"
         ],
         "Resource" :[
            "arn:aws:s3-outposts:region:DestinationBucket-account-
ID:outpost/DESTINATION-OUTPOST-ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/
object/*"
         ]
      }
   ]
}
```

> **Note**
>
> 如果來源 Outposts 儲存貯體中的物件已標記，請注意下列情況：
> 如果來源 Outposts 儲存貯體擁有者將 `s3-outposts:GetObjectVersionTagging` 與 `s3-outposts:ReplicateTags` 動作的許可授予 S3 on Outposts 來複寫物件標籤 (透過 IAM 角色)，Amazon S3 會連同物件一起複寫標籤。如需 IAM 角色的資訊，請參閱 建立 IAM 角色。

建立 Outposts 上的複寫規則

S3 Replication on Outposts 是跨相同或不同 AWS Outposts 的儲存貯體自動非同步的物件複寫。複寫會將來源 Outposts 儲存貯體中新建立的物件和物件更新複製至目的地 Outposts 儲存貯體。如需詳細資訊，請參閱 複寫 S3 on Outposts 的物件。

> **Note**
>
> 不會複寫您設定複寫規則之前就已存在於來源 Outposts 儲存貯體中的物件。換句話說，S3 on Outposts 不會追溯複寫物件。若要複寫您在複寫組態之前建立的物件，您可使用 `CopyObject` API 操作以將其複製到相同的儲存貯體。複製物件之後，這些物件會在儲存貯體中顯示為「新」物件，且複寫組態將套用至這些物件。如需複製物件的詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的 使用 AWS SDK for Java 在 Amazon S3 on Outposts 儲存貯體中複製物件 和 CopyObject。

設定複寫時，會將複寫規則新增至來源 Outposts 儲存貯體。複寫規則會定義要複寫的來源 Outposts 儲存貯體物件，以及存放已複寫物件的目的地 Outposts 儲存貯體。您可以建立規則，以特定的金鑰名稱前綴、一或多個物件標籤、或兩種都用，複寫儲存貯體中的所有物件，或一部分的物件。目的地 Outposts 儲存貯體可以在與來源 Outposts 儲存貯體相同的 Outpost 中，也可以在不同的 Outpost 中。

對於 S3 on Outposts 複寫規則，您必須同時提供來源 Outposts 儲存貯體的存取點 Amazon Resource Name (ARN) 和目的地 Outposts 儲存貯體的存取點 ARN，而不是來源和目的地 Outposts 儲存貯體名稱。

如果您指定要刪除的物件版本 ID，S3 on Outposts 會刪除來源 Outposts 儲存貯體中的該物件版本。但不會在目的地 Outposts 儲存貯體中進行刪除。換句話說，它不會從目的地 Outposts 儲存貯體中刪除相同的物件版本。此行為可防止資料遭到惡意刪除。

當您將複寫規則新增至 Outposts 儲存貯體時，預設會啟用此規則，讓規則在您儲存之後立即運作。

在此範例中，您會設定來源與目的地 Outposts 儲存貯體為不同 Outposts 且同一 AWS 帳戶 所擁有的複寫。提供了使用 Amazon S3 主控台、AWS Command Line Interface (AWS CLI)、AWS SDK for Java 和 AWS SDK for .NET 的範例。如需跨帳戶 S3 on Outposts 許可的相關資訊，請參閱 在來源與目的地 Outposts 儲存貯體分屬於不同 AWS 帳戶 時授予許可。

如需設定 S3 on Outposts 複寫規則的先決條件，請參閱 建立複寫規則的先決條件。

使用 S3 主控台

當目的地 Amazon S3 on Outposts 儲存貯體與來源 Outposts 儲存貯體位於不同的 Outpost 時，請依照這些步驟設定複寫規則。

如果目的地 Outposts 儲存貯體位在與來源 Outposts 儲存貯體不同的帳戶中，您必須將儲存貯體政策新增至目的地 Outposts 儲存貯體，以將複寫目的地 Outposts 儲存貯體中物件的許可授予來源 Outposts 儲存貯體帳戶擁有者。如需詳細資訊，請參閱 當來源值區和目的地值區屬於不同時授與權限 AWS 帳戶。

建立複寫規則

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在 Outposts 儲存貯體清單中，選擇您要使用做為來源儲存貯體的儲存貯體名稱。
3. 選擇管理索引標籤，向下捲動至複寫規則區段，然後選擇建立複寫規則。
4. 對於複寫規則名稱，輸入規則名稱，以利之後識別此規則。此名稱為必要，且在儲存貯體內必須是唯一的。
5. 在狀態下，預設會選擇已啟用。已啟用規則在您儲存它之後就會立即運作。如果希望稍後再啟用此規則，請選擇已停用。
6. 在優先順序下，如果發生規則重疊，則規則的優先順序值會決定要套用的規則。當物件包含在多個複寫規則的範圍內時，S3 on Outposts 會使用這些優先順序值來避免衝突。依預設，新規則會以最高優先順序新增至複製組態。數字愈高，優先順序愈高。

   若要變更規則的優先順序，請在儲存規則之後，從複製規則清單中選擇規則名稱、選擇動作，然後選擇編輯優先順序。
7. 在來源儲存貯體之下，您有下列選項可用來設定複寫來源：

   - 若要複寫整個儲存貯體，請選擇套用至儲存貯體中的所有物件。
   - 若要將字首或標籤篩選套用至複寫來源，請選擇使用一或多個篩選條件限制此規則的範圍。您可以合併字首與標籤。

- 若要複寫具有相同字首的所有物件，請在字首下的方塊中輸入字首。使用字首篩選條件以限制複寫名稱以相同字串 (例如，`pictures`) 開頭的所有物件。

  如果您輸入的字首是資料夾名稱，您必須使用 / (正斜線) 作為最後一個字元 (例如，`pictures/`)。

- 若要複寫具有一個或多個相同物件標籤的所有物件，請選擇新增標籤，然後在方塊中輸入鍵/值對。若要新增另一個索引標籤，請重複此程序，。如需物件標籤的詳細資訊，請參閱 新增 S3 on Outposts 儲存貯體的標籤。

8. 若要存取 S3 on Outposts 來源儲存貯體以進行複寫，請在來源存取點名稱下選擇連接至來源儲存貯體的存取點。
9. 在目的地下，選擇您想要 S3 on Outposts 在其中複寫物件的目的地 Outposts 儲存貯體的存取點 ARN。目的地 Outposts 儲存貯體可以位於與來源 Outposts 儲存貯體相同或不同的 AWS 帳戶。

   如果目的地儲存貯體位在與來源 Outposts 儲存貯體不同的帳戶中，您必須將儲存貯體政策新增至目的地 Outposts 儲存貯體，以將複寫目的地 Outposts 儲存貯體中物件的許可授予來源 Outposts 儲存貯體帳戶擁有者。如需詳細資訊，請參閱 在來源與目的地 Outposts 儲存貯體分屬於不同 AWS 帳戶 時授予許可。

   > **Note**
   >
   > 如果未在目的地 Outposts 儲存貯體上啟用版本控制，您會收到包含啟用版本控制按鈕的警告訊息。選擇此按鈕，以在儲存貯體上啟用版本控制。

10. 設定 S3 on Outposts 可擔任並代您複寫物件的 AWS Identity and Access Management (IAM) 服務角色。

    若要設定 IAM 角色，請在 IAM 角色下，執行下列其中一個動作：

    - 若要讓 S3 on Outposts 為您的複寫組態建立新的 IAM 角色，請選擇從現有的 IAM 角色中選擇，然後選擇建立新角色。當您儲存規則時，系統會為符合所選擇來源與目的地 Outposts 儲存貯體的 IAM 角色產生新原則。建議您選擇建立新角色。
    - 您也可以選擇使用現有 IAM 角色。如果這麼做，則必須選擇將必要複寫許可授予 S3 on Outposts 的角色。如果此角色未依您的複寫規則授予 S3 on Outposts 足夠的許可，則複寫會失敗。

      若要選擇現有角色，請選擇從現有 IAM 角色中選擇，然後從下拉式功能表中選擇角色。您也可以選擇輸入 IAM 角色 ARN，然後輸入 IAM 角色的 Amazon Resource Name (ARN)。

> **Important**
>
> 當您新增複寫規則至 S3 on Outposts 儲存貯體時，必須擁有 `iam:CreateRole` 和 `iam:PassRole` 許可，才能建立和傳遞授予 S3 on Outposts 複寫許可的 IAM 角色。如需詳細資訊，請參閱《IAM 使用者指南》中授予使用者將角色傳遞至 AWS 服務 的許可。

11. 依預設，Outposts 儲存貯體中的所有物件都會加密。如需 S3 on Outposts 加密的詳細資訊，請參閱 S3 on Outposts 中的資料加密。只有使用 Amazon S3 管理金鑰的伺服器端加密 (SSE-S3) 進行加密的物件才可加以複寫。不支援使用 AWS Key Management Service (AWS KMS) 金鑰的伺服器端加密 (SSE-KMS) 或使用經由客戶提供加密金鑰的伺服器端加密 (SSE-C) 進行加密的物件複寫。
12. 設定複寫規則組態時，視需要啟用下列其他選項：
    - 如果您想要在複寫組態中啟用 S3 on Outposts 複寫指標，請選取複寫指標。如需詳細資訊，請參閱 使用複寫指標監控進度。
    - 如果您想要在複寫組態中啟用刪除標記複寫，請選取 Delete marker replication (刪除標記複寫)。如需詳細資訊，請參閱 刪除操作對複寫的影響。
    - 如果您要將對複本所做的中繼資料變更複寫回來源物件，請選取複本修改同步。如需詳細資訊，請參閱 啟用 Outposts 上的 Amazon S3 複本修改同步時的複寫狀態。
13. 若要完成，請選擇建立規則。

儲存規則之後，即可編輯、啟用、停用或刪除規則。若要這麼做，請前往來源 Outposts 儲存貯體的管理索引標籤，向下捲動至複寫規則區段，選擇您的規則，然後選擇編輯規則。

使用 AWS CLI

若要在來源與目的地 Outposts 儲存貯體由相同 AWS 帳戶 擁有時使用 AWS CLI 設定複寫，請執行以下操作：

- 建立來源與目的地 Outposts 儲存貯體。
- 在兩個儲存貯體上啟用版本控制。
- 建立 IAM 角色，授予 S3 on Outposts 複寫物件的許可。
- 將複寫組態新增至來源 Outposts 儲存貯體。

您可以測試以驗證設定。

設定相同 AWS 帳戶 擁有來源與目的地 Outposts 儲存貯體時的複寫

1. 設定 AWS CLI 的憑證描述檔。在此範例中，我們使用描述檔名稱 `acctA`。如需設定憑證描述檔的相關資訊，請參閱《AWS Command Line Interface 使用者指南》中的具名描述檔。

   > **Important**
   >
   > 用於此練習的描述檔必須有必要的許可。例如，您可以在複寫組態中指定 S3 on Outposts 可以擔任的 IAM 服務角色。只有當您所用的描述檔有 `iam:CreateRole` 和 `iam:PassRole` 許可時才可執行此作業。如需詳細資訊，請參閱《IIAM 使用者指南》《中授予使用者將角色傳遞至 AWS 服務 的許可。如果您使用管理員憑證建立具名描述檔，具名描述檔將擁有執行所有任務的必要許可。

2. 建立 *source* 儲存貯體並對它啟用版本控制。下列 `create-bucket` 命令會在美國東部 (維吉尼亞北部) (`us-east-1`) 區域中建立 *SOURCE-OUTPOSTS-BUCKET* 儲存貯體。若要使用此命令，請以您自己的資訊取代 *user input placeholders*。

   ```
   aws s3control create-bucket --bucket SOURCE-OUTPOSTS-BUCKET --outpost-id SOURCE-OUTPOST-ID --profile acctA --region us-east-1
   ```

   下列 `put-bucket-versioning` 命令啟用 *SOURCE-OUTPOSTS-BUCKET* 儲存貯體上的版本控制。若要使用此命令，請以您自己的資訊取代 *user input placeholders*。

   ```
   aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/bucket/SOURCE-OUTPOSTS-BUCKET --versioning-configuration Status=Enabled --profile acctA
   ```

3. 建立 *destination* 儲存貯體並對它啟用版本控制。下列 `create-bucket` 命令會在美國西部 (奧勒岡) (`us-west-2`) 區域中建立 *DESTINATION-OUTPOSTS-BUCKET* 儲存貯體。若要使用此命令，請以您自己的資訊取代 *user input placeholders*。

   > **Note**
   >
   > 若要在來源與目的地 Outposts 儲存貯體都位於同一個 AWS 帳戶 時設定複寫組態，您可以使用同一個具名描述檔。此範例使用 `acctA`。若要在儲存貯體屬於不同 AWS 帳戶 時測試複寫組態，則需為每個儲存貯體指定不同的描述檔。

```
aws s3control create-bucket --bucket DESTINATION-OUTPOSTS-BUCKET --create-bucket-
configuration LocationConstraint=us-west-2 --outpost-id DESTINATION-OUTPOST-ID --
profile acctA --region us-west-2
```

下列 put-bucket-versioning 命令啟用 *DESTINATION-OUTPOSTS-BUCKET* 儲存貯體上的版本控制。若要使用此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/bucket/DESTINATION-
OUTPOSTS-BUCKET --versioning-configuration Status=Enabled --profile acctA
```

4. 建立 IAM 服務角色。稍後在複寫組態中，您會將此服務角色新增至 *SOURCE-OUTPOSTS-BUCKET* 儲存貯體。S3 on Outposts 就會擔任此角色以代您複寫物件。建立 IAM 角色需要兩個步驟：

   a. 建立 IAM 角色。

      i. 複製下列信任政策，並將它儲存至本機電腦目前目錄下名稱為 s3-on-outposts-role-trust-policy.json 的檔案中。此政策會授予 S3 on Outposts 服務主體擔任該服務角色的許可。

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

      ii. 執行下列 命令以建立角色。以您自己的資訊取代 *user input placeholders*。

```
aws iam create-role --role-name replicationRole --assume-role-policy-
document file://s3-on-outposts-role-trust-policy.json --profile acctA
```

   b. 將許可政策連接到服務角色。

i. 複製下列許可政策，並將它儲存至本機電腦目前目錄中名為 s3-on-outposts-role-permissions-policy.json 的檔案。此政策會授予各種 S3 on Outposts 儲存貯體與物件動作的許可。若要使用此政策，請以您自己的資訊取代 *user input placeholders*。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:GetObjectVersionForReplication",
            "s3-outposts:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/bucket/SOURCE-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/accesspoint/SOURCE-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:ReplicateObject",
            "s3-outposts:ReplicateDelete"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/bucket/DESTINATION-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      }
   ]
}
```

ii. 執行下列命令以建立政策，並將它連接至角色。以您自己的資訊取代 *user input placeholders*。

```
aws iam put-role-policy --role-name replicationRole --policy-
document file://s3-on-outposts-role-permissions-policy.json --policy-
name replicationRolePolicy --profile acctA
```

5. 將複寫組態新增至 *SOURCE-OUTPOSTS-BUCKET* 儲存貯體。

   a. 雖然 S3 on Outposts API 要求複寫組態必須為 XML 格式，但 AWS CLI 會要求您指定 JSON 格式的複寫組態。將下列 JSON 儲存至您電腦本機目錄下的 replication.json 檔案中。若要使用此組態，請以您自己的資訊取代 *user input placeholders*。

```
{
  "Role": "IAM-role-ARN",
  "Rules": [
    {
      "Status": "Enabled",
      "Priority": 1,
      "DeleteMarkerReplication": { "Status": "Disabled" },
      "Filter" : { "Prefix": "Tax"},
      "Destination": {
        "Bucket":
        "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-
ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT"
      }
    }
  ]
}
```

   b. 執行下列 put-bucket-replication 命令，將複寫組態新增至您的來源 Outposts 儲存貯體。若要使用此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control put-bucket-replication --account-id 123456789012 --
bucket arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-
ID/bucket/SOURCE-OUTPOSTS-BUCKET --replication-configuration file://
replication.json --profile acctA
```

   c. 使用 get-bucket-replication 命令來擷取複寫組態。若要使用此命令，請以您自己的資訊取代 *user input placeholders*。

```
aws s3control get-bucket-replication --account-id 123456789012 --bucket
 arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/
bucket/SOURCE-OUTPOSTS-BUCKET --profile acctA
```

6. 在 Amazon S3 主控台中測試設定：

   a. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。

   b. 在 *SOURCE-OUTPOSTS-BUCKET* 儲存貯體中，建立名為 Tax 的資料夾。

   c. 將範例物件新增至 *SOURCE-OUTPOSTS-BUCKET* 儲存貯體的 Tax 資料夾。

   d. 在 *DESTINATION-OUTPOSTS-BUCKET* 儲存貯體中驗證下列事項：

      - S3 on Outposts 已複寫物件。

        > **Note**
        >
        > S3 on Outposts 複寫物件所需的時間長短取決於物件大小。如需如何查看複寫狀態的相關資訊，請參閱 取得複寫狀態資訊。

      - 在物件屬性標籤中，複寫狀態已設定為複寫 (將此識別為複本物件)。

管理複寫

本節描述 S3 on Outposts 中可用的其他複寫組態選項、如何判斷複寫狀態，以及如何疑難排解複寫。如需核心複寫組態的資訊，請參閱 設定複寫。

主題

- 使用複寫指標監控進度
- 取得複寫狀態資訊
- 故障排除複寫
- 針對 Outposts 上的 S3 複寫使用 EventBridge

使用複寫指標監控進度

Outposts 上的 S3 複寫提供了複寫組態中複寫規則的詳細指標。透過追蹤擱置複寫的位元組、複寫延遲和作業擱置，您可以每 5 分鐘以複寫指標，監控複寫進度。您也可以設定 Amazon EventBridge 以接收複寫失敗通知，協助疑難排解任何組態問題。

啟用複寫指標後，Outposts 上的 S3 複寫會將下列指標發佈至 Amazon CloudWatch：

- 擱置複寫的位元組 — 針對指定的複寫規則，擱置複寫的物件位元組總數。

- 複寫延遲 — 針對指定的複寫規則，複寫目的地儲存貯體位於來源儲存貯體後方的秒數上限。
- 擱置複寫的作業 – 針對指定的複寫規則，擱置複寫的作業數量。作業包含物件、刪除標記和標籤。

> **Note**
>
> Outposts 上的 S3 複寫指標的計費方式與 CloudWatch 自訂指標相同。如需詳細資訊，請參閱 CloudWatch 定價。

取得複寫狀態資訊

複寫狀態可協助您判斷要由 Amazon S3 on Outposts 複寫之物件的目前狀態。來源物件的複寫狀態將會傳回 `PENDING`、`COMPLETED` 或 `FAILED`。複本的複寫狀態將會傳回 `REPLICA`。

複寫狀態概觀

在複寫中，您有一個來源儲存貯體，您可以在其上設定複寫與 S3 on Outposts 複寫物件的目的地儲存貯體。當您從這些儲存貯體請求物件 (使用 `GetObject`) 或物件中繼資料 (使用 `HeadObject`) 時，S3 on Outposts 會在回應中傳回 `x-amz-replication-status` 標頭，如下所示：

- 當您從來源儲存貯體請求物件時，如果請求中的物件符合複寫資格，S3 on Outposts 即會傳回 `x-amz-replication-status` 標頭。

  例如，假設您在複寫組態中指定物件前綴 `TaxDocs`，告知 S3 on Outposts 複寫僅具有金鑰名稱前綴 `TaxDocs` 的物件。系統會複寫您上傳且具有此金鑰名稱前綴 (例如 `TaxDocs/document1.pdf`) 的任何物件。針對具有此金鑰名稱前綴的物件請求，S3 on Outposts 會傳回 `x-amz-replication-status` 標頭，以及代表物件複寫狀態的下列其中一個值：`PENDING`、`COMPLETED` 或 `FAILED`。

  > **Note**
  >
  > 若物件複寫在您上傳完物件後失敗，則您無法重試複寫。您必須再次上傳物件。對於缺少複寫角色許可或儲存貯體許可等問題，物件會轉換成 `FAILED` 狀態。對於暫時性錯誤，例如，如果儲存貯體或 Outpost 無法使用，複寫狀態不會轉換成 `FAILED`，但會保持 `PENDING`。資源恢復線上狀態後，S3 on Outposts 會繼續複寫這些物件。

- 當您從目的地儲存貯體請求物件時，如果您請求中的物件是 S3 on Outposts 建立的複本，S3 on Outposts 會傳回值為 `REPLICA` 的 `x-amz-replication-status` 標頭。

> **Note**
>
> 在從已啟用複寫的來源儲存貯體中刪除物件之前，您應該先檢查物件的複寫狀態，確保已複寫物件。

啟用 Outposts 上的 Amazon S3 複本修改同步時的複寫狀態

當您的複寫規則啟用 S3 on Outposts 複本修改同步時，複本可以報告 `REPLICA` 以外的狀態。如果中繼資料變更正在複寫過程中，則複寫 `x-amz-replication-status` 標頭會傳回 `PENDING`。如果複本修改同步無法複寫中繼資料，則複寫標頭會傳回 `FAILED`。如果中繼資料正確複寫，複寫標頭會傳回值 `REPLICA`。

故障排除複寫

在您設定複寫之後，如果物件複本未出現在目的地 Amazon S3 on Outposts 儲存貯體中，請使用這些故障診斷技巧以識別並修正問題。

- S3 on Outposts 複寫物件所需的時間長短取決於幾個因素，包括來源和目的地 Outposts 的距離，以及物件的大小。

  您可以檢查來源物件的複寫狀態。如果物件複寫狀態為 `PENDING`，表示 S3 on Outposts 尚未完成複寫。如果物件複寫狀態為 `FAILED`，請檢查來源儲存貯體中所設的複寫組態。
- 在來源儲存貯體中的複寫組態中，驗證下列項目：
  - 目的地儲存貯體的存取點 Amazon Resource Name (ARN) 正確。
  - 金鑰名稱前綴正確。例如，如果您設定組態來複寫具有前綴 `Tax` 的物件，則只會複寫具有 `Tax/document1` 或 `Tax/document2` 等金鑰名稱的物件。不會複寫具有金鑰名稱 `document3` 的物件。
  - 狀態為 `Enabled`。
- 確認在任何儲存貯體上均沒有已暫停的版本控制。來源與目的地儲存貯體都必須啟用版本控制。
- 如果目的地儲存貯體的擁有者是另一個 AWS 帳戶，請確認儲存貯體擁有者在目的地儲存貯體上有儲存貯體政策，以允許來源儲存貯體擁有者複寫物件。如需範例，請參閱 在來源與目的地 Outposts 儲存貯體分屬於不同 AWS 帳戶 時授予許可。
- 如果目的地儲存貯體中未出現物件複本，可能是下列問題阻礙了複寫作業：
  - 如果來源儲存貯體中的物件是由另一個複寫組態所建立的複本，則 S3 on Outposts 不會複寫該複本。例如，如果您將複寫組態從儲存貯體 A 設定到儲存貯體 B，再設定到儲存貯體 C，則 S3 on Outposts 不會將儲存貯體 B 中的物件複本複寫至儲存貯體 C。

如果您想要將儲存貯體 A 的物件複製到儲存貯體 B 和儲存貯體 C，請在不同的複寫規則中，為來源儲存貯體複寫組態設定多個儲存貯體目的地。例如，在來源儲存貯體 A 上建立兩個複寫規則，其中一個規則可複寫到目的地儲存貯體 B，另一個規則複寫到目的地儲存貯體 C。

- 來源儲存貯體擁有者可授予其他 AWS 帳戶 上傳物件的許可。根據預設，來源儲存貯體擁有者不具其他帳戶所建立之物件的任何許可。複寫組態只會複寫來源儲存貯體擁有者具備存取許可的物件。為了預防複寫問題發生，來源儲存貯體擁有者可以授予其他 AWS 帳戶 許可來有條件地建立物件，並要求這些物件的明確存取許可。如需政策範例，請參閱「授予跨帳戶許可，以在確保儲存貯體擁有者具有完全控制時上傳物件」。

- 假設您在複寫組態中新增一個規則，以複寫含特定標籤的物件子集。在此情況下，您必須於建立物件時指派特定標籤金鑰與值，以便 S3 on Outposts 複寫物件。如果您先建立物件，之後才將標籤新增至現有物件，S3 on Outposts 就不會複寫該物件。
- 如果儲存貯體政策拒絕存取下列任何動作的複寫角色，則複寫會失敗：

  來源儲存貯體：

  ```
  "s3-outposts:GetObjectVersionForReplication",
  "s3-outposts:GetObjectVersionTagging"
  ```

  目的地儲存貯體：

  ```
  "s3-outposts:ReplicateObject",
  "s3-outposts:ReplicateDelete",
  "s3-outposts:ReplicateTags"
  ```

- 當物件未複寫至其目的地 Outposts 時，Amazon EventBridge 可以通知您。如需詳細資訊，請參閱「針對 Outposts 上的 S3 複寫使用 EventBridge」。

針對 Outposts 上的 S3 複寫使用 EventBridge

Amazon S3 on Outposts 會與 Amazon EventBridge 整合，並使用 `s3-outposts` 命名空間。EventBridge 是無伺服器事件匯流排服務，可讓您用於將應用程式與來自各種來源的資料互相連線。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的什麼是 Amazon EventBridge？。

您也可以設定 Amazon EventBridge 以接收複寫失敗事件通知，協助疑難排解任何複寫組態問題。當物件未複寫至其目的地 Outposts 時，EventBridge 可以在執行個體中通知您。如需詳細了解複寫物件的目前狀態，請參閱 複寫狀態概觀。

每當 Outposts 儲存貯體發生事件，S3 on Outposts 就會將事件傳送至 EventBridge。與其他目的地不同，您不需要選取想要傳遞的事件類型。您也可以使用 EventBridge 規則將事件路由至其他目標。啟用 EventBridge 後，S3 on Outposts 會將下列所有事件傳送至 EventBridge。

事件類型	描述	命名空間
`Operation FailedReplication`	複寫規則內的物件複寫失敗。如需詳細了解 Outposts 上的 S3 複寫失敗原因，請參閱 使用 EventBridge 檢視 Outposts 上的 S3 複寫失敗原因。	`s3-outposts`

使用 EventBridge 檢視 Outposts 上的 S3 複寫失敗原因

下表列出 Outposts 上的 S3 複寫失敗原因。您可以透過 Amazon Simple Queue Service (Amazon SQS)、Amazon Simple Notification Service (Amazon SNS)、AWS Lambda 或 Amazon CloudWatch Logs，設定 EventBridge 規則以發佈和檢視失敗原因。如需詳細了解針對 EventBridge 使用這些資源的必要權限，請參閱針對 EventBridge 使用資源型政策。

複寫失敗原因	描述
`AssumeRoleNotPermitted`	S3 on Outposts 無法擔任複寫組態中指定的 AWS Identity and Access Management (IAM) 角色。
`DstBucketNotFound`	S3 on Outposts 找不到複寫組態中指定的目的地儲存貯體。
`DstBucketUnversioned`	Outposts 目的地儲存貯體上未啟用版本控制。若要以 Outposts 上的 S3 複寫來複寫物件，您必須啟用目的地儲存貯體上的版本控制。
`DstDelObjNotPermitted`	S3 on Outposts 無法將刪除項目複寫到目的地儲存貯體。可能缺少目的地儲存貯體的 `s3-outposts:ReplicateDelete` 許可。
`DstMultipartCompleteNotPermitted`	S3 on Outposts 無法完成目的地儲存貯體中物件的分段上傳。可能缺少目的地儲存貯體的

複寫失敗原因	描述
	s3-outposts:ReplicateObject 許可。
DstMultipartInitNotPermitted	S3 on Outposts 無法起始目的地儲存貯體中物件的分段上傳。可能缺少目的地儲存貯體的 s3-outposts:ReplicateObject 許可。
DstMultipartPartUploadNotPermitted	S3 on Outposts 無法在目的地儲存貯體上傳分段物件。可能缺少目的地儲存貯體的 s3-outposts:ReplicateObject 許可。
DstOutOfCapacity	S3 on Outposts 無法複寫到目的地 Outpost，因為 Outpost 不在 S3 儲存容量中。
DstPutObjNotPermitted	S3 on Outposts 無法將物件複寫到目的地儲存貯體。可能缺少目的地儲存貯體的 s3-outposts:ReplicateObject 許可。
DstPutTaggingNotPermitted	S3 on Outposts 無法將物件標籤複寫到目的地儲存貯體。可能缺少目的地儲存貯體的 s3-outposts:ReplicateObject 許可。
DstVersionNotFound	S3 on Outposts 無法在目的地儲存貯體中找到所需的物件版本，以複寫該物件版本的中繼資料。
SrcBucketReplicationConfigMissing	S3 on Outposts 找不到與來源 Outposts 儲存貯體相關聯的存取點複寫組態。
SrcGetObjNotPermitted	S3 on Outposts 無法存取來源儲存貯體中的物件以進行複寫。可能缺少來源儲存貯體的 s3-outposts:GetObjectVersionForReplication 許可。

複寫失敗原因	描述
SrcGetTaggingNotPermitted	S3 on Outposts 無法從來源儲存貯體存取物件標籤資訊。可能缺少來源儲存貯體的 s3-outposts:GetObjectVersionTagging 許可。
SrcHeadObjectNotPermitted	S3 on Outposts 無法從來源儲存貯體擷取物件中繼資料。可能缺少來源儲存貯體的 s3-outposts:GetObjectVersionForReplication 許可。
SrcObjectNotEligible	物件不符合複寫資格。物件或物件標籤不符合複寫組態。

如需詳細了解複寫疑難排解，請參閱下列主題：

- 建立 IAM 角色
- 故障排除複寫

以 CloudWatch 監控 EventBridge

Amazon EventBridge 與 Amazon CloudWatch 整合用以進行監控。EventBridge 會每分鐘自動將指標傳送至 CloudWatch。這些指標包括已符合規則的事件數量，以及規則叫用目標的次數。在 EventBridge 中執行規則時，與該規則關聯的所有目標都會受到叫用。您可採取下列方式，透過 CloudWatch 監控 EventBridge 行為。

- 您可以從 CloudWatch 儀表板，監控 EventBridge 規則的可用 EventBridge 指標。然後，您可以使用 CloudWatch 的功能 (例如 CloudWatch 警示) 在特定指標上設定警示。如果這些指標達到您在警示中指定的自訂閾值，您就會收到通知，且可採取相應動作。
- 您可以將 Amazon CloudWatch Logs 設定為 EventBridge 規則的目標。接著，EventBridge 會建立日誌串流，而 CloudWatch Logs 會將事件中的文字儲存為日誌項目。如需詳細資訊，請參閱 EventBridge 和 CloudWatch Logs。

如需詳細了解偵錯 EventBridge 事件傳遞和封存事件，請參閱下列主題：

- 事件重試政策和使用無效字母佇列

- 封存 EventBridge 事件

通過使用在 Outposts 上共享 S3 AWS RAM

Outposts 上的 Amazon S3 支援使用 AWS Resource Access Manager (AWS RAM) 在組織內的多個帳戶之間共用 S3 容量。透過 S3 on Outposts 共用，您可以允許其他人在您的 Outpost 上建立和管理儲存貯體、端點和存取點。

本主題示範如 AWS RAM 何使用 AWS 帳戶 在 Outposts 上與 AWS 組織中的其他人共用 S3 以及相關資源。

必要條件

- Outpost 擁有者帳戶已在 AWS Organizations中設定組織。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的建立組織。
- 該組織包括您想要在 Outposts 容量上與之 AWS 帳戶 共享 S3 的組織。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的傳送邀請給 AWS 帳戶。
- 選取您要共用的下列選項之一。必須選取第二個資源 (子網路或 Outposts)，才能存取端點。端點是要存取存放在 S3 on Outposts 上的資料時的一項網路要求。

選項 1	選項 2
S3 on Outposts 允許使用者在 Outposts 和存取點上建立儲存貯體，並將物件新增到這些儲存貯體。	S3 on Outposts 允許使用者在 Outposts 和存取點上建立儲存貯體，並將物件新增到這些儲存貯體。
子網 允許使用者使用您的 Virtual Private Cloud (VPC) 以及與您的子網路關聯的端點。	Outposts 允許使用者查看 S3 容量圖表和 AWS Outposts 主控台首頁。也允許使用者在共用 Outposts 上建立子網路並建立端點。

程序

1. 使用擁有前哨的 AWS 帳戶 登入，然後在 https://console.aws.amazon.com/ram 開啟主 AWS RAM 控台。 AWS Management Console

2. 請確定您已在中啟用 AWS Organizations 共用功能 AWS RAM。如需詳細資訊，請參閱《AWS RAM 使用者指南》中的在 AWS Organizations中啟用資源共用。
3. 使用先決條件中的選項 1 或選項 2 建立資源共享。如果您有多個 S3 on Outposts 資源，請選取要共用的資源的 Amazon Resource Name (ARN)。若要啟用端點，請共用您的子網路或 Outpost。

   如需如何建立資源共用的資訊，請參閱《AWS RAM 使用者指南》中的建立資源共用。
4. 您與 AWS 帳戶 之共享資源的現在應該可以在 Outposts 上使用 S3。根據您在先決條件中選取的選項，請向帳戶使用者提供下列資訊：

選項 1	選項 2
Outpost ID	Outpost ID
VPC ID	
子網路 ID	
安全群組 ID	

> **Note**
>
> 使用者可以使用 AWS RAM 主控台 AWS Command Line Interface (AWS CLI)、 AWS SDK 或 REST API 來確認資源已與他們共用。使用者可以使用 get-resource-sharesCLI 命令來檢視其現有的資源共用。

使用範例

在您與另一個帳戶共用您的 S3 on Outposts 資源後，該帳戶可以管理您的 Outpost 上的儲存貯體和物件。如果您共用 Subnets (子網路) 資源，那麼該帳戶可以使用您建立的端點。下列範例會示範使用者在您共用這些資源之後，如何使用與您的 Outpost 互動。 AWS CLI

Example ：建立儲存貯體

下列範例會在前哨上建立名為*##-s3-* 儲存區 1 的值區。*op-01ac5d28a6a232904*在使用此命令之前，請將每個 *user input placeholder* 取代為適合您的使用案例的值。

```
aws s3control create-bucket --bucket example-s3-bucket1 --outpost-
id op-01ac5d28a6a232904
```

如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 create-bucket。

Example ：建立存取點

下列範例會使用下表中的範例參數，在 Outpost 上建立存取點。使用此指令之前，請先將這些*user input placeholder*值和程 AWS 區域 式碼取代為您的使用案例的適當值。

Parameter (參數)	Value
帳戶 ID	*111122223333*
存取點名稱	*example-outpost-access-point*
Outpost ID	*op-01ac5d28a6a232904*
Outpost 儲存貯體名稱	*example-s3-bucket1*
VPC ID	*vpc-1a2b3c4d5e6f7g8h9*

> **Note**
>
> 「帳戶 ID」參數必須是值區擁有者 (即共用使用者) 的 AWS 帳戶 ID。

```
aws s3control create-access-point --account-id 111122223333 --name example-outpost-
access-point \
--bucket arn:aws:s3-outposts:us-east-1:111122223333:outpost/op-01ac5d28a6a232904/
bucket/example-s3-bucket1 \
--vpc-configuration VpcId=vpc-1a2b3c4d5e6f7g8h9
```

若要取得有關此指令的更多資訊，請參閱AWS CLI 參考create-access-point中的〈〉。

Example ：上傳物件

下列範例會透過由 AWS 帳戶 *111122223333* 擁有之 Outpost *op-01ac5d28a6a232904* 上的存取點 *example-outpost-access-point*，從使用者的本機檔案系統上傳檔案 *my_image.jpg* 到名為

images/my_image.jpg 的物件。使用此指令之前，請先將這些*user input placeholder*值和程 AWS 區域 式碼取代為您的使用案例的適當值。

```
aws s3api put-object --bucket arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/example-outpost-access-
point \
--body my_image.jpg --key images/my_image.jpg
```

如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 put-object。

> **Note**
>
> 如果此操作導致 Resource not found (找不到資源) 錯誤或無回應，則您的 VPC 可能沒有共用端點。
> 若要檢查是否有共用端點，請使用list-shared-endpoints AWS CLI 指令。如果沒有共享端點，請與 Outpost 擁有者合作建立一個端點。如需詳細資訊，請參閱 Amazon 簡易儲存服務 API 參考ListSharedEndpoints中的。

Example ：建立端點

下列範例在共用 Outpost 上建立端點。在使用此命令之前，請將 Outpost ID、子網路 ID 和安全群組 ID 的 *user input placeholder* 值取代為適合您的使用案例的值。

> **Note**
>
> 使用者只有在資源共用包含 Outposts 資源時，才能執行此操作。

```
aws s3outposts create-endpoint --outposts-id op-01ac5d28a6a232904 --subnet-id XXXXXX --
security-group-id XXXXXXX
```

如需此命令的詳細資訊，請參閱 AWS CLI 參考中的 create-endpoint。

其他使用 S3 on Outposts 的 AWS 服務

其他在 AWS Outposts 本機上執行的 AWS 服務，也可以使用 Amazon S3 on Outposts 容量。在 Amazon CloudWatch 中，`S3Outposts` 命名空間顯示 S3 on Outposts 內儲存貯體指標的詳細資訊，

但這些指標不包括其他 AWS 服務 的使用量。若要管理由其他 AWS 服務 使用的 S3 on Outposts 容量，請參閱下列表格中的資訊。

AWS 服務	描述	進一步了解
Simple Storage Service (Amazon S3)	所有 S3 on Outposts 使用量都有匹配的帳戶和儲存貯體 CloudWatch 指標。	請參閱指標
Amazon Elastic Block Store (Amazon EBS)	對於 Outposts 上的 Amazon EBS，您可以選擇 AWS Outpost 作為快照目標，並存放在本地 S3 on Outpost 中。	進一步了解
Amazon Relationa l Database Service (Amazon RDS)	您可以使用 Amazon RDS 本地備份將 RDS 備份存放在本地 Outpost 上。	進一步了解

監控 S3 on Outposts

使用 Outposts 上的 Amazon S3，您可以在 Outposts 上建立 S3 儲存貯體，並輕鬆地在現場部署存放和擷取物件，以供需要本機資料存取、本機資料處理和資料存放區的應用程式使用。 AWS Outposts 上的 S3 提供新的儲存類別 S3 Outposts (0UTP0STS)，該類別使用 Amazon S3 API，其設計用於在您的多個裝置和伺服器上持久且冗餘地存放資料。 AWS Outposts您可以使用存取點和透過 Virtual Private Cloud (VPC) 的端點連線，與您的 Outpost 儲存貯體進行通訊。就像在 Amazon S3 儲存貯體一樣，您在 Outpost 儲存貯體上可以使用同樣的 API 和功能，包括存取政策、加密和標記。您可以透過 AWS Management Console、 AWS Command Line Interface (AWS CLI)、 AWS 開發套件或 REST API 在 Outposts 上使用 S3。如需更多資訊，請參閱什麼是 Amazon S3 on Outposts？

如需監控 Amazon S3 on Outposts 儲存貯體容量的詳細資訊，請參閱下列主題。

主題

- 使用 Amazon 指標管理 Outposts 容量上的 S3 CloudWatch
- 使用 Amazon CloudWatch 活動在 Outposts 事件通知上接收 S3
- 使用 AWS CloudTrail 日誌監控 S3 on Outposts

使用 Amazon 指標管理 Outposts 容量上的 S3 CloudWatch

為了協助管理 Outpost 上的固定 S3 容量，建議您建立 CloudWatch 警示，以便在儲存使用率超過特定閾值時通知您。如需 Outposts 上 S3 指 CloudWatch 標的詳細資訊，請參閱CloudWatch 度量。如果無足夠的空間可以在您的 Outpost 上存放物件，API 會傳回容量不足例外狀況 (ICE)。若要釋放空間，您可以建立觸發明確資料刪除的 CloudWatch 警示，或使用生命週期到期原則將物件過期。若要在刪除之前儲存資料，您可 AWS DataSync 以使用將資料從 Outposts 儲存貯體上的 Amazon S3 複製到. AWS 區域若要取得有關使用的更多資訊 DataSync，請參閱《使AWS DataSync 用指南》 AWS DataSync中的〈入門〉。

CloudWatch 度量

`S3Outposts` 命名空間包含下列 Amazon S3 on Outposts 儲存貯體指標。您可以監控佈建的 S3 on Outposts 位元組總數、物件可用的位元組總數，以及特定儲存貯體的所有物件大小總計。所有直接 S3 用量都存在儲存貯體或帳戶相關指標。間接 S3 用量 (例如在 Outpost 上存放 Amazon Elastic Block Store 本機快照或 Amazon Relational Database Service 備份) 會耗用 S3 容量，但不包含在儲存貯體或帳戶相關指標中。如需 Amazon EBS 本機快照的詳細資訊，請參閱 Outposts 上的 Amazon EBS 本機快照。若要查看您的 Amazon EBS 成本報告，請造訪 https://console.aws.amazon.com/billing/。

> **Note**
>
> S3 on Outposts 僅支援下列指標，而不支援其他 Amazon S3 指標。
> 由於 Outposts 上的 S3 有固定的容量限制，因此建議您建立 CloudWatch 警示，以在儲存使用率超過特定閾值時通知您。

指標	描述	時段	單位	Type
`Outpost` `talByte`	Outpost 的佈建容量總計 (位元組)。	5 分鐘	位元組	S3 on Outposts
`Outpostl` `eeBytes`	存放客戶資料的 Outpost 可用位元組計數。	5 分鐘	位元組	S3 on Outposts
`BucketU` `dBytes`	指定儲存貯體的所有物件總大小。	5 分鐘	位元組	S3 on Outposts。僅限直接 S3 用量。

指標	描述	時段	單位	Type
AccountUnencryptedBytes	指定 Outposts 帳戶的所有物件大小總計。	5 分鐘	位元組	S3 on Outposts。僅限直接 S3 用量。
BytesPendingReplication	針對指定複寫規則等待複寫的物件位元組總數。如需如何啟用複寫指標的詳細資訊，請參閱在 Outposts 之間建立複寫規則。	5 分鐘	位元組	選用。適用於 S3 Replication on Outposts。
OperationsPendingReplication	針對指定複寫規則等待複寫的操作總數。如需如何啟用複寫指標的詳細資訊，請參閱在 Outposts 之間建立複寫規則。	5 分鐘	計數	選用。適用於 S3 Replication on Outposts。
ReplicationLatency	針對指定的複寫規則，複寫目的地儲存貯體落後來源儲存貯體的目前延遲秒數。如需如何啟用複寫指標的詳細資訊，請參閱在 Outposts 之間建立複寫規則。	5 分鐘	秒鐘	選用。適用於 S3 Replication on Outposts。

使用 Amazon CloudWatch 活動在 Outposts 事件通知上接收 S3

您可以使用 CloudWatch 事件為 Outposts API 事件上的任何 Amazon S3 建立規則。建立規則時，您可以選擇透過所有支援的 CloudWatch 目標接收通知，包括 Amazon Simple Queue Service (Amazon SQS)、Amazon Simple Notification Service (Amazon SNS) 和 AWS Lambda。如需詳細資訊，請參閱 Amazon CloudWatch 事件使用者指南中可作為 CloudWatch 事件目標的AWS 服務清單。若要選擇要在 Outposts 上與 S3 搭配使用的目標服務，請參閱使用 Amazon CloudWatch 事件使用者指南中的建立 AWS CloudTrail在 AWS API 呼叫時觸發的 CloudWatch 事件規則。

> **Note**
>
> 對於 Outposts 物件操作的 S3，只有當您設定用來接收這些事件的追蹤 (選擇性地使用事件選取器) 時，傳送的 AWS API 呼叫事件才 CloudTrail 會符合您的規則。若要取得更多資訊，請參閱《使用指南》中的〈AWS CloudTrail 使用 CloudTrail 記錄檔〉。

Example

以下是 DeleteObject 操作的範例規則。若要使用此範例規則，請使用 S3 on Outposts 儲存貯體的名稱取代 *example-s3-bucket1*。

```
{
  "source": [
    "aws.s3-outposts"
  ],
  "detail-type": [
    "AWS API call through CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "s3-outposts.amazonaws.com"
    ],
    "eventName": [
      "DeleteObject"
    ],
    "requestParameters": {
      "bucketName": [
        "example-s3-bucket1"
      ]
    }
  }
}
```

使用 AWS CloudTrail 日誌監控 S3 on Outposts

Outposts 上的 Amazon S3 整合了這項服務 AWS CloudTrail，該服務可提供使用者、角色或在 Outposts AWS 服務 中 S3 中所採取的動作記錄。您可以使用 AWS CloudTrail 來取得 S3 on Outposts 儲存貯體層級和物件層級請求的相關資訊，以稽核和記錄 S3 on Outposts 事件活動。若要為所有 Outposts 值區或特定 Outposts 值區清單啟用 CloudTrail 資料事件，您必須在中手動建立追蹤。CloudTrail如需有關 CloudTrail 記錄檔項目的詳細資訊，請參閱 Outposts 日誌檔項目上的 S3。

> **Note**
>
> - 最佳做法是為 AWS CloudTrail 資料事件 Outposts 值區建立生命週期政策。設定生命週期政策，在您需要稽核日誌檔的時段之後，定期移除這些日誌檔。這麼做可降低 Amazon Athena 針對每個查詢分析的資料量。如需詳細資訊，請參閱 在值區上設定生命週期組態。

- 如需如何查詢 CloudTrail 日誌的範例，請參閱AWS 大數據部落格文章使用和 Amazon Athena 分析安全性、合規 AWS CloudTrail 和操作活動。

在 Outposts 儲存貯體上為 S3 中的物件啟用 CloudTrail 記錄功能

您可以使用 Amazon S3 主控台設定 AWS CloudTrail 追蹤，以記錄 Outposts 儲存貯體上 Amazon S3 中物件的資料事件。 CloudTrail 支援在 Outposts 物件層級 API 操作上記錄 S3，例如`GetObject`、`DeleteObject`、和。`PutObject`這些事件稱為資料事件。

依預設， CloudTrail 追蹤不會記錄資料事件。不過，您可以設定追蹤來記錄所指定之 S3 on Outposts 儲存貯體的資料事件，或記錄 AWS 帳戶中所有 S3 on Outposts 儲存貯體的資料事件。如需詳細資訊，請參閱 使用記錄 Amazon S3 API 呼叫 AWS CloudTrail。

CloudTrail 不會在 CloudTrail 事件歷史記錄中填入資料事件。此外，並非 Outposts 儲存貯體層級 API 作業上的所有 S3 都會填入事件歷史記錄中。 CloudTrail 如需如何查詢 CloudTrail 日誌的詳細資訊，請參閱使用 Amazon CloudWatch 日誌篩選模式和 Amazon Athena 在 AWS 知識中心查詢 CloudTrail 日誌。

若要設定追蹤來記錄 S3 on Outposts 儲存貯體的資料事件，您可以使用 AWS CloudTrail 主控台或 Amazon S3 主控台。如果您要設定追蹤記錄 Outposts 儲存貯體上所有 S3 的資料事件 AWS 帳戶，則使用 CloudTrail 主控台會更容易。如需使用 CloudTrail 主控台設定追蹤以在 Outposts 資料事件上記錄 S3 的相關資訊，請參閱使用AWS CloudTrail 者指南中的資料事件。

Important

資料事件需支付額外的費用。如需詳細資訊，請參閱 AWS CloudTrail 定價。

下列程序顯示如何使用 Amazon S3 主控台設定 CloudTrail 追蹤，以記錄 Outposts 儲存貯體上 S3 的資料事件。

Note

建立儲存貯體 AWS 帳戶 的使用者擁有該儲存貯體，並且是唯一可以在 Outposts 資料事件上設定 S3 要傳送到 AWS CloudTrail的儲存貯體。

為 Outposts 儲存貯體上 S3 中的物件啟用 CloudTrail 資料事件記錄

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Outposts buckets (Outposts 儲存貯體)。
3. 選擇您要使 CloudTrail用記錄其資料事件的 Outposts 值區的名稱。
4. 選擇 Properties (屬性)。
5. 在「資AWS CloudTrail 料事件」區段中，選擇「設定於」 CloudTrail。

   AWS CloudTrail 主控台隨即開啟。

   您可以建立新 CloudTrail 追蹤或重複使用現有追蹤，並在 Outposts 上設定 S3 資料事件以記錄在追蹤中。
6. 在 CloudTrail 主控台 [儀表板] 頁面上，選擇 [建立追蹤]
7. 在步驟 1 選擇追蹤屬性頁面上，提供追蹤的名稱、選擇 S3 儲存貯體來存放追蹤記錄、指定您想要的任何其他設定，然後選擇下一步。
8. 在步驟 2 選擇日誌事件頁面的事件類型下，選擇資料事件。

   針對資料事件類型，選擇 S3 Outposts。選擇下一步。

> **Note**
>
> - 當您建立追蹤，並針對 S3 on Outposts 並設定資料事件記錄時，必須正確地指定資料事件類型。
>   - 如果您使用主 CloudTrail 控台，請為資料事件類型選擇 S3 Outposts。如需有關如何在 CloudTrail 主控台中建立追蹤的資訊，請參閱《AWS CloudTrail 使用指南》中的使用主控台建立和更新追蹤。如需如何在 CloudTrail主控台中設定 S3 on Outposts 資料事件記錄的相關資訊，請參閱AWS CloudTrail 使用者指南中的記錄 Amazon S3 物件的資料事件。
>   - 如果您使用 AWS Command Line Interface (AWS CLI) 或 AWS SDK，請將`resources.type`欄位設定為`AWS::S3Outposts::Object`。如需有關如何在 Outposts 資料事件上記錄 S3 的詳細資訊 AWS CLI，請參閱AWS CloudTrail 使用者指南中的 Outposts 事件上的記錄 S3。
> - 如果您使用 CloudTrail 主控台或 Amazon S3 主控台設定追蹤以記錄 Outposts 儲存貯體上 S3 的資料事件，Amazon S3 主控台會顯示儲存貯體已啟用物件層級日誌記錄。

9. 在步驟 3 檢閱並建立頁面上，檢閱您設定的追蹤屬性和日誌事件。然後，選擇建立追蹤。

若要在 Outposts 儲存貯體上停用 S3 中物件的 CloudTrail 資料事件記錄

1. 請登入 AWS Management Console 並開啟 CloudTrail 主控台，網址為 https://console.aws.amazon.com/cloudtrail/。
2. 在左側導覽窗格中，選擇追蹤。
3. 選擇您已建立來記錄 S3 on Outposts 儲存貯體之事件的追蹤名稱。
4. 在追蹤的詳細資訊窗格上，選擇右上角的停止記錄。
5. 在出現的對話方塊中，選擇停止記錄。

使用 Amazon S3 on Outposts 進行開發

透過 Amazon S3 on Outposts，您可以在 AWS Outposts 上建立 S3 儲存貯體，並針對需要本機資料存取權限、本機資料處理和資料駐留的應用程式，在內部部署輕鬆存放和擷取物件。S3 on Outposts 提供一個全新的儲存類別，即 S3 Outposts (OUTPOSTS)，它使用 Amazon S3 API，目的是在您的 AWS Outposts Outposts 上的多個裝置和伺服器上以持久、備援的方式存放資料。您可以使用存取點和透過 Virtual Private Cloud (VPC) 的端點連線，與您的 Outpost 儲存貯體進行通訊。就像在 Amazon S3 儲存貯體一樣，您在 Outpost 儲存貯體上可以使用同樣的 API 和功能，包括存取政策、加密和標記。您可以透過 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。如需詳細資訊，請參閱 什麼是 Amazon S3 on Outposts？。

下列主題提供使用 S3 on Outposts 進行開發的資訊

主題

- Amazon S3 on Outposts API 操作
- 使用適用於 Java 的開發套件設定 S3 on Outposts 的 S3 控制用戶端
- 通過 IPv6 向 Outposts 上的 S3 發出請求

Amazon S3 on Outposts API 操作

本主題列出了您可以與 Amazon S3 on Outposts 搭配使用的 Amazon S3、Amazon S3 Control 和 Amazon S3 on Outposts API 操作。

主題

- 適用於管理物件的 Amazon S3 API 操作
- 適用於管理儲存貯體的 Amazon S3 Control API 操作
- 適用於管理 Outposts 的 S3 on Outposts API 操作

適用於管理物件的 Amazon S3 API 操作

S3 on Outposts 設計為與 Amazon S3 一樣使用相同的物件 API 操作。您必須使用存取點來存取 Outpost 儲存貯體中的任何物件。當搭配 S3 on Outposts 使用物件 API 操作時，您可以提供 Outposts 存取點 Amazon Resource Name (ARN) 或存取點別名。如需存取點別名的詳細資訊，請參閱 針對您的 S3 on Outposts 儲存貯體存取點使用儲存貯體樣式別名。

Amazon S3 on Outposts 支援下列 Amazon S3 API 操作：

- AbortMultipartUpload
- CompleteMultipartUpload
- CopyObject
- CreateMultipartUpload
- DeleteObject
- DeleteObjects
- DeleteObjectTagging
- GetObject
- GetObjectTagging
- HeadBucket
- HeadObject
- ListMultipartUploads
- ListObjects
- ListObjectsV2
- ListObjectVersions
- ListParts
- PutObject
- PutObjectTagging
- UploadPart

- UploadPartCopy

適用於管理儲存貯體的 Amazon S3 Control API 操作

S3 on Outposts 支援下列適用於儲存貯體的 Amazon S3 Control API 操作。

- CreateAccessPoint
- CreateBucket
- DeleteAccessPoint
- DeleteAccessPointPolicy
- DeleteBucket
- DeleteBucketLifecycleConfiguration
- DeleteBucketPolicy
- DeleteBucketReplication
- DeleteBucketTagging
- GetAccessPoint
- GetAccessPointPolicy
- GetBucket
- GetBucketLifecycleConfiguration
- GetBucketPolicy
- GetBucketReplication
- GetBucketTagging
- GetBucketVersioning
- ListAccessPoints
- ListRegionalBuckets
- PutAccessPointPolicy
- PutBucketLifecycleConfiguration
- PutBucketPolicy
- PutBucketReplication
- PutBucketTagging

- PutBucketVersioning

適用於管理 Outposts 的 S3 on Outposts API 操作

S3 on Outposts 支援下列適用於管理端點的 Amazon S3 on Outposts API 操作。

- CreateEndpoint
- DeleteEndpoint
- ListEndpoints
- ListOutpostsWithS3
- ListSharedEndpoints

使用適用於 Java 的開發套件設定 S3 on Outposts 的 S3 控制用戶端

下列範例使用 AWS SDK for Java 為 Amazon S3 on Outposts 設定 Amazon S3 控制用戶端。若要使用此範例，請以您自己的資訊取代每個 *user input placeholder*。

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;

public AWSS3Control createS3ControlClient() {

    String accessKey = AWSAccessKey;
    String secretKey = SecretAccessKey;
    BasicAWSCredentials awsCreds = new BasicAWSCredentials(accessKey, secretKey);

    return AWSS3ControlClient.builder().enableUseArnRegion()
            .withCredentials(new AWSStaticCredentialsProvider(awsCreds))
            .build();

}
```

通過 IPv6 向 Outposts 上的 S3 發出請求

Outposts 上的 Amazon S3 和 Outposts 雙堆疊端點上的 S3 支援使用 IPv6 或 IPv4 通訊協定對 Outposts 儲存貯體上的 S3 的請求。使用 Outposts S3 的 IPv6 支援，您可以透過 IPv6 網路上的 Outposts API 透過 S3 存取和操作儲存貯體和控制平面資源。

> **Note**
>
> IPv6 網路不支援 Outposts 物件動作 (例如PutObject或GetObject) 上的 S3。

透過 IPv6 網路存取 Outposts 上的 S3 不會產生額外費用。有關 Outposts S3 的詳細資訊，請參閱 Outposts 定價上的 S3。

主題

- IPv6 入門
- 使用雙堆疊端點透過 IPv6 網路提出要求
- 在 IAM 原則中使用 IPv6 地址
- 測試 IP 地址相容性
- 搭配使用 IPv6 AWS PrivateLink
- 在 Outposts 雙堆疊端點上使用 S3

IPv6 入門

若要透過 IPv6 向 Outposts 儲存貯體上的 S3 發出請求，您必須使用雙堆疊端點。下節說明如何使用雙堆疊端點透過 IPv6 提出請求。

在嘗試通過 IPv6 訪問 Outposts 存儲桶上的 S3 之前，以下是重要的考慮因素：

- 存取儲存貯體的用戶端與網路必須啟用才能使用 IPv6。
- 虛擬託管式與路徑式請求都支援 IPv6 存取。如需詳細資訊，請參閱 在 Outposts 雙堆疊端點上使用 S3。
- 如果您在 AWS Identity and Access Management (IAM) 使用者中使用來源 IP 位址篩選，或在 Outposts 儲存貯體政策上使用 S3，則必須更新政策以包含 IPv6 位址範圍。

  > **Note**
  >
  > 此要求僅適用於 Outposts 儲存貯體作業上的 S3，以及跨 IPv6 網路的控制平面資源。IPv6 網路不支援在 Outposts 上的 Amazon S3 物件動作。

- 使用 IPv6 時，伺服器會存取 IPv6 格式的日誌檔案輸出 IP 地址。您必須更新用來剖析 Outposts 日誌檔案上 S3 的現有工具、指令碼和軟體，以便它們能夠剖析 IPv6 格式化的遠端 IP 位址。更新後的工具、指令碼和軟體將會正確剖析 IPv6 格式化的遠端 IP 位址。

使用雙堆疊端點透過 IPv6 網路提出要求

若要透過 IPv6 在 Outposts API 呼叫上使用 S3 發出請求，您可以透過AWS CLI或 AWS SDK 使用雙堆疊端點。無論您是透過 IPv6 通訊協定還是 IPv4 通訊協定存取 Outposts 上的 S3，Amazon S3 控制 API 操作和 Outposts 上的 S3 API 操作都以相同的方式運作。不過，請注意，不支援 IPv6 網路上的 Outposts 物件動作 (例如`PutObject`或`GetObject`) 上的 S3。

使用 AWS Command Line Interface (AWS CLI) 與 AWS 開發套件時，您可使用參數或標記以變更為雙堆疊端點。您也可以直接將雙堆疊端點指定為組態檔案中 Outposts 端點上 S3 的覆寫。

您可以使用雙堆疊端點從以下任何一種方式透過 IPv6 存取 Outposts 儲存貯體上的 S3：

- AWS CLI，請參閱從 AWS CLI 使用雙堆疊端點。
- AWS 開發套件，請參閱「在來自開發套件的 Outposts 雙堆疊端點上使用 S3 AWS」。

在 IAM 原則中使用 IPv6 地址

在嘗試使用 IPv6 通訊協定存取 Outposts 儲存貯體上的 S3 之前，請確定用於 IP 位址篩選的 Outposts 儲存貯體上的 IAM 使用者或 S3 政策已更新為包含 IPv6 位址範圍。如果未更新 IP 位址篩選政策以處理 IPv6 位址，您可能會在嘗試使用 IPv6 通訊協定時失去對 Outposts 儲存貯體上 S3 的存取權。

篩選 IP 位址的 IAM 政策會使用 IP 位址條件運算子。Outposts 儲存貯體上的下列 S3 使用 IP 位址條件運算子，識別允許 IPv4 位址的 54.240.143.* IP 範圍。任何超出此範圍的 IP 地址都將被拒絕訪問 Outposts 存儲桶（`DOC-EXAMPLE-BUCKET`）上的 S3。因為所有的 IPv6 地址都在允許的範圍外，所以此原則可避免 IPv6 地址得以存取 `DOC-EXAMPLE-BUCKET`。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3outposts:*",
      "Resource": "arn:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
bucket/DOC-EXAMPLE-BUCKET/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
```

```
  ]
}
```

您可以修改 Outposts 儲存貯體政策Condition元素上的 S3，以允許 IPv4 (54.240.143.0/24) 和 IPv6 (2001:DB8:1234:5678::/64) 位址範圍，如下列範例所示。您可使用本例所示的同類型 Condition 區塊，更新您的 IAM 使用者與儲存貯體原則。

```
       "Condition": {
         "IpAddress": {
            "aws:SourceIp": [
              "54.240.143.0/24",
               "2001:DB8:1234:5678::/64"
             ]
          }
        }
```

使用 IPv6 之前，您必須更新所有相關的 IAM 使用者與儲存貯體原則，它們會使用 IP 地址篩選條件允許 IPv6 地址範圍。除現有的 IPv4 地址範圍外，建議您也更新 IAM 原則的貴組織 IPv6 地址範圍。如需允許透過 IPv6 與 IPv4 存取的儲存貯體原則範例，請參閱限制特定 IP 地址的存取。

您可以使用 https://console.aws.amazon.com/iam/的 IAM 主控台來檢閱 IAM 使用者原則。如需 IAM 的詳細資訊，請參閱《IAM 使用者指南》。如需在 Outposts 儲存貯體政策上編輯 S3 的相關資訊，請參閱新增或編輯 Amazon S3 on Outposts 儲存貯體的儲存貯體政策。

測試 IP 地址相容性

如果您使用的是 Linux 或 Unix 執行個體，或 macOS X 平台，則可以測試您透過 IPv6 對雙堆疊端點的存取。例如，若要在 Outposts 端點上透過 IPv6 測試與 Amazon S3 的連線，請使用以下dig命令：

```
dig s3-outposts.us-west-2.api.aws AAAA +short
```

如果已正確設定透過 IPv6 網路的雙堆疊端點，dig命令會傳回連接的 IPv6 位址。例如：

```
dig s3-outposts.us-west-2.api.aws AAAA +short

2600:1f14:2588:4800:b3a9:1460:159f:ebce

2600:1f14:2588:4802:6df6:c1fd:ef8a:fc76

2600:1f14:2588:4801:d802:8ccf:4e04:817
```

搭配使用 IPv6 AWS PrivateLink

Outposts 上的 S3 支援AWS PrivateLink服務和端點的 IPv6 通訊協定。透過 IPv6 通訊協定的AWS PrivateLink支援，您可以從內部部署或其他私人連線，透過 IPv6 網路連線到 VPC 內的服務端點。Outposts 上 S3 的 AWS PrivateLink IPv6 支援也可讓您AWS PrivateLink與雙堆疊端點整合。如需如何啟用 IPv6 的步驟AWS PrivateLink，請參閱透過AWS PrivateLink服務和端點加快 IPv6 採用速度。

> **Note**
>
> 若要將支援的 IP 位址類型從 IPv4 更新為 IPv6，請參閱《AWS PrivateLink使用手冊》中的「修改支援的 IP 位址類型」。

搭配使用 IPv6 AWS PrivateLink

如果您使AWS PrivateLink用 IPv6，則必須建立 IPv6 或雙堆疊 VPC 介面端點。有關如何使用建立 VPC 端點的一般步驟AWS Management Console，請參閱使用指南中的使用介面 VPC 端點存取 AWSAWS PrivateLink服務。

AWS Management Console

使用下列程序建立連線至 Outposts 上 S3 的介面 VPC 端點。

1. 登入 AWS Management Console，並在 https://console.aws.amazon.com/vpc/ 開啟 VPC 主控台。
2. 在導覽窗格中選擇 Endpoints (端點)。
3. 選擇 建立端點。
4. 在 Service category (服務類別) 中，選擇 AWS services。
5. 對於服務名稱，選擇 Outposts 服務上的 S3（我們東部-1.3-前哨）。
6. 對於 VPC，選擇您將在 Outposts 上存取 S3 的 VPC。
7. 對於子網路，請為每個可用區域選擇一個子網路，您將在 Outposts 上存取 S3。您無法在相同的可用區域內選取多個子網路。針對您選取的每個子網路，都會建立新的端點網路介面。根據預設，子網路 IP 位址範圍中的 IP 位址會指派給端點網路介面。若要指定端點網路介面的 IP 位址，請選擇「指定 IP 位址」，然後輸入子網路位址範圍中的 IPv6 位址。
8. 對於 IP 位址類型，請選擇雙堆疊。將 IPv4 和 IPv6 位址指派給您的端點網路介面。只有當所有選取的子網都具有 IPv4 和 IPv6 地址範圍時，才支援此選項。

9. 對於安全群組，請選擇要與 VPC 端點的端點網路介面關聯的安全群組。依預設，預設安全群組與 VPC 相關聯。
10. 針對政策，選擇完整存取，以允許 VPC 端點上所有資源的所有主體進行所有操作。否則，請選擇 [自訂] 附加 VPC 端點原則，該策略可控制主體在 VPC 端點上對資源執行動作的權限。只有服務支援 VPC 端點政策時，此選項才可用。如需詳細資訊，請參閱端點策略。
11. (選用) 若要新增標籤，請選擇 Add new tag (新增標籤)，然後輸入標籤的鍵和值。
12. 選擇 建立端點。

Example — Outposts 儲存貯體政策上的 S3

若要允許 Outposts 上的 S3 與您的 VPC 端點互動，您可以如下所示更新 Outposts 上的 S3 政策：

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3-outposts:*",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

AWS CLI

> **Note**
>
> 若要在虛擬私人雲端端點上啟用 IPv6 網路，您必須為 Outposts 上 S3 的`SupportedIpAddressType`篩選器`IPv6`設定。

下列範例會使用指`create-vpc-endpoint`令來建立新的雙堆疊介面端點。

```
aws ec2 create-vpc-endpoint \
--vpc-id vpc-12345678 \
--vpc-endpoint-type Interface \
--service-name com.amazonaws.us-east-1.s3-outposts \
--subnet-id subnet-12345678 \
--security-group-id sg-12345678 \
--ip-address-type dualstack \
```

```
--dns-options "DnsRecordIpType=dualstack"
```

視AWS PrivateLink服務組態而定，VPC 端點服務提供者可能需要接受新建立的端點連線，才能使用這些連線。如需詳細資訊，請參閱AWS PrivateLink使用指南中的接受和拒絕端點連線要求。

下列範例會使用指`modify-vpc-endpoint`令將僅限 IPV 的 VPC 端點更新為雙堆疊端點。雙堆疊端點允許存取 IPv4 和 IPv6 網路。

```
aws ec2 modify-vpc-endpoint \
--vpc-endpoint-id vpce-12345678 \
--add-subnet-ids subnet-12345678 \
--remove-subnet-ids subnet-12345678 \
--ip-address-type dualstack \
--dns-options "DnsRecordIpType=dualstack"
```

如需有關如何啟用 IPv6 網路的詳細資訊AWS PrivateLink，請參閱透過AWS PrivateLink服務和端點加快 IPv6 採用。

在 Outposts 雙堆疊端點上使用 S3

Outposts 上的 S3 雙堆疊端點支援透過 IPv6 和 IPv4 向 Outposts 儲存貯體上的 S3 請求。本節說明如何在 Outposts 雙堆疊端點上使用 S3。

主題

- Outposts 雙堆疊端點上的 S3
- 從 AWS CLI 使用雙堆疊端點
- 在來自開發套件的 Outposts 雙堆疊端點上使用 S3 AWS

Outposts 雙堆疊端點上的 S3

當您向雙堆疊端點發出要求時，Outposts 儲存貯體上的 S3 會解析為 IPv6 或 IPv4 位址。如需透過 IPv6 存取 Outposts 儲存貯體上 S3 的詳細資訊，請參閱通過 IPv6 向 Outposts 上的 S3 發出請求。

若要透過雙堆疊端點存取 Outposts 儲存貯體上的 S3，請使用路徑樣式端點名稱。Outposts 上的 S3 僅支援地區雙堆疊端點名稱，這表示您必須將區域指定為名稱的一部分。

對於雙堆疊路徑樣式 FIP 端點，請使用下列命名慣例：

```
s3-outposts-fips.region.api.aws
```

對於雙堆疊非 FIP 端點，請使用下列命名慣例：

```
s3-outposts.region.api.aws
```

> **Note**
>
> Outposts 上的 S3 不支援虛擬託管式端點名稱。

從 AWS CLI 使用雙堆疊端點

本節提供可用於要求雙堆疊端點的 AWS CLI 命令範例。如需設定 AWS CLI 的說明，請參閱使用 AWS CLI 和 SDK for Java 開始使用。

您可以在檔案中的設定AWS Config檔`true`中`use_dualstack_endpoint`將組態值設定為，以將`s3`和`s3api`AWS CLI命令發出的所有 Amazon S3 請求導向指定區域的雙堆疊端點。您可以在組態檔案中或使用`--region`選項的指令中指定 Region。

搭配使用雙堆疊端點時AWS CLI，僅支援`path`定址樣式。在配置文件中設置的地址樣式會決定存儲桶名稱是在主機名稱還是在 URL 中。如需詳細資訊，請參閱《AWS CLI 使用者指南》中的 s3outposts。

若要透過使用雙堆疊端點AWS CLI，請將`--endpoint-url`參數與`http://s3.dualstack.region.amazonaws.com`或`https://s3-outposts-fips.region.api.aws`端點搭配使用，以執行任何`s3control`或`s3outposts`命令。

例如：

```
$  aws s3control list-regional-buckets --endpoint-url https://s3-outposts.region.api.aws
```

在來自開發套件的 Outposts 雙堆疊端點上使用 S3 AWS

本節提供如何使用 AWS 開發套件存取雙堆疊端點的範例。

AWS SDK for Java 2.x 雙堆疊端點範例

下列範例說明在 Outposts 用戶端上使用建立 S3 時，如何使用`S3ControlClient`和`S3OutpostsClient`類別啟用雙堆疊端點。AWS SDK for Java 2.x如需在 Outposts 上為 Amazon S3 建立和測試工作 Java 範例的指示，請參閱使用 AWS CLI 和 SDK for Java 開始使用。

Example — 建立啟用雙堆疊端點的**S3ControlClient**類別

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListRegionalBucketsRequest;
import software.amazon.awssdk.services.s3control.model.ListRegionalBucketsResponse;
import software.amazon.awssdk.services.s3control.model.S3ControlException;


public class DualStackEndpointsExample1 {

    public static void main(String[] args) {
        Region clientRegion = Region.of("us-east-1");
        String accountId = "111122223333";
        String navyId = "9876543210";

        try {
            // Create an S3ControlClient with dual-stack endpoints enabled.
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                                                             .region(clientRegion)
                                                             .dualstackEnabled(true)
                                                             .build();
            ListRegionalBucketsRequest listRegionalBucketsRequest =
 ListRegionalBucketsRequest.builder()

       .accountId(accountId)

       .outpostId(navyId)

       .build();

            ListRegionalBucketsResponse listBuckets =
 s3ControlClient.listRegionalBuckets(listRegionalBucketsRequest);
            System.out.printf("ListRegionalBuckets Response: %s%n",
 listBuckets.toString());
       } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 on Outposts
 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
       }
       catch (S3ControlException e) {
```

```
            // Unknown exceptions will be thrown as an instance of this type.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 on Outposts couldn't be contacted for a response, or the
 client
            // couldn't parse the response from Amazon S3 on Outposts.
            e.printStackTrace();
        }
    }
}
```

Example — 創建啟**S3OutpostsClient**用雙堆棧端點

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3outposts.S3OutpostsClient;
import software.amazon.awssdk.services.s3outposts.model.ListEndpointsRequest;
import software.amazon.awssdk.services.s3outposts.model.ListEndpointsResponse;
import software.amazon.awssdk.services.s3outposts.model.S3OutpostsException;


public class DualStackEndpointsExample2 {

    public static void main(String[] args) {
        Region clientRegion = Region.of("us-east-1");

        try {
            // Create an S3OutpostsClient with dual-stack endpoints enabled.
            S3OutpostsClient s3OutpostsClient = S3OutpostsClient.builder()
                                                                .region(clientRegion)
                                                                .dualstackEnabled(true)
                                                                .build();
            ListEndpointsRequest listEndpointsRequest =
 ListEndpointsRequest.builder().build();

            ListEndpointsResponse listEndpoints =
 s3OutpostsClient.listEndpoints(listEndpointsRequest);
            System.out.printf("ListEndpoints Response: %s%n",
 listEndpoints.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 on Outposts
 couldn't process
```

```
            // it, so it returned an error response.
            e.printStackTrace();
        }
        catch (S3OutpostsException e) {
            // Unknown exceptions will be thrown as an instance of this type.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 on Outposts couldn't be contacted for a response, or the
 client
            // couldn't parse the response from Amazon S3 on Outposts.
            e.printStackTrace();
        }
    }
}
```

如果您AWS SDK for Java 2.x在視窗上使用，您可能必須設定下列 Java 虛擬機器 (JVM) 屬性：

```
java.net.preferIPv6Addresses=true
```

使用 AWS 開發套件的 Amazon S3 程式碼範例

下列程式碼範例說明如何搭配 AWS 軟體開發套件 (SDK) 使用 Amazon S3。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境和跨服務範例中查看內容中的動作。

Scenarios (案例) 是向您展示如何呼叫相同服務中的多個函數來完成特定任務的程式碼範例。

Cross-service examples (跨服務範例) 是跨多個 AWS 服務執行的應用程式範例。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含入門相關資訊和舊版 SDK 的詳細資訊。

開始使用

您好 Amazon S3

下列程式碼範例示範如何開始使用 Amazon S3。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

C MakeLists .txt 的 CMake 文件的代碼。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")
```

```
# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_s3.cpp 來源檔案的程式碼。

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
```

```
using namespace Aws::Auth;

/*
 *  A "Hello S3" starter application which initializes an Amazon Simple Storage
 Service (Amazon S3) client
 *  and lists the Amazon S3 buckets in the selected region.
 *
 *  main function
 *
 *  Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        // You don't normally have to test that you are authenticated. But the
 S3 service permits anonymous requests, thus the s3Client will return "success"
 and 0 buckets even if you are unauthenticated, which can be confusing to a new
 user.
       auto provider =
 Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-tag");
       auto creds = provider->GetAWSCredentials();
       if (creds.IsEmpty()) {
           std::cerr << "Failed authentication" << std::endl;
       }

       Aws::S3::S3Client s3Client(clientConfig);
       auto outcome = s3Client.ListBuckets();

       if (!outcome.IsSuccess()) {
           std::cerr << "Failed with error: " << outcome.GetError() <<
 std::endl;
           result = 1;
       } else {
           std::cout << "Found " << outcome.GetResult().GetBuckets().size()
```

```
                        << " buckets\n";
            for (auto &bucket: outcome.GetResult().GetBuckets()) {
                std::cout << bucket.GetName() << std::endl;
            }
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考ListBuckets中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
```

```
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 s3Client := s3.NewFromConfig(sdkConfig)
 count := 10
 fmt.Printf("Let's list up to %v buckets for your account.\n", count)
 result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
 if err != nil {
  fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Buckets) == 0 {
  fmt.Println("You don't have any buckets!")
 } else {
  if count > len(result.Buckets) {
   count = len(result.Buckets)
  }
  for _, bucket := range result.Buckets[:count] {
   fmt.Printf("\t%v\n", *bucket.Name)
  }
 }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考ListBuckets中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
```

```
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBuckets(s3);
    }

    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考ListBuckets中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";

// When no region or credentials are provided, the SDK will use the
// region and credentials from the local AWS config.
const client = new S3Client({});

export const helloS3 = async () => {
  const command = new ListBucketsCommand({});

  const { Buckets } = await client.send(command);
  console.log("Buckets: ");
  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
  return Buckets;
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ListBuckets中的。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
use Aws\S3\S3Client;
```

```
$client = new S3Client(['region' => 'us-west-2']);
$results = $client->listBuckets();
var_dump($results);
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考ListBuckets中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import boto3


def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) resource and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
    s3_resource = boto3.resource("s3")
    print("Hello, Amazon S3! Let's list your buckets:")
    for bucket in s3_resource.buckets.all():
        print(f"\t{bucket.name}")


if __name__ == "__main__":
    hello_s3()
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListBuckets中的 Python (博托 3) API 參考。

程式碼範例

- 使用 AWS 開發套件為 Amazon S3 執行動作
  - 搭AbortMultipartUpload配 AWS 開發套件或 CLI 使用
  - 搭AbortMultipartUploads配 AWS 開發套件或 CLI 使用
  - 搭CompleteMultipartUpload配 AWS 開發套件或 CLI 使用
  - 搭CopyObject配 AWS 開發套件或 CLI 使用
  - 搭CreateBucket配 AWS 開發套件或 CLI 使用
  - 搭CreateMultiRegionAccessPoint配 AWS 開發套件或 CLI 使用
  - 搭CreateMultipartUpload配 AWS 開發套件或 CLI 使用
  - 搭DeleteBucket配 AWS 開發套件或 CLI 使用
  - 搭DeleteBucketAnalyticsConfiguration配 AWS 開發套件或 CLI 使用
  - 搭DeleteBucketCors配 AWS 開發套件或 CLI 使用
  - 搭DeleteBucketEncryption配 AWS 開發套件或 CLI 使用
  - 搭DeleteBucketInventoryConfiguration配 AWS 開發套件或 CLI 使用
  - 搭DeleteBucketLifecycle配 AWS 開發套件或 CLI 使用
  - 搭DeleteBucketMetricsConfiguration配 AWS 開發套件或 CLI 使用
  - 搭DeleteBucketPolicy配 AWS 開發套件或 CLI 使用
  - 搭DeleteBucketReplication配 AWS 開發套件或 CLI 使用
  - 搭DeleteBucketTagging配 AWS 開發套件或 CLI 使用
  - 搭DeleteBucketWebsite配 AWS 開發套件或 CLI 使用
  - 搭DeleteObject配 AWS 開發套件或 CLI 使用
  - 搭DeleteObjectTagging配 AWS 開發套件或 CLI 使用
  - 搭DeleteObjects配 AWS 開發套件或 CLI 使用
  - 搭DeletePublicAccessBlock配 AWS 開發套件或 CLI 使用
  - 搭GetBucketAccelerateConfiguration配 AWS 開發套件或 CLI 使用
  - 搭GetBucketAcl配 AWS 開發套件或 CLI 使用
  - 搭GetBucketAnalyticsConfiguration配 AWS 開發套件或 CLI 使用
  - 搭GetBucketCors配 AWS 開發套件或 CLI 使用
  - 搭GetBucketEncryption配 AWS 開發套件或 CLI 使用
  - 搭GetBucketInventoryConfiguration配 AWS 開發套件或 CLI 使用
  - 搭GetBucketLifecycleConfiguration配 AWS 開發套件或 CLI 使用

- 搭GetBucketLocation配 AWS 開發套件或 CLI 使用
- 搭GetBucketLogging配 AWS 開發套件或 CLI 使用
- 搭GetBucketMetricsConfiguration配 AWS 開發套件或 CLI 使用
- 搭GetBucketNotification配 AWS 開發套件或 CLI 使用
- 搭GetBucketPolicy配 AWS 開發套件或 CLI 使用
- 搭GetBucketPolicyStatus配 AWS 開發套件或 CLI 使用
- 搭GetBucketReplication配 AWS 開發套件或 CLI 使用
- 搭GetBucketRequestPayment配 AWS 開發套件或 CLI 使用
- 搭GetBucketTagging配 AWS 開發套件或 CLI 使用
- 搭GetBucketVersioning配 AWS 開發套件或 CLI 使用
- 搭GetBucketWebsite配 AWS 開發套件或 CLI 使用
- 搭GetObject配 AWS 開發套件或 CLI 使用
- 搭GetObjectAcl配 AWS 開發套件或 CLI 使用
- 搭GetObjectLegalHold配 AWS 開發套件或 CLI 使用
- 搭GetObjectLockConfiguration配 AWS 開發套件或 CLI 使用
- 搭GetObjectRetention配 AWS 開發套件或 CLI 使用
- 搭GetObjectTagging配 AWS 開發套件或 CLI 使用
- 搭GetPublicAccessBlock配 AWS 開發套件或 CLI 使用
- 搭HeadBucket配 AWS 開發套件或 CLI 使用
- 搭HeadObject配 AWS 開發套件或 CLI 使用
- 搭ListBucketAnalyticsConfigurations配 AWS 開發套件或 CLI 使用
- 搭ListBucketInventoryConfigurations配 AWS 開發套件或 CLI 使用
- 搭ListBuckets配 AWS 開發套件或 CLI 使用
- 搭ListMultipartUploads配 AWS 開發套件或 CLI 使用
- 搭ListObjectVersions配 AWS 開發套件或 CLI 使用
- 搭ListObjects配 AWS 開發套件或 CLI 使用
- 搭ListObjectsV2配 AWS 開發套件或 CLI 使用
- 搭PutBucketAccelerateConfiguration配 AWS 開發套件或 CLI 使用
- 搭PutBucketAcl配 AWS 開發套件或 CLI 使用
- 搭PutBucketCors配 AWS 開發套件或 CLI 使用

- 搭PutBucketEncryption配 AWS 開發套件或 CLI 使用
- 搭PutBucketLifecycleConfiguration配 AWS 開發套件或 CLI 使用
- 搭PutBucketLogging配 AWS 開發套件或 CLI 使用
- 搭PutBucketNotification配 AWS 開發套件或 CLI 使用
- 搭PutBucketNotificationConfiguration配 AWS 開發套件或 CLI 使用
- 搭PutBucketPolicy配 AWS 開發套件或 CLI 使用
- 搭PutBucketReplication配 AWS 開發套件或 CLI 使用
- 搭PutBucketRequestPayment配 AWS 開發套件或 CLI 使用
- 搭PutBucketTagging配 AWS 開發套件或 CLI 使用
- 搭PutBucketVersioning配 AWS 開發套件或 CLI 使用
- 搭PutBucketWebsite配 AWS 開發套件或 CLI 使用
- 搭PutObject配 AWS 開發套件或 CLI 使用
- 搭PutObjectAcl配 AWS 開發套件或 CLI 使用
- 搭PutObjectLegalHold配 AWS 開發套件或 CLI 使用
- 搭PutObjectLockConfiguration配 AWS 開發套件或 CLI 使用
- 搭PutObjectRetention配 AWS 開發套件或 CLI 使用
- 搭RestoreObject配 AWS 開發套件或 CLI 使用
- 搭SelectObjectContent配 AWS 開發套件或 CLI 使用
- 搭UploadPart配 AWS 開發套件或 CLI 使用

- 使用 AWS 開發套件的 Amazon S3 案例
  - 使用開發套件 AWS 為 Amazon S3 建立預先簽署的 URL
  - 使用 AWS 開發套件列出 Amazon S3 物件的網頁
  - 使用開發套件刪除不完整的分段上傳到 Amazon S3 AWS
  - 將 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的所有物件下載至本機目錄
  - 使用開發套件從多區域存取點取得 Amazon S3 物件 AWS
  - 使用 AWS 開發套件從 Amazon S3 儲存貯體取得物件，並指定如果修改後的標頭
  - 使用開 AWS 發套件開始使用 Amazon S3 儲存貯體和物件
  - 使用開 AWS 發套件開始使用 Amazon S3 物件的加密功能
  - 使用開 AWS 發套件開始使用 Amazon S3 物件的標籤
  - 使用 AWS 開發套件取得 Amazon S3 物件的合法保留組態

- 使用 AWS 開發套件使用 Amazon S3 物件鎖定功能
- 使用開發套件 AWS 管理 Amazon S3 儲存貯體的存取控制清單 (ACL)
- 使用開發套件使用 Lambda 函數批次管理版本控制的 Amazon S3 物件 AWS
- 使用開發套件 AWS 剖析 Amazon S3 URI
- 使用開發套件執行 Amazon S3 物件的多部分複本 AWS
- 使用開發套件執行 Amazon S3 物件的多部分上傳 AWS
- 使用 AWS 開發套件追蹤 Amazon S3 物件上傳或下載
- 使用 AWS SDK 進行單元和集成測試的示例方法
- 以遞迴的方式將本機目錄上傳至 Amazon Simple Storage Service (Amazon S3) 儲存貯體
- 使用 AWS 開發套件在 Amazon S3 上傳或下載大型檔案
- 使用 AWS 開發套件將大小不明的串流上傳至 Amazon S3 物件
- 使用總和檢查碼搭配使用開發套件的 Amazon S3 物件 AWS
- 使用開發套件使用 Amazon S3 版本控制物件 AWS

- 使 AWS 用開發套件的 Amazon S3 無伺服器範例
  - 使用 Amazon S3 觸發條件調用 Lambda 函數
- 使 AWS 用開發套件的 Amazon S3 跨服務範例
  - 建置 Amazon Transcribe 應用程式
  - 使用 AWS SDK 將文本轉換為語音並返回文本
  - 建立相片資產管理應用程式，讓使用者以標籤管理相片
  - 建立 Amazon Textract Explorer 應用程式
  - 使用開發套件使用 Amazon 重新認知功能偵測影像中的個人防護裝置 AWS
  - 使用 AWS SDK 檢測從圖像中提取的文本中的實體
  - 使用 AWS SDK 偵測影像中的臉孔
  - 使用開發套件使用 Amazon Rekognition 偵測影像中的物件 AWS
  - 使用開發套件使用 Amazon Rekognition 偵測影片中的人物和物件 AWS
  - 使用 SDK 儲存 EXIF 和其他影像資訊 AWS
  - 使用 S3 物件 Lambda 為您的應用程式轉換資料

使用 AWS 開發套件為 Amazon S3 執行動作

下列程式碼範例示範如何使用 AWS 開發套件執行個別 Amazon S3 動作。這些摘錄會呼叫 Amazon S3 API，是必須在內容中執行之大型程式的程式碼摘錄。每個範例都包含一個連結 GitHub，您可以在其中找到設定和執行程式碼的指示。

下列範例僅包含最常使用的動作。如需完整的列表，請參閱 Amazon Simple Storage Service (Amazon S3) API 參考資料。

範例

- 搭AbortMultipartUpload配 AWS 開發套件或 CLI 使用
- 搭AbortMultipartUploads配 AWS 開發套件或 CLI 使用
- 搭CompleteMultipartUpload配 AWS 開發套件或 CLI 使用
- 搭CopyObject配 AWS 開發套件或 CLI 使用
- 搭CreateBucket配 AWS 開發套件或 CLI 使用
- 搭CreateMultiRegionAccessPoint配 AWS 開發套件或 CLI 使用
- 搭CreateMultipartUpload配 AWS 開發套件或 CLI 使用
- 搭DeleteBucket配 AWS 開發套件或 CLI 使用
- 搭DeleteBucketAnalyticsConfiguration配 AWS 開發套件或 CLI 使用
- 搭DeleteBucketCors配 AWS 開發套件或 CLI 使用
- 搭DeleteBucketEncryption配 AWS 開發套件或 CLI 使用
- 搭DeleteBucketInventoryConfiguration配 AWS 開發套件或 CLI 使用
- 搭DeleteBucketLifecycle配 AWS 開發套件或 CLI 使用
- 搭DeleteBucketMetricsConfiguration配 AWS 開發套件或 CLI 使用
- 搭DeleteBucketPolicy配 AWS 開發套件或 CLI 使用
- 搭DeleteBucketReplication配 AWS 開發套件或 CLI 使用
- 搭DeleteBucketTagging配 AWS 開發套件或 CLI 使用
- 搭DeleteBucketWebsite配 AWS 開發套件或 CLI 使用
- 搭DeleteObject配 AWS 開發套件或 CLI 使用
- 搭DeleteObjectTagging配 AWS 開發套件或 CLI 使用
- 搭DeleteObjects配 AWS 開發套件或 CLI 使用
- 搭DeletePublicAccessBlock配 AWS 開發套件或 CLI 使用
- 搭GetBucketAccelerateConfiguration配 AWS 開發套件或 CLI 使用

- 搭GetBucketAcl配 AWS 開發套件或 CLI 使用
- 搭GetBucketAnalyticsConfiguration配 AWS 開發套件或 CLI 使用
- 搭GetBucketCors配 AWS 開發套件或 CLI 使用
- 搭GetBucketEncryption配 AWS 開發套件或 CLI 使用
- 搭GetBucketInventoryConfiguration配 AWS 開發套件或 CLI 使用
- 搭GetBucketLifecycleConfiguration配 AWS 開發套件或 CLI 使用
- 搭GetBucketLocation配 AWS 開發套件或 CLI 使用
- 搭GetBucketLogging配 AWS 開發套件或 CLI 使用
- 搭GetBucketMetricsConfiguration配 AWS 開發套件或 CLI 使用
- 搭GetBucketNotification配 AWS 開發套件或 CLI 使用
- 搭GetBucketPolicy配 AWS 開發套件或 CLI 使用
- 搭GetBucketPolicyStatus配 AWS 開發套件或 CLI 使用
- 搭GetBucketReplication配 AWS 開發套件或 CLI 使用
- 搭GetBucketRequestPayment配 AWS 開發套件或 CLI 使用
- 搭GetBucketTagging配 AWS 開發套件或 CLI 使用
- 搭GetBucketVersioning配 AWS 開發套件或 CLI 使用
- 搭GetBucketWebsite配 AWS 開發套件或 CLI 使用
- 搭GetObject配 AWS 開發套件或 CLI 使用
- 搭GetObjectAcl配 AWS 開發套件或 CLI 使用
- 搭GetObjectLegalHold配 AWS 開發套件或 CLI 使用
- 搭GetObjectLockConfiguration配 AWS 開發套件或 CLI 使用
- 搭GetObjectRetention配 AWS 開發套件或 CLI 使用
- 搭GetObjectTagging配 AWS 開發套件或 CLI 使用
- 搭GetPublicAccessBlock配 AWS 開發套件或 CLI 使用
- 搭HeadBucket配 AWS 開發套件或 CLI 使用
- 搭HeadObject配 AWS 開發套件或 CLI 使用
- 搭ListBucketAnalyticsConfigurations配 AWS 開發套件或 CLI 使用
- 搭ListBucketInventoryConfigurations配 AWS 開發套件或 CLI 使用
- 搭ListBuckets配 AWS 開發套件或 CLI 使用
- 搭ListMultipartUploads配 AWS 開發套件或 CLI 使用

- 搭ListObjectVersions配 AWS 開發套件或 CLI 使用
- 搭ListObjects配 AWS 開發套件或 CLI 使用
- 搭ListObjectsV2配 AWS 開發套件或 CLI 使用
- 搭PutBucketAccelerateConfiguration配 AWS 開發套件或 CLI 使用
- 搭PutBucketAcl配 AWS 開發套件或 CLI 使用
- 搭PutBucketCors配 AWS 開發套件或 CLI 使用
- 搭PutBucketEncryption配 AWS 開發套件或 CLI 使用
- 搭PutBucketLifecycleConfiguration配 AWS 開發套件或 CLI 使用
- 搭PutBucketLogging配 AWS 開發套件或 CLI 使用
- 搭PutBucketNotification配 AWS 開發套件或 CLI 使用
- 搭PutBucketNotificationConfiguration配 AWS 開發套件或 CLI 使用
- 搭PutBucketPolicy配 AWS 開發套件或 CLI 使用
- 搭PutBucketReplication配 AWS 開發套件或 CLI 使用
- 搭PutBucketRequestPayment配 AWS 開發套件或 CLI 使用
- 搭PutBucketTagging配 AWS 開發套件或 CLI 使用
- 搭PutBucketVersioning配 AWS 開發套件或 CLI 使用
- 搭PutBucketWebsite配 AWS 開發套件或 CLI 使用
- 搭PutObject配 AWS 開發套件或 CLI 使用
- 搭PutObjectAcl配 AWS 開發套件或 CLI 使用
- 搭PutObjectLegalHold配 AWS 開發套件或 CLI 使用
- 搭PutObjectLockConfiguration配 AWS 開發套件或 CLI 使用
- 搭PutObjectRetention配 AWS 開發套件或 CLI 使用
- 搭RestoreObject配 AWS 開發套件或 CLI 使用
- 搭SelectObjectContent配 AWS 開發套件或 CLI 使用
- 搭UploadPart配 AWS 開發套件或 CLI 使用

搭**AbortMultipartUpload**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用AbortMultipartUpload。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 刪除不完整的分段上傳

CLI

AWS CLI

中止指定的分段上傳

下列abort-multipart-upload指令會中止儲存貯體multipart/01中金鑰的多部分上傳。my-bucket

```
aws s3api abort-multipart-upload \
    --bucket my-bucket \
    --key multipart/01 \
    --upload-id
 dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3U
```

此命令所需的上傳 ID 由輸出create-multipart-upload，也可以使用擷取list-multipart-uploads。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考AbortMultipartUpload中的。

PowerShell

適用的工具 PowerShell

範例 1：此指令會中止 5 天前建立的分段上傳。

```
Remove-S3MultipartUpload -BucketName test-files -DaysBefore 5
```

範例 2：此指令會中止在 2014 年 1 月 2 日之前建立的分段上傳。

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "Thursday, January
 02, 2014"
```

範例 3：此指令會中止在 2014 年 1 月 2 日 10:45：37 之前建立的分段上傳。

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "2014/01/02
 10:45:37"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程AbortMultipartUpload式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭AbortMultipartUploads配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用AbortMultipartUploads。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;

/// <summary>
/// This example shows how to use the Amazon Simple Storage Service
/// (Amazon S3) to stop a multi-part upload process using the Amazon S3
/// TransferUtility.
/// </summary>
public class AbortMPU
{
    public static async Task Main()
    {
        string bucketName = "doc-example-bucket";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the S3 client object's constructor.
        // For example: RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();
```

```
            await AbortMPUAsync(client, bucketName);
        }

        /// <summary>
        /// Cancels the multi-part copy process.
        /// </summary>
        /// <param name="client">The initialized client object used to create
        /// the TransferUtility object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the
        /// multi-part copy operation is in progress.</param>
        public static async Task AbortMPUAsync(IAmazonS3 client, string
bucketName)
        {
            try
            {
                var transferUtility = new TransferUtility(client);

                // Cancel all in-progress uploads initiated before the specified
date.
                await transferUtility.AbortMultipartUploadsAsync(
                    bucketName, DateTime.Now.AddDays(-7));
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考AbortMultipartUploads中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**CompleteMultipartUpload**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CompleteMultipartUpload。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 執行分段複製
- 執行分段上傳
- 使用檢查總和

CLI

AWS CLI

下列命令會完成值my-bucket區multipart/01中金鑰的多部分上傳：

```
aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket my-bucket --key 'multipart/01' --upload-id
 dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3U
```

此命令所需的上傳 ID 由輸出create-multipart-upload，也可以使用擷取list-multipart-uploads。

上述命令中的 multipart upload 選項採用 JSON 結構，該結構描述了應重新組合到完整檔案中的多部分上傳部分。在此範例中，file://前置詞用於從名為的本機資料夾中的檔案載入 JSON 結構mpustruct。

輸出結構：

```
{
  "Parts": [
    {
      "ETag": "e868e0f4719e394144ef36531ee6824c",
      "PartNumber": 1
    },
    {
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",
      "PartNumber": 2
    },
    {
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",
      "PartNumber": 3
    }
```

```
  ]
}
```

每次使用upload-part指令上載零件時，都會輸出每個零件的 ETag 值，也可以透過呼叫擷取list-parts或透過取得每個零件的 MD5 總和來計算。

輸出：

```
{
    "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",
    "Bucket": "my-bucket",
    "Location": "https://my-bucket.s3.amazonaws.com/multipart%2F01",
    "Key": "multipart/01"
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CompleteMultipartUpload中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await
        .unwrap();
```

- 如需 API 的詳細資訊，請參閱 AWS SDK CompleteMultipartUpload中的 Rust API 參考資料。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭CopyObject配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CopyObject。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用儲存貯體和物件
- 開始使用加密

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

public class CopyObject
{
    public static async Task Main()
    {
        // Specify the AWS Region where your buckets are located if it is
        // different from the AWS Region of the default user.
        IAmazonS3 s3Client = new AmazonS3Client();

        // Remember to change these values to refer to your Amazon S3
objects.
        string sourceBucketName = "doc-example-bucket1";
        string destinationBucketName = "doc-example-bucket2";
        string sourceObjectKey = "testfile.txt";
```

```
            string destinationObjectKey = "testfilecopy.txt";

            Console.WriteLine($"Copying {sourceObjectKey} from {sourceBucketName}
 to ");
            Console.WriteLine($"{destinationBucketName} as
 {destinationObjectKey}");

            var response = await CopyingObjectAsync(
                s3Client,
                sourceObjectKey,
                destinationObjectKey,
                sourceBucketName,
                destinationBucketName);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("\nCopy complete.");
            }
        }

        /// <summary>
        /// This method calls the AWS SDK for .NET to copy an
        /// object from one Amazon S3 bucket to another.
        /// </summary>
        /// <param name="client">The Amazon S3 client object.</param>
        /// <param name="sourceKey">The name of the object to be copied.</param>
        /// <param name="destinationKey">The name under which to save the copy.</
param>
        /// <param name="sourceBucketName">The name of the Amazon S3 bucket
        /// where the file is located now.</param>
        /// <param name="destinationBucketName">The name of the Amazon S3
        /// bucket where the copy should be saved.</param>
        /// <returns>Returns a CopyObjectResponse object with the results from
        /// the async call.</returns>
        public static async Task<CopyObjectResponse> CopyingObjectAsync(
            IAmazonS3 client,
            string sourceKey,
            string destinationKey,
            string sourceBucketName,
            string destinationBucketName)
        {
            var response = new CopyObjectResponse();
            try
            {
```

```
                var request = new CopyObjectRequest
                {
                    SourceBucket = sourceBucketName,
                    SourceKey = sourceKey,
                    DestinationBucket = destinationBucketName,
                    DestinationKey = destinationKey,
                };
                response = await client.CopyObjectAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error copying object: '{ex.Message}'");
            }

            return response;
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考CopyObject中的。

Bash

AWS CLI 與 bash 腳本

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
```

```
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CopyObject中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
 &fromBucket, const Aws::String &toBucket,
                             const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
            .WithKey(objectKey)
            .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;

    } else {
        std::cout << "Successfully copied " << objectKey << " from " <<
 fromBucket <<
                  " to " << toBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考CopyObject中的。

CLI

AWS CLI

下列指令會將物件從複製bucket-1到bucket-2：

```
aws s3api copy-object --copy-source bucket-1/test.txt --key test.txt --bucket
 bucket-2
```

輸出：

```
{
    "CopyObjectResult": {
```

```
        "LastModified": "2015-11-10T01:07:25.000Z",
        "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""
    },
    "VersionId": "YdnYvTCVDqRRFA.NFJjy36p0hxifMlkA"
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CopyObject中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(sourceBucket string, destinationBucket
 string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(context.TODO(), &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't copy object from %v:%v to %v:%v. Here's why: %v\n",
   sourceBucket, objectKey, destinationBucket, objectKey, err)
```

```
 }
 return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考CopyObject中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用 S3Client 複製物件。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <objectKey> <fromBucket> <toBucket>
```

```
                Where:
                    objectKey - The name of the object (for example, book.pdf).
                    fromBucket - The S3 bucket name that contains the object (for
example, bucket1).
                    toBucket - The S3 bucket to copy the object to (for example,
bucket2).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String objectKey = args[0];
        String fromBucket = args[1];
        String toBucket = args[2];
        System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        copyBucketObject(s3, fromBucket, objectKey, toBucket);
        s3.close();
    }

    public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
                .sourceBucket(fromBucket)
                .sourceKey(objectKey)
                .destinationBucket(toBucket)
                .destinationKey(objectKey)
                .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
        return "";
    }
}
```

使用 S3 TransferManager 將物件從一個儲存貯體複製到另一個儲存貯體。檢視完整檔案並測試。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

    public String copyObject(S3TransferManager transferManager, String
 bucketName,
            String key, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
                .sourceBucket(bucketName)
                .sourceKey(key)
                .destinationBucket(destinationBucket)
                .destinationKey(destinationKey)
                .build();

        CopyRequest copyRequest = CopyRequest.builder()
                .copyObjectRequest(copyObjectRequest)
                .build();

        Copy copy = transferManager.copy(copyRequest);

        CompletedCopy completedCopy = copy.completionFuture().join();
        return completedCopy.response().copyObjectResult().eTag();
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考CopyObject中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

複製物件

```
import { S3Client, CopyObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new CopyObjectCommand({
    CopySource: "SOURCE_BUCKET/SOURCE_OBJECT_KEY",
    Bucket: "DESTINATION_BUCKET",
    Key: "NEW_OBJECT_KEY",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考CopyObject中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK CopyObject中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

簡單複製物件。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with object copying before continuing.");
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考CopyObject中的。

PowerShell

適用的工具 PowerShell

示例 1：此命令將對象「sample.txt」從存儲桶「測試文件」複製到同一存儲桶，但使用「sample-copy.txt」的新密鑰。

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-
copy.txt
```

示例 2：此命令將對象「sample.txt」從存儲桶「測試文件」複製到帶有密鑰「sample-copy.txt」的存儲桶「備份文件」。

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-
copy.txt -DestinationBucket backup-files
```

範例 3：此命令會從值區「測試檔案」將物件「sample.txt」下載至名稱為「local-sample.txt」的本機檔案。

```
Copy-S3Object -BucketName test-files -Key sample.txt -LocalFile local-sample.txt
```

範例 4：將單一物件下載至指定的檔案。下載的文件將在 c:\downloads\data\archive.zip 找到

```
Copy-S3Object -BucketName test-files -Key data/archive.zip -LocalFolder c:
\downloads
```

範例 5：將符合指定 key prefix 的所有物件下載至本機資料夾。相對索引鍵階層會保留為整體下載位置中的子資料夾。

```
Copy-S3Object -BucketName test-files -KeyPrefix data -LocalFolder c:\downloads
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程CopyObject式參考中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
```

```
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
key.
                            This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件CopyObject中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

複製物件

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
 why: #{e.message}"
  end
end

# Example usage:
```

```
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

複製物件，然後將伺服器端加密新增至目標物件。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the
 target key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
```

```
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
 why: #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key,
 target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考CopyObject中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
pub async fn copy_object(
    client: &Client,
    bucket_name: &str,
    object_key: &str,
    target_key: &str,
) -> Result<CopyObjectOutput, SdkError<CopyObjectError>> {
    let mut source_bucket_and_object: String = "".to_owned();
    source_bucket_and_object.push_str(bucket_name);
    source_bucket_and_object.push('/');
    source_bucket_and_object.push_str(object_key);

    client
        .copy_object()
        .copy_source(source_bucket_and_object)
        .bucket(bucket_name)
        .key(target_key)
        .send()
        .await
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK CopyObject中的 Rust API 參考資料。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
TRY.
    lo_s3->copyobject(
      iv_bucket = iv_dest_bucket
      iv_key = iv_dest_object
      iv_copysource = |{ iv_src_bucket }/{ iv_src_object }|
    ).
    MESSAGE 'Object copied to another bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
  CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱 AWS SDK CopyObject中的 SAP ABAP API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        _ = try await client.copyObject(input: input)
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK CopyObject中的斯威夫特 API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭CreateBucket配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CreateBucket。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用儲存貯體和物件
- 使用版本化物件

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        /// <summary>
        /// Shows how to create a new Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket to create.</param>
        /// <returns>A boolean value representing the success or failure of
        /// the bucket creation process.</returns>
        public static async Task<bool> CreateBucketAsync(IAmazonS3 client, string
bucketName)
        {
            try
            {
                var request = new PutBucketRequest
                {
                    BucketName = bucketName,
                    UseClientRegion = true,
                };

                var response = await client.PutBucketAsync(request);
                return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error creating bucket: '{ex.Message}'");
                return false;
            }
        }
```

建立啟用物件鎖定的值區。

```
    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
enableObjectLock)
    {
```

```
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考CreateBucket中的。

Bash

AWS CLI 與 bash 腳本

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
#############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#############################################################################
function iecho() {
```

```
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
########################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

########################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
########################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally
 unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
```

```
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi

  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
```

```
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreateBucket中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(

 Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                         clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
```

```
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
                  " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考CreateBucket中的。

CLI

AWS CLI

範例 1：若要建立值區

下列create-bucket範例會建立名為的值區my-bucket：

```
aws s3api create-bucket \
    --bucket my-bucket \
    --region us-east-1
```

輸出：

```
{
    "Location": "/my-bucket"
}
```

如需詳細資訊，請參閱 Amazon S3 使用者指南中的建立儲存貯體。

範例 2：若要建立強制擁有者的值區

下列create-bucket範例會建立名my-bucket為使用 S3 物件擁有權強制執行儲存貯體擁有者設定的儲存貯體。

```
aws s3api create-bucket \
```

```
    --bucket my-bucket \
    --region us-east-1 \
    --object-ownership BucketOwnerEnforced
```

輸出：

```
{
    "Location": "/my-bucket"
}
```

如需詳細資訊，請參閱《Amazon S3 使用者指南》中的控制物件的所有權並停用 ACL。

範例 3：在「us-east-1」區域之外建立值區

下列create-bucket範例會建立區域my-bucket中名為的值eu-west-1區。位於以外的區域us-east-1需要指定適當LocationConstraint的區域，才能在所需的區域中建立值區。

```
aws s3api create-bucket \
    --bucket my-bucket \
    --region eu-west-1 \
    --create-bucket-configuration LocationConstraint=eu-west-1
```

輸出：

```
{
    "Location": "http://my-bucket.s3.amazonaws.com/"
}
```

如需詳細資訊，請參閱 Amazon S3 使用者指南中的建立儲存貯體。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreateBucket中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用預設組態建立值區。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(name string, region string) error {
 _, err := basics.S3Client.CreateBucket(context.TODO(), &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  log.Printf("Couldn't create bucket %v in Region %v. Here's why: %v\n",
   name, region, err)
 }
 return err
}
```

建立具有物件鎖定的值區，並等待它存在。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}
```

```
// CreateBucketWithLock creates a new S3 bucket with optional object locking
 enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考CreateBucket中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

建立儲存貯體。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class CreateBucket {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the bucket to create. The bucket
 name must be unique, or an error occurs.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Creating a bucket named %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        createBucket(s3, bucketName);
        s3.close();
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
 s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

建立啟用物件鎖定的值區。

```
    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考CreateBucket中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

建立儲存貯體。

```
import { CreateBucketCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new CreateBucketCommand({
```

```
    // The name of the bucket. Bucket names are unique and have several other
 constraints.
    // See https://docs.aws.amazon.com/AmazonS3/latest/userguide/
bucketnamingrules.html
    Bucket: "bucket-name",
  });

  try {
    const { Location } = await client.send(command);
    console.log(`Bucket created with location ${Location}`);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考CreateBucket中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun createNewBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK CreateBucket中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

建立儲存貯體。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考CreateBucket中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用預設設定建立儲存貯體。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def create(self, region_override=None):
        """
        Create an Amazon S3 bucket in the default Region for the account or in
 the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this
 is
                                not specified, the Region configured in your
 shared
                                credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
            self.bucket.create(CreateBucketConfiguration={"LocationConstraint":
 region})

            self.bucket.wait_until_exists()
            logger.info("Created bucket '%s' in region=%s", self.bucket.name,
 region)
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name,
                region,
            )
```

```
            raise error
```

透過生命週期組態建立版本控制儲存貯體。

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
```

```
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件CreateBucket中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                  create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      "None. You must create a bucket before you can get its location!"
    else
```

```
      @bucket.client.get_bucket_location(bucket:
 @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("doc-example-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考CreateBucket中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
pub async fn create_bucket(
    client: &Client,
    bucket_name: &str,
    region: &str,
) -> Result<CreateBucketOutput, SdkError<CreateBucketError>> {
    let constraint = BucketLocationConstraint::from(region);
    let cfg = CreateBucketConfiguration::builder()
        .location_constraint(constraint)
```

```
        .build();
    client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK CreateBucket中的 Rust API 參考資料。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    TRY.
        lo_s3->createbucket(
            iv_bucket = iv_bucket_name
        ).
        MESSAGE 'S3 bucket created.' TYPE 'I'.
      CATCH /aws1/cx_s3_bucketalrdyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
      CATCH /aws1/cx_s3_bktalrdyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
    ENDTRY.
```

- 如需 API 詳細資訊，請參閱 AWS SDK CreateBucket中的 SAP ABAP API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
public func createBucket(name: String) async throws {
    let config = S3ClientTypes.CreateBucketConfiguration(
        locationConstraint: .usEast2
    )
    let input = CreateBucketInput(
        bucket: name,
        createBucketConfiguration: config
    )
    _ = try await client.createBucket(input: input)
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK CreateBucket中的斯威夫特 API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`CreateMultiRegionAccessPoint`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`CreateMultiRegionAccessPoint`。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

設定 S3 控制用戶端，將請求傳送至 us-west-2 區域。

```
        suspend fun createS3ControlClient(): S3ControlClient {
            // Configure your S3ControlClient to send requests to US West
(Oregon).
            val s3Control = S3ControlClient.fromEnvironment {
                region = "us-west-2"
            }
            return s3Control
        }
```

建立多區域存取點。

```
    suspend fun createMrap(
        s3Control: S3ControlClient,
        accountIdParam: String,
        bucketName1: String,
        bucketName2: String,
        mrapName: String,
    ): String {
        println("Creating MRAP ...")
        val createMrapResponse: CreateMultiRegionAccessPointResponse =
            s3Control.createMultiRegionAccessPoint {
                accountId = accountIdParam
                clientToken = UUID.randomUUID().toString()
                details {
                    name = mrapName
                    regions = listOf(
                        Region {
                            bucket = bucketName1
                        },
```

```
                        Region {
                            bucket = bucketName2
                        },
                    )
                }
            }
        val requestToken: String? = createMrapResponse.requestTokenArn

        // Use the request token to check for the status of the
CreateMultiRegionAccessPoint operation.
        if (requestToken != null) {
            waitForSucceededStatus(s3Control, requestToken, accountIdParam)
            println("MRAP created")
        }

        val getMrapResponse =
            s3Control.getMultiRegionAccessPoint(
                input = GetMultiRegionAccessPointRequest {
                    accountId = accountIdParam
                    name = mrapName
                },
            )
        val mrapAlias = getMrapResponse.accessPoint?.alias
        return "arn:aws:s3::$accountIdParam:accesspoint/$mrapAlias"
    }
```

等待多區域存取點可用。

```
        suspend fun waitForSucceededStatus(
            s3Control: S3ControlClient,
            requestToken: String,
            accountIdParam: String,
            timeBetweenChecks: Duration = 1.minutes,
        ) {
            var describeResponse: DescribeMultiRegionAccessPointOperationResponse
            describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )
```

```
            var status: String? = describeResponse.asyncOperation?.requestStatus
            while (status != "SUCCEEDED") {
                delay(timeBetweenChecks)
                describeResponse =
s3Control.describeMultiRegionAccessPointOperation(
                    input = DescribeMultiRegionAccessPointOperationRequest {
                        accountId = accountIdParam
                        requestTokenArn = requestToken
                    },
                )
                status = describeResponse.asyncOperation?.requestStatus
                println(status)
            }
        }
```

- 如需詳細資訊，請參閱適用於 Kotlin 的AWS SDK 開發人員指南。
- 有關 API 的詳細信息，請參閱 AWS SDK CreateMultiRegionAccessPoint中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**CreateMultipartUpload**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`CreateMultipartUpload`。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 執行分段複製
- 執行分段上傳
- 使用檢查總和

CLI

AWS CLI

以下命令`my-bucket`使用密鑰`multipart/01`在存儲桶中創建一個多部分上傳：

```
aws s3api create-multipart-upload --bucket my-bucket --key 'multipart/01'
```

輸出：

```
{
    "Bucket": "my-bucket",
    "UploadId":
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
    "Key": "multipart/01"
}
```

完成的文件將在存儲桶01中名為的文件夾multipart中命名my-bucket。儲存上傳 ID、金鑰和值區名稱，以便與upload-part指令搭配使用。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreateMultipartUpload中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await
        .unwrap();
```

- 如需 API 的詳細資訊，請參閱 AWS SDK CreateMultipartUpload中的 Rust API 參考資料。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteBucket配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteBucket。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用儲存貯體和物件

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        /// <summary>
        /// Shows how to delete an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to
delete.</param>
        /// <returns>A boolean value that represents the success or failure of
        /// the delete operation.</returns>
        public static async Task<bool> DeleteBucketAsync(IAmazonS3 client, string
bucketName)
        {
            var request = new DeleteBucketRequest
            {
                BucketName = bucketName,
            };

            var response = await client.DeleteBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeleteBucket中的。

Bash

AWS CLI 與 bash 腳本

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
```

```
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteBucket中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {

    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考DeleteBucket中的。

CLI

AWS CLI

下列指令會刪除名為的值區my-bucket：

```
aws s3api delete-bucket --bucket my-bucket --region us-east-1
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteBucket中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is
 returned.
func (basics BucketBasics) DeleteBucket(bucketName string) error {
 _, err := basics.S3Client.DeleteBucket(context.TODO(), &s3.DeleteBucketInput{
```

```
  Bucket: aws.String(bucketName)})
 if err != nil {
  log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
 }
 return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考DeleteBucket中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucket)
                .build();

        s3.deleteBucket(deleteBucketRequest);
        s3.close();
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DeleteBucket中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

刪除儲存貯體。

```
import { DeleteBucketCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Delete a bucket.
export const main = async () => {
  const command = new DeleteBucketCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeleteBucket中的。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

刪除空的儲存貯體。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
```

```
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考DeleteBucket中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會從值區「測試檔案」移除所有物件和物件版本，然後刪除值區。該命令將在繼續之前提示進行確認。新增-Force 開關以抑制確認。請注意，不能刪除非空的值區。

```
Remove-S3Bucket -BucketName test-files -DeleteBucketContent
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteBucket式參考中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
```

```
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete(self):
        """
        Delete the bucket. The bucket must be empty or an error is raised.
        """
        try:
            self.bucket.delete()
            self.bucket.wait_until_not_exists()
            logger.info("Bucket %s successfully deleted.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DeleteBucket中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
 # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
 #
 # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
 def delete_bucket(bucket)
   puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
   answer = gets.chomp.downcase
   if answer == "y"
     bucket.objects.batch_delete!
     bucket.delete
     puts("Emptied and deleted bucket #{bucket.name}.\n")
```

```
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考DeleteBucket中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
pub async fn delete_bucket(client: &Client, bucket_name: &str) -> Result<(),
 Error> {
    client.delete_bucket().bucket(bucket_name).send().await?;
    println!("Bucket deleted");
    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK DeleteBucket中的 Rust API 參考資料。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    TRY.

        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name
        ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- 如需 API 詳細資訊，請參閱 AWS SDK DeleteBucket中的 SAP ABAP API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
        _ = try await client.deleteBucket(input: input)
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteBucket中的斯威夫特 API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteBucketAnalyticsConfiguration配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteBucketAnalyticsConfiguration。

CLI

AWS CLI

若要刪除值區的分析設定

下列delete-bucket-analytics-configuration範例會移除指定值區和 ID 的分析設定。

```
aws s3api delete-bucket-analytics-configuration \
    --bucket my-bucket \
    --id 1
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteBucketAnalyticsConfiguration中的。

PowerShell

適用的工具 PowerShell

範例 1：命令會移除指定 S3 儲存貯體中名稱為「testfilter」的分析篩選器。

```
Remove-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId
 'testfilter'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteBucketAnalyticsConfiguration式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteBucketCors配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteBucketCors。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        /// <summary>
        /// Deletes a CORS configuration from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to delete the CORS configuration from the bucket.</param>
        private static async Task DeleteCORSConfigurationAsync(AmazonS3Client
client)
        {
            DeleteCORSConfigurationRequest request = new
DeleteCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            await client.DeleteCORSConfigurationAsync(request);
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeleteBucketCors中的。

CLI

AWS CLI

下列指令會從名my-bucket為的值區刪除跨來源資源共用設定：

```
aws s3api delete-bucket-cors --bucket my-bucket
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteBucketCors中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
        try:
            self.bucket.Cors().delete()
            logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete CORS from bucket '%s'.",
 self.bucket.name)
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DeleteBucketCors中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考DeleteBucketCors中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteBucketEncryption配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteBucketEncryption。

CLI

AWS CLI

刪除值區的伺服器端加密組態

下列delete-bucket-encryption範例會刪除指定值區的伺服器端加密組態。

```
aws s3api delete-bucket-encryption \
    --bucket my-bucket
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteBucketEncryption中的。

PowerShell

適用的工具 PowerShell

範例 1：這會停用為提供的 S3 儲存貯體啟用的加密。

```
Remove-S3BucketEncryption -BucketName 's3casetestbucket'
```

輸出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
 target "s3casetestbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteBucketEncryption式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteBucketInventoryConfiguration配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteBucketInventoryConfiguration。

CLI

AWS CLI

若要刪除值區的庫存組態

下列delete-bucket-inventory-configuration範例會刪除具有指定值區之識別碼的1詳細目錄組態。

```
aws s3api delete-bucket-inventory-configuration \
    --bucket my-bucket \
    --id 1
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteBucketInventoryConfiguration中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會移除與指定 S3 儲存貯體對應的名為 testInventoryName " 的入侵項目。

```
Remove-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId
 'testInventoryName'
```

輸出：

```
Confirm
Are you sure you want to perform this action?
```

```
Performing the operation "Remove-S3BucketInventoryConfiguration
 (DeleteBucketInventoryConfiguration)" on target "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程[DeleteBucketInventoryConfiguration](式參考中的。)

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱[搭配 AWS SDK 使用此服務](。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。)

搭**DeleteBucketLifecycle**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteBucketLifecycle。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](。)

```
        /// <summary>
        /// This method removes the Lifecycle configuration from the named
        /// S3 bucket.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the RemoveLifecycleConfigAsync method.</param>
        /// <param name="bucketName">A string representing the name of the
        /// S3 bucket from which the configuration will be removed.</param>
        public static async Task RemoveLifecycleConfigAsync(IAmazonS3 client,
string bucketName)
        {
            var request = new DeleteLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
```

```
            await client.DeleteLifecycleConfigurationAsync(request);
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeleteBucketLifecycle中的。

CLI

AWS CLI

下列指令會從名為的值區刪除生命週期組態my-bucket：

```
aws s3api delete-bucket-lifecycle --bucket my-bucket
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteBucketLifecycle中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_lifecycle_configuration(self):
```

```
        """
        Remove the lifecycle configuration from the specified bucket.
        """
        try:
            self.bucket.LifecycleConfiguration().delete()
            logger.info(
                "Deleted lifecycle configuration for bucket '%s'.",
self.bucket.name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete lifecycle configuration for bucket '%s'.",
                self.bucket.name,
            )
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DeleteBucketLifecycle中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteBucketMetricsConfiguration配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteBucketMetricsConfiguration。

CLI

AWS CLI

若要刪除值區的指標組態

下列delete-bucket-metrics-configuration範例會移除指定值區和 ID 的指標組態。

```
aws s3api delete-bucket-metrics-configuration \
    --bucket my-bucket \
    --id 123
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteBucketMetricsConfiguration中的。

PowerShell

適用的工具 PowerShell

範例 1：命令會移除指定 S3 儲存貯體中名稱為「testmetrics」的指標篩選器。

```
Remove-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId
 'testmetrics'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteBucketMetricsConfiguration式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteBucketPolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteBucketPolicy。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,
                                    const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
 client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess()) {
```

```
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucketPolicy: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考DeleteBucketPolicy中的。

CLI

AWS CLI

下列命令會從名為的值區刪除值區政策my-bucket：

```
aws s3api delete-bucket-policy --bucket my-bucket
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteBucketPolicy中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the policy from
 (for example, bucket1).""";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    // Delete the bucket policy.
    public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
        DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3.deleteBucketPolicy(delReq);
```

```
            System.out.println("Done!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DeleteBucketPolicy中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

刪除儲存貯體政策。

```
import { DeleteBucketPolicyCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// This will remove the policy from the bucket.
export const main = async () => {
  const command = new DeleteBucketPolicyCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeleteBucketPolicy中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteBucketPolicy中的 Kotlin API 參考。

PowerShell

適用的工具 PowerShell

範例 1：命令會移除與指定 S3 儲存貯體相關聯的儲存貯體政策。

```
Remove-S3BucketPolicy -BucketName 's3testbucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteBucketPolicy式參考中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
            self.bucket.Policy().delete()
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception(
                "Couldn't delete policy for bucket '%s'.", self.bucket.name
            )
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DeleteBucketPolicy中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  def delete_policy
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's
 why: #{e.message}"
    false
  end

end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考DeleteBucketPolicy中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`DeleteBucketReplication`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`DeleteBucketReplication`。

CLI

AWS CLI

下列命令會從名稱為的值區刪除複寫組態my-bucket：

```
aws s3api delete-bucket-replication --bucket my-bucket
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteBucketReplication中的。

PowerShell

適用的工具 PowerShell

範例 1：刪除與名為「mybucket' 的值區相關聯的複製組態。請注意，此操作需要 s3：操 DeleteReplicationConfiguration 作的許可。在進行操作之前，系統將提示您進行確認-要抑制確認，請使用-Force 開關。

```
Remove-S3BucketReplication -BucketName mybucket
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteBucketReplication式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteBucketTagging配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteBucketTagging。

CLI

AWS CLI

下列指令會從名為的值區刪除標記組態my-bucket：

```
aws s3api delete-bucket-tagging --bucket my-bucket
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteBucketTagging中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會移除與指定 S3 儲存貯體相關聯的所有標籤。

```
Remove-S3BucketTagging -BucketName 's3testbucket'
```

輸出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
 "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteBucketTagging式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteBucketWebsite配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteBucketWebsite。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                     const Aws::S3::S3ClientConfiguration
 &clientConfig) {
```

```
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
            client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考DeleteBucketWebsite中的。

CLI

AWS CLI

下列指令會從名為的值區刪除網站設定my-bucket：

```
aws s3api delete-bucket-website --bucket my-bucket
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteBucketWebsite中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the website
 configuration from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket:
 %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }
```

```
    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName)
 {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DeleteBucketWebsite中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

從儲存貯體刪除網站組態

```
import { DeleteBucketWebsiteCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Disable static website hosting on the bucket.
export const main = async () => {
  const command = new DeleteBucketWebsiteCommand({
    Bucket: "test-bucket",
  });
```

```
  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeleteBucketWebsite中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會停用指定 S3 儲存貯體的靜態網站託管屬性。

```
Remove-S3BucketWebsite -BucketName 's3testbucket'
```

輸出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
 "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteBucketWebsite式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteObject配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteObject。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 使用版本化物件

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

刪除未進行版本控制之 S3 儲存貯體中的物件。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete an object from a non-versioned Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteObject
{
    /// <summary>
    /// The Main method initializes the necessary variables and then calls
    /// the DeleteObjectNonVersionedBucketAsync method to delete the object
    /// named by the keyName parameter.
    /// </summary>
    public static async Task Main()
    {
        const string bucketName = "doc-example-bucket";
        const string keyName = "testfile.txt";

        // If the Amazon S3 bucket is located in an AWS Region other than the
        // Region of the default account, define the AWS Region for the
        // Amazon S3 bucket in your call to the AmazonS3Client constructor.
        // For example RegionEndpoint.USWest2.
```

```
            IAmazonS3 client = new AmazonS3Client();
            await DeleteObjectNonVersionedBucketAsync(client, bucketName,
keyName);
        }

        /// <summary>
        /// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
        /// desired object from the named bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to delete
        /// an object from an Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the bucket from which the
        /// object will be deleted.</param>
        /// <param name="keyName">The name of the object to delete.</param>
        public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
client, string bucketName, string keyName)
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };

                Console.WriteLine($"Deleting object: {keyName}");
                await client.DeleteObjectAsync(deleteObjectRequest);
                Console.WriteLine($"Object: {keyName} deleted from
{bucketName}.");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error encountered on server.
Message:'{ex.Message}' when deleting an object.");
            }
        }
    }
```

刪除已進行版本控制之 S3 儲存貯體中的物件。

```
    using System;
```

```
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example creates an object in an Amazon Simple Storage Service
    /// (Amazon S3) bucket and then deletes the object version that was
    /// created.
    /// </summary>
    public class DeleteObjectVersion
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "verstioned-object.txt";

            // If the AWS Region of the default user is different from the AWS
            // Region of the Amazon S3 bucket, pass the AWS Region of the
            // bucket region to the Amazon S3 client object's constructor.
            // Define it like this:
            //      RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 client = new AmazonS3Client();

            await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// This method creates and then deletes a versioned object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// create and delete the object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// object will be created and deleted.</param>
        /// <param name="keyName">The key name of the object to create.</param>
        public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3
client, string bucketName, string keyName)
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(client, bucketName,
keyName);
```

```
                // Delete the object by specifying an object key and a version
 ID.
                DeleteObjectRequest request = new DeleteObjectRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID,
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// This method is used to create the temporary Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object which will be
 used
        /// to create the temporary Amazon S3 object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// will be created.</param>
        /// <param name="objectKey">The name of the Amazon S3 object co create.</
param>
        /// <returns>The Version ID of the created object.</returns>
        public static async Task<string> PutAnObject(IAmazonS3 client, string
 bucketName, string objectKey)
        {
            PutObjectRequest request = new PutObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                ContentBody = "This is the content body!",
            };

            PutObjectResponse response = await client.PutObjectAsync(request);
            return response.VersionId;
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeleteObject中的。

Bash

AWS CLI 與 Bash 腳本

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function delete_item_in_bucket
#
# This function deletes the specified file from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - The key (file name) in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function delete_item_in_bucket() {
  local bucket_name=$1
  local key=$2
  local response
```

```
  response=$(aws s3api delete-object \
    --bucket "$bucket_name" \
    --key "$key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteObject中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                              const Aws::String &fromBucket,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
            .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
```

```
                    err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考DeleteObject中的。

CLI

AWS CLI

下列指令會從名為的值區刪test.txt除名為的物件my-bucket：

```
aws s3api delete-object --bucket my-bucket --key test.txt
```

如果啟用了值區版本控制，輸出將包含刪除標記的版本 ID：

```
{
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tNDlWlGq",
  "DeleteMarker": true
}
```

如需刪除物件的詳細資訊，請參閱 Amazon S3 開發人員指南中的刪除物件。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteObject中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
 string, versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
    }
   }
  }
```

```
 } else {
  deleted = true
 }
 return deleted, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考DeleteObject中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

刪除物件。

```
import { DeleteObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new DeleteObjectCommand({
    Bucket: "test-bucket",
    Key: "test-key.txt",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeleteObject中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

刪除物件。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def delete(self):
        """
        Deletes the object.
        """
        try:
            self.object.delete()
            self.object.wait_until_not_exists()
            logger.info(
                "Deleted object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't delete object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
```

```
            raise
```

透過刪除較新的物件版本，將物件還原至先前的版本。

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
```

```
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
 f"{object_key}."
        )
```

透過刪除物件啟用的刪除標記，恢復被刪除的物件。

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest
 version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
```

```
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.",
 object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

建立 Lambda 處理常式從 S3 物件中移除刪除標記。此處理常式可有效清除版本化的儲存貯體中無關的刪除標記。

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
```

```
    :return: A result structure that Amazon S3 uses to interpret the result of
 the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
 obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a
 delete
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete
 marker."
            )

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]
["HTTPHeaders"].get(
```

```
                "x-amz-delete-marker", "false"
            )
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
                    # Mark request timeout as a temporary failure so it will be
retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
                        )
                        logger.info(result_string)
                    else:
                        raise
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
                    f"present or is not 'true'."
                )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
```

```
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DeleteObject中的 Python (博托 3) API 參考。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
async fn remove_object(client: &Client, bucket: &str, key: &str) -> Result<(),
 Error> {
    client
        .delete_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await?;

    println!("Object deleted.");

    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK DeleteObject中的 Rust API 參考資料。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
TRY.
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key
    ).
    MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱 AWS SDK DeleteObject中的 SAP ABAP API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
public func deleteFile(bucket: String, key: String) async throws {
```

```
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.deleteObject(input: input)
        } catch {
            throw error
        }
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteObject中的斯威夫特 API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteObjectTagging配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteObjectTagging。

CLI

AWS CLI

若要刪除物件的標籤組

下列delete-object-tagging範例會從物件中刪除具有指定索引鍵的標籤doc1.rtf。

```
aws s3api delete-object-tagging \
    --bucket my-bucket \
    --key doc1.rtf
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteObjectTagging中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會移除與指定 S3 儲存貯體中具有索引鍵 'testfile.txt' 之物件相關聯的所有標籤。

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 's3testbucket' -Select
 '^Key'
```

輸出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target
 "testfile.txt".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
testfile.txt
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteObjectTagging式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteObjects配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteObjects。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用儲存貯體和物件

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

刪除 S3 儲存貯體中的所有物件。

```
        /// <summary>
        /// Delete all of the objects stored in an existing Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket from which the
        /// contents will be deleted.</param>
        /// <returns>A boolean value that represents the success or failure of
        /// deleting all of the objects in the bucket.</returns>
        public static async Task<bool> DeleteBucketContentsAsync(IAmazonS3
client, string bucketName)
        {
            // Iterate over the contents of the bucket and delete all objects.
            var request = new ListObjectsV2Request
            {
                BucketName = bucketName,
            };

            try
            {
                ListObjectsV2Response response;

                do
                {
                    response = await client.ListObjectsV2Async(request);
                    response.S3Objects
                        .ForEach(async obj => await
client.DeleteObjectAsync(bucketName, obj.Key));

                    // If the response is truncated, set the request
ContinuationToken
```

```
                    // from the NextContinuationToken property of the response.
                    request.ContinuationToken = response.NextContinuationToken;
                }
                while (response.IsTruncated);

                return true;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error deleting objects: {ex.Message}");
                return false;
            }
        }
```

刪除未進行版本控制之 S3 儲存貯體中的多個物件。

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete multiple objects from an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
    public class DeleteMultipleObjects
    {
        /// <summary>
        /// The Main method initializes the Amazon S3 client and the name of
        /// the bucket and then passes those values to MultiObjectDeleteAsync.
        /// </summary>
        public static async Task Main()
        {
            const string bucketName = "doc-example-bucket";

            // If the Amazon S3 bucket from which you wish to delete objects is
not
            // located in the same AWS Region as the default user, define the
            // AWS Region for the Amazon S3 bucket as a parameter to the client
            // constructor.
```

```
            IAmazonS3 s3Client = new AmazonS3Client();

            await MultiObjectDeleteAsync(s3Client, bucketName);
        }

        /// <summary>
        /// This method uses the passed Amazon S3 client to first create and then
        /// delete three files from the named bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// Amazon S3 methods.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where
objects
        /// will be created and then deleted.</param>
        public static async Task MultiObjectDeleteAsync(IAmazonS3 client, string
bucketName)
        {
            // Create three sample objects which we will then delete.
            var keysAndVersions = await PutObjectsAsync(client, 3, bucketName);

            // Now perform the multi-object delete, passing the key names and
            // version IDs. Since we are working with a non-versioned bucket,
            // the object keys collection includes null version IDs.
            DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keysAndVersions,
            };

            // You can add a specific object key to the delete request using the
            // AddKey method of the multiObjectDeleteRequest.
            try
            {
                DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException e)
            {
                PrintDeletionErrorStatus(e);
            }
```

```
        }

        /// <summary>
        /// Prints the list of errors raised by the call to DeleteObjectsAsync.
        /// </summary>
        /// <param name="ex">A collection of exceptions returned by the call to
        /// DeleteObjectsAsync.</param>
        public static void PrintDeletionErrorStatus(DeleteObjectsException ex)
        {
            DeleteObjectsResponse errorResponse = ex.Response;
            Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

            Console.WriteLine($"Successfully deleted
{errorResponse.DeletedObjects.Count}.");
            Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");

            Console.WriteLine("Printing error data...");
            foreach (DeleteError deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// This method creates simple text file objects that can be used in
        /// the delete method.
        /// </summary>
        /// <param name="client">The Amazon S3 client used to call
PutObjectAsync.</param>
        /// <param name="number">The number of objects to create.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created.</param>
        /// <returns>A list of keys (object keys) and versions that the calling
        /// method will use to delete the newly created files.</returns>
        public static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, int number, string bucketName)
        {
            List<KeyVersion> keys = new List<KeyVersion>();
            for (int i = 0; i < number; i++)
            {
                string key = "ExampleObject-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
```

```
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                PutObjectResponse response = await
client.PutObjectAsync(request);

                // For non-versioned bucket operations, we only need the
                // object key.
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                };
                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

刪除已進行版本控制之 S3 儲存貯體中的多個物件。

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete objects in a version-enabled Amazon
    /// Simple StorageService (Amazon S3) bucket.
    /// </summary>
    public class DeleteMultipleObjects
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";

            // If the AWS Region for your Amazon S3 bucket is different from
```

```
            // the AWS Region of the default user, define the AWS Region for
            // the Amazon S3 bucket and pass it to the client constructor
            // like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 s3Client;

            s3Client = new AmazonS3Client();
            await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
        }

        /// <summary>
        /// This method removes multiple versions and objects from a
        /// version-enabled Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string
bucketName)
        {
            // Delete objects (specifying object version in the request).
            await DeleteObjectVersionsAsync(client, bucketName);

            // Delete objects (without specifying object version in the request).
            var deletedObjects = await DeleteObjectsAsync(client, bucketName);

            // Additional exercise - remove the delete markers Amazon S3 returned
from
            // the preceding response. This results in the objects reappearing
            // in the bucket (you can verify the appearance/disappearance of
            // objects in the console).
            await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
        }

        /// <summary>
        /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
        /// them again. The method returns a list of the Amazon S3 objects
deleted.
```

```
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PubObjectsAsync and NonVersionedDeleteAsync.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created and then deleted.</param>
        /// <returns>A list of DeletedObjects.</returns>
        public static async Task<List<DeletedObject>>
DeleteObjectsAsync(IAmazonS3 client, string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

            // Delete objects using only keys. Amazon S3 creates a delete marker
and
            // returns its version ID in the response.
            List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
            return deletedObjects;
        }

        /// <summary>
        /// This method creates several temporary objects and then deletes them.
        /// </summary>
        /// <param name="client">The S3 client.</param>
        /// <param name="bucketName">Name of the bucket.</param>
        /// <returns>Async task.</returns>
        public static async Task DeleteObjectVersionsAsync(IAmazonS3 client,
string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

            // Delete the specific object versions.
            await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
        }

        /// <summary>
        /// Displays the list of information about deleted files to the console.
        /// </summary>
        /// <param name="e">Error information from the delete process.</param>
        private static void DisplayDeletionErrors(DeleteObjectsException e)
        {
            var errorResponse = e.Response;
```

```
            Console.WriteLine($"No. of objects successfully deleted =
 {errorResponse.DeletedObjects.Count}");
            Console.WriteLine($"No. of objects failed to delete =
 {errorResponse.DeleteErrors.Count}");
            Console.WriteLine("Printing error data...");
            foreach (var deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
 {deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// Delete multiple objects from a version-enabled bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        private static async Task VersionedDeleteAsync(IAmazonS3 client, string
 bucketName, List<KeyVersion> keys)
        {
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keys, // This includes the object keys and specific
 version IDs.
            };

            try
            {
                Console.WriteLine("Executing VersionedDelete...");
                DeleteObjectsResponse response = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted all the
 {response.DeletedObjects.Count} items");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
```

```
            }
        }

        /// <summary>
        /// Deletes multiple objects from a non-versioned Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        /// <returns>A list of the deleted objects.</returns>
        private static async Task<List<DeletedObject>>
 NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion>
 keys)
        {
            // Create a request that includes only the object key names.
            DeleteObjectsRequest multiObjectDeleteRequest = new
 DeleteObjectsRequest();
            multiObjectDeleteRequest.BucketName = bucketName;

            foreach (var key in keys)
            {
                multiObjectDeleteRequest.AddKey(key.Key);
            }

            // Execute DeleteObjectsAsync.
            // The DeleteObjectsAsync method adds a delete marker for each
            // object deleted. You can verify that the objects were removed
            // using the Amazon S3 console.
            DeleteObjectsResponse response;
            try
            {
                Console.WriteLine("Executing NonVersionedDelete...");
                response = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
 response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException ex)
            {
```

```
                DisplayDeletionErrors(ex);
                throw; // Some deletions failed. Investigate before continuing.
            }

            // This response contains the DeletedObjects list which we use to
delete the delete markers.
            return response.DeletedObjects;
        }

        /// <summary>
        /// Deletes the markers left after deleting the temporary objects.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="deletedObjects">A list of the objects that were
deleted.</param>
        private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client,
string bucketName, List<DeletedObject> deletedObjects)
        {
            var keyVersionList = new List<KeyVersion>();

            foreach (var deletedObject in deletedObjects)
            {
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = deletedObject.Key,
                    VersionId = deletedObject.DeleteMarkerVersionId,
                };
                keyVersionList.Add(keyVersion);
            }

            // Create another request to delete the delete markers.
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keyVersionList,
            };

            // Now, delete the delete marker to bring your objects back to the
bucket.
```

```
            try
            {
                Console.WriteLine("Removing the delete markers .....");
                var deleteObjectResponse = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted the
 {deleteObjectResponse.DeletedObjects.Count} delete markers");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Create temporary Amazon S3 objects to show how object deletion wors
 in an
        /// Amazon S3 bucket with versioning enabled.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// PutObjectAsync to create temporary objects for the example.</param>
        /// <param name="bucketName">A string representing the name of the S3
        /// bucket where we will create the temporary objects.</param>
        /// <param name="number">The number of temporary objects to create.</
param>
        /// <returns>A list of the KeyVersion objects.</returns>
        private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
 client, string bucketName, int number)
        {
            var keys = new List<KeyVersion>();

            for (var i = 0; i < number; i++)
            {
                string key = "ObjectToDelete-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                var response = await client.PutObjectAsync(request);
                KeyVersion keyVersion = new KeyVersion
```

```
                {
                    Key = key,
                    VersionId = response.VersionId,
                };

                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeleteObjects中的。

Bash

AWS CLI 與 Bash 腳本

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
```

```
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[DeleteObjects](#)中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {

 deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
            client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }
```

```
        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考DeleteObjects中的。

CLI

AWS CLI

下列指令會從名為的值區刪除物件my-bucket：

```
aws s3api delete-objects --bucket my-bucket --delete file://delete.json
```

delete.json是指定要刪除之物件的目前目錄中的 JSON 文件：

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

輸出：

```
{
    "Deleted": [
        {
            "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUPO1koHwzU",
            "Key": "test1.txt",
            "DeleteMarker": true
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteObjects中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
```

```
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
     log.Printf("Bucket %s does not exist.\n", bucket)
     err = noBucket
    }
   } else if len(delOut.Errors) > 0 {
    for _, outErr := range delOut.Errors {
     log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
    }
    err = fmt.Errorf("%s", *delOut.Errors[0].Message)
   }
  }
  return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考DeleteObjects中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucketName>

                Where:
                   bucketName - the Amazon S3 bucket name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specfied Amazon S3 bucket.
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putOb;
        ObjectIdentifier objectId;

        for (int i = 0; i < 3; i++) {
            String keyName = "delete object example " + i;
            objectId = ObjectIdentifier.builder()
                    .key(keyName)
                    .build();
```

```
            putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            s3.putObject(putOb, RequestBody.fromString(keyName));
            keys.add(objectId);
        }

        System.out.println(keys.size() + " objects successfully created.");

        // Delete multiple objects in one request.
        Delete del = Delete.builder()
                .objects(keys)
                .build();

        try {
            DeleteObjectsRequest multiObjectDeleteRequest =
 DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(del)
                    .build();

            s3.deleteObjects(multiObjectDeleteRequest);
            System.out.println("Multiple objects are deleted!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DeleteObjects中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

刪除多個物件。

```
import { DeleteObjectsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new DeleteObjectsCommand({
    Bucket: "test-bucket",
    Delete: {
      Objects: [{ Key: "object1.txt" }, { Key: "object2.txt" }],
    },
  });

  try {
    const { Deleted } = await client.send(command);
    console.log(
      `Successfully deleted ${Deleted.length} objects from S3 bucket. Deleted
 objects:`,
    );
    console.log(Deleted.map((d) => ` • ${d.Key}`).join("\n"));
  } catch (err) {
    console.error(err);
  }
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeleteObjects中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteObjects中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

刪除金鑰清單中一整組物件。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (count($check) <= 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考DeleteObjects中的。

PowerShell

適用的工具 PowerShell

示例 1：此命令從存儲桶「測試文件」中刪除對象「sample.txt」。執行指令之前，系統會提示您確認；若要抑制提示，請使用-Force 參數。

```
Remove-S3Object -BucketName test-files -Key sample.txt
```

範例 2：假設值區已設定為啟用物件版本，此命令會從值區「測試檔案」移除物件「sample.txt」的指定版本。

```
Remove-S3Object -BucketName test-files -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

示例 3：此命令從存儲桶「測試文件」中刪除對象「sample1.txt」，「sample2.txt」和「sample3.txt」作為單個批處理操作。無論刪除的成功或錯誤狀態為何，服務回應都會列出所有已處理的金鑰。若只要取得服務無法處理之金鑰的錯誤，請新增-ReportErrorsOnly 參數 (此參數也可以使用別名-Quiet 指定。

```
Remove-S3Object -BucketName test-files -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

範例 4：此範例使用含有-KeyCollection 參數的內嵌運算式來取得要刪除之物件的索引鍵。Get-S3Object返回亞馬遜 .s3.Model.s3 對象實例的集合，每個實例都具有標識對象的字符串類型的關鍵成員。

```
Remove-S3Object -bucketname "test-files" -KeyCollection (Get-S3Object "test-
files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

範例 5：此範例會取得值區中具有索引鍵前置詞「首碼/子首碼」的所有物件，並將其刪除。請注意，傳入物件會一次處理一個。對於大型集合，請考慮將集合傳遞至指令程式的-InputObject (alias-S3ObjectCollection) 參數，以便在單一呼叫服務的批次中執行刪除作業。

```
Get-S3Object -BucketName "test-files" -KeyPrefix "prefix/subprefix" | Remove-
S3Object -Force
```

範例 6：此範例會將代表刪除標記的 Amazon.S3.Model.s3 ObjectVersion 執行個體集合傳送至指令程式以進行刪除。請注意，傳入物件會一次處理一個。對於大型集合，請考慮將集合傳遞至

指令程式的-InputObject (alias-S3ObjectCollection) 參數，以便在單一呼叫服務的批次中執行刪除作業。

```
(Get-S3Version -BucketName "test-files").Versions | Where {$_.IsDeleteMarker -eq
 "True"} | Remove-S3Object -Force
```

範例 7：此指令碼示範如何透過建構要與-KeyAndVersionCollection 參數搭配使用的物件陣列，來執行一組物件的批次刪除 (在本例中為 delete 標記)。

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
 {$_.IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
 $marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -
Force
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteObjects式參考中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用物件金鑰清單刪除一整組物件。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
```

```
        self.key = self.object.key


    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.

        :param bucket: The bucket that contains the objects. This is a Boto3
Bucket
                       resource.
        :param object_keys: The list of keys that identify the objects to remove.
        :return: The response that contains data about which objects were deleted
                 and any that could not be deleted.
        """
        try:
            response = bucket.delete_objects(
                Delete={"Objects": [{"Key": key} for key in object_keys]}
            )
            if "Deleted" in response:
                logger.info(
                    "Deleted objects '%s' from bucket '%s'.",
                    [del_obj["Key"] for del_obj in response["Deleted"]],
                    bucket.name,
                )
            if "Errors" in response:
                logger.warning(
                    "Could not delete objects '%s' from bucket '%s'.",
                    [
                        f"{del_obj['Key']}: {del_obj['Code']}"
                        for del_obj in response["Errors"]
                    ],
                    bucket.name,
                )
        except ClientError:
            logger.exception("Couldn't delete any objects from bucket %s.",
bucket.name)
            raise
        else:
            return response
```

刪除儲存貯體中的所有物件。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def empty_bucket(bucket):
        """
        Remove all objects from a bucket.

        :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
        """
        try:
            bucket.objects.delete()
            logger.info("Emptied bucket '%s'.", bucket.name)
        except ClientError:
            logger.exception("Couldn't empty bucket '%s'.", bucket.name)
            raise
```

會透過刪除其所有版本，永久刪除已建立版本的物件。

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
```

```
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DeleteObjects中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
 # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
 #
 # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
 def delete_bucket(bucket)
   puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
   answer = gets.chomp.downcase
   if answer == "y"
     bucket.objects.batch_delete!
     bucket.delete
     puts("Emptied and deleted bucket #{bucket.name}.\n")
   end
 rescue Aws::Errors::ServiceError => e
   puts("Couldn't empty and delete bucket #{bucket.name}.")
   puts("\t#{e.code}: #{e.message}")
   raise
 end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考DeleteObjects中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
pub async fn delete_objects(client: &Client, bucket_name: &str) ->
 Result<Vec<String>, Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    let mut delete_objects: Vec<ObjectIdentifier> = vec![];
    for obj in objects.contents() {
        let obj_id = ObjectIdentifier::builder()
            .set_key(Some(obj.key().unwrap().to_string()))
            .build()
            .map_err(Error::from)?;
        delete_objects.push(obj_id);
    }

    let return_keys = delete_objects.iter().map(|o| o.key.clone()).collect();

    if !delete_objects.is_empty() {
        client
            .delete_objects()
            .bucket(bucket_name)
            .delete(
                Delete::builder()
                    .set_objects(Some(delete_objects))
                    .build()
                    .map_err(Error::from)?,
            )
            .send()
            .await?;
    }
```

```
    let objects: ListObjectsV2Output =
client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(Error::unhandled(
            "There were still objects left in the bucket.",
        )),
    }
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK DeleteObjects中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public func deleteObjects(bucket: String, keys: [String]) async throws {
        let input = DeleteObjectsInput(
            bucket: bucket,
            delete: S3ClientTypes.Delete(
                objects: keys.map({ S3ClientTypes.ObjectIdentifier(key: $0) }),
                quiet: true
            )
        )

        do {
```

```
            let output = try await client.deleteObjects(input: input)

            // As of the last update to this example, any errors are returned
            // in the `output` object's `errors` property. If there are any
            // errors in this array, throw an exception. Once the error
            // handling is finalized in later updates to the AWS SDK for
            // Swift, this example will be updated to handle errors better.

            guard let errors = output.errors else {
                return  // No errors.
            }
            if errors.count != 0 {
                throw ServiceHandlerError.deleteObjectsError
            }
        } catch {
            throw error
        }
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteObjects中的斯威夫特 API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeletePublicAccessBlock配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeletePublicAccessBlock。

CLI

AWS CLI

刪除值區的區塊公用存取設定

下列delete-public-access-block範例會移除指定值區上的區塊公用存取設定。

```
aws s3api delete-public-access-block \
    --bucket my-bucket
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeletePublicAccessBlock中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會關閉指定值區的封鎖公用存取設定。

```
Remove-S3PublicAccessBlock -BucketName 's3testbucket' -Force -Select
 '^BucketName'
```

輸出：

```
s3testbucket
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeletePublicAccessBlock式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketAccelerateConfiguration配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketAccelerateConfiguration。

CLI

AWS CLI

若要擷取值區的加速組態

下列get-bucket-accelerate-configuration範例會擷取指定值區的加速組態。

```
aws s3api get-bucket-accelerate-configuration \
    --bucket my-bucket
```

輸出：

```
{
    "Status": "Enabled"
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketAccelerateConfiguration中的。

PowerShell

適用的工具 PowerShell

範例 1：如果針對指定值區啟用了傳輸加速設定，此命令會傳回 Enabled 值。

```
Get-S3BucketAccelerateConfiguration -BucketName 's3testbucket'
```

輸出：

```
Value
-----
Enabled
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketAccelerateConfiguration式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketAcl配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketAcl。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 管理存取控制清單 (ACL)

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        /// <summary>
        /// Get the access control list (ACL) for the new bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to get the
        /// access control list (ACL) of the bucket.</param>
        /// <param name="newBucketName">The name of the newly created bucket.</
param>
        /// <returns>An S3AccessControlList.</returns>
        public static async Task<S3AccessControlList>
 GetACLForBucketAsync(IAmazonS3 client, string newBucketName)
        {
            // Retrieve bucket ACL to show that the ACL was properly applied to
            // the new bucket.
            GetACLResponse getACLResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = newBucketName,
            });

            return getACLResponse.AccessControlList;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetBucketAcl中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
```

```
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
            s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                      << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                          << getGranteeTypeString(grantee.GetType()) <<
std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
```

```
                            << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string.
*/

Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string.
*/

Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
```

```
            return "Can list objects in this bucket, create/overwrite/delete "
                   "objects in this bucket, and read/write this "
                   "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }

    return "Permission unknown";
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考GetBucketAcl中的。

CLI

AWS CLI

下列命令會擷取名為的值區的存取控制清單my-bucket：

```
aws s3api get-bucket-acl --bucket my-bucket
```

輸出：

```
{
    "Owner": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
    },
    "Grants": [
        {
            "Grantee": {
```

```
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
            },
            "Permission": "FULL_CONTROL"
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketAcl中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                  <bucketName> <objectKey>

                Where:
                  bucketName - The Amazon S3 bucket to get the access control
list (ACL) for.
                  objectKey - The object to get the ACL for.\s
                """;

       if (args.length != 2) {
           System.out.println(usage);
           System.exit(1);
       }

       String bucketName = args[0];
       String objectKey = args[1];
       System.out.println("Retrieving ACL for object: " + objectKey);
       System.out.println("in bucket: " + bucketName);
       Region region = Region.US_EAST_1;
       S3Client s3 = S3Client.builder()
               .region(region)
               .build();

       getBucketACL(s3, objectKey, bucketName);
       s3.close();
       System.out.println("Done!");
   }

   public static String getBucketACL(S3Client s3, String objectKey, String
bucketName) {
       try {
           GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   .build();

           GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
           List<Grant> grants = aclRes.grants();
           String grantee = "";
           for (Grant grant : grants) {
               System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
               grantee = grant.grantee().id();
```

```
            }

            return grantee;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考GetBucketAcl中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

取得 ACL 許可。

```
import { GetBucketAclCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketAclCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
```

```
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考GetBucketAcl中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_acl(self):
        """
        Get the ACL of the bucket.

        :return: The ACL of the bucket.
        """
        try:
            acl = self.bucket.Acl()
            logger.info(
                "Got ACL for bucket %s. Owner is %s.", self.bucket.name,
 acl.owner
            )
```

```
        except ClientError:
            logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
            raise
        else:
            return acl
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件GetBucketAcl中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketAnalyticsConfiguration配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketAnalyticsConfiguration。

CLI

AWS CLI

若要擷取具有特定 ID 的值區的分析設定

下列get-bucket-analytics-configuration範例會顯示指定值區和 ID 的分析設定。

```
aws s3api get-bucket-analytics-configuration \
    --bucket my-bucket \
    --id 1
```

輸出：

```
{
    "AnalyticsConfiguration": {
        "StorageClassAnalysis": {},
        "Id": "1"
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketAnalyticsConfiguration中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會傳回指定 S3 儲存貯體中名稱為「testfilter」的分析篩選器詳細資料。

```
Get-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId
 'testfilter'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketAnalyticsConfiguration式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketCors配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketCors。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        /// <summary>
        /// Retrieve the CORS configuration applied to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to retrieve the CORS configuration.</param>
        /// <returns>The created CORS configuration object.</returns>
        private static async Task<CORSConfiguration>
RetrieveCORSConfigurationAsync(AmazonS3Client client)
        {
            GetCORSConfigurationRequest request = new
GetCORSConfigurationRequest()
```

```
            {
                BucketName = BucketName,
            };
            var response = await client.GetCORSConfigurationAsync(request);
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
            return configuration;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考[GetBucketCors](#)中的。

CLI

AWS CLI

下列命令會擷取名my-bucket為的值區的跨來源資源共用設定：

```
aws s3api get-bucket-cors --bucket my-bucket
```

輸出：

```
{
    "CORSRules": [
        {
            "AllowedHeaders": [
                "*"
            ],
            "ExposeHeaders": [
                "x-amz-server-side-encryption"
            ],
            "AllowedMethods": [
                "PUT",
                "POST",
                "DELETE"
            ],
            "MaxAgeSeconds": 3000,
            "AllowedOrigins": [
                "http://www.example.com"
            ]
        },
```

```
        {
            "AllowedHeaders": [
                "Authorization"
            ],
            "MaxAgeSeconds": 3000,
            "AllowedMethods": [
                "GET"
            ],
            "AllowedOrigins": [
                "*"
            ]
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketCors中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

取得儲存貯體的 CORS 政策。

```
import { GetBucketCorsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketCorsCommand({
    Bucket: "test-bucket",
  });

  try {
    const { CORSRules } = await client.send(command);
    CORSRules.forEach((cr, i) => {
      console.log(
```

```
        `\nCORSRule ${i + 1}`,
        `\n${"-".repeat(10)}`,
        `\nAllowedHeaders: ${cr.AllowedHeaders.join(" ")}`,
        `\nAllowedMethods: ${cr.AllowedMethods.join(" ")}`,
        `\nAllowedOrigins: ${cr.AllowedOrigins.join(" ")}`,
        `\nExposeHeaders: ${cr.ExposeHeaders.join(" ")}`,
        `\nMaxAgeSeconds: ${cr.MaxAgeSeconds}`,
      );
    });
  } catch (err) {
    console.error(err);
  }
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考GetBucketCors中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
    def get_cors(self):
        """
        Get the CORS rules for the bucket.

        :return The CORS rules for the specified bucket.
        """
        try:
            cors = self.bucket.Cors()
            logger.info(
                "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
 self.bucket.name
            )
        except ClientError:
            logger.exception(("Couldn't get CORS for bucket %s.",
 self.bucket.name))
            raise
        else:
            return cors
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件GetBucketCors中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
```

```
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
 configuration for the bucket.
  def get_cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
 why: #{e.message}"
    nil
  end

end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考GetBucketCors中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketEncryption配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketEncryption。

CLI

AWS CLI

擷取值區的伺服器端加密組態

下列get-bucket-encryption範例會擷取值區的伺服器端加密組態my-bucket。

```
aws s3api get-bucket-encryption \
    --bucket my-bucket
```

輸出：

```
{
    "ServerSideEncryptionConfiguration": {
```

```
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }
        ]
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketEncryption中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會傳回與指定值區相關聯的所有伺服器端加密規則。

```
Get-S3BucketEncryption -BucketName 's3casetestbucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketEncryption式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketInventoryConfiguration配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketInventoryConfiguration。

CLI

AWS CLI

若要擷取值區的庫存組態

下列get-bucket-inventory-configuration範例會擷取具有 ID 之指定值區的詳細目錄組態1。

```
aws s3api get-bucket-inventory-configuration \
```

```
    --bucket my-bucket \
    --id 1
```

輸出：

```
{
    "InventoryConfiguration": {
        "IsEnabled": true,
        "Destination": {
            "S3BucketDestination": {
                "Format": "ORC",
                "Bucket": "arn:aws:s3:::my-bucket",
                "AccountId": "123456789012"
            }
        },
        "IncludedObjectVersions": "Current",
        "Id": "1",
        "Schedule": {
            "Frequency": "Weekly"
        }
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketInventoryConfiguration中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會針對指定的 S3 儲存貯體傳回名為「測試庫存」的詳細資料。

```
Get-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId
 'testinventory'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketInventoryConfiguration式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketLifecycleConfiguration配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketLifecycleConfiguration。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        /// <summary>
        /// Returns a configuration object for the supplied bucket name.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the GetLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">The name of the S3 bucket for which a
        /// configuration will be created.</param>
        /// <returns>Returns a new LifecycleConfiguration object.</returns>
        public static async Task<LifecycleConfiguration>
RetrieveLifecycleConfigAsync(IAmazonS3 client, string bucketName)
        {
            var request = new GetLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetBucketLifecycleConfiguration中的。

CLI

AWS CLI

下列命令會擷取名為的值區的生命週期組態my-bucket：

```
aws s3api get-bucket-lifecycle-configuration --bucket my-bucket
```

輸出：

```
{
    "Rules": [
        {
            "ID": "Move rotated logs to Glacier",
            "Prefix": "rotated/",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Date": "2015-11-10T00:00:00.000Z",
                    "StorageClass": "GLACIER"
                }
            ]
        },
        {
            "Status": "Enabled",
            "Prefix": "",
            "NoncurrentVersionTransitions": [
                {
                    "NoncurrentDays": 0,
                    "StorageClass": "GLACIER"
                }
            ],
            "ID": "Move old versions to Glacier"
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketLifecycleConfiguration中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_lifecycle_configuration(self):
        """
        Get the lifecycle configuration of the bucket.

        :return: The lifecycle rules of the specified bucket.
        """
        try:
            config = self.bucket.LifecycleConfiguration()
            logger.info(
                "Got lifecycle rules %s for bucket '%s'.",
                config.rules,
                self.bucket.name,
            )
        except:
            logger.exception(
                "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
        else:
```

```
            return config.rules
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件GetBucketLifecycleConfiguration中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketLocation配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketLocation。

CLI

AWS CLI

如果存在條件約束my-bucket，下列命令會擷取名為的值區的位置限制：

```
aws s3api get-bucket-location --bucket my-bucket
```

輸出：

```
{
    "LocationConstraint": "us-west-2"
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketLocation中的。

PowerShell

適用的工具 PowerShell

範例 1：如果存在條件約束，此命令會傳回值區 's3testbucket' 的位置限制。

```
Get-S3BucketLocation -BucketName 's3testbucket'
```

輸出：

```
Value
-----
ap-south-1
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketLocation式參考中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
async fn show_buckets(strict: bool, client: &Client, region: &str) -> Result<(),
 Error> {
    let resp = client.list_buckets().send().await?;
    let buckets = resp.buckets();
    let num_buckets = buckets.len();

    let mut in_region = 0;

    for bucket in buckets {
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;

            if r.location_constraint().unwrap().as_ref() == region {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
    }
```

```
    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK GetBucketLocation中的 Rust API 參考資料。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketLogging配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketLogging。

CLI

AWS CLI

若要擷取值區的記錄狀態

下列get-bucket-logging範例會擷取指定值區的記錄狀態。

```
aws s3api get-bucket-logging \
    --bucket my-bucket
```

輸出：

```
{
    "LoggingEnabled": {
        "TargetPrefix": "",
        "TargetBucket": "my-bucket-logs"
          }
```

```
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[GetBucketLogging](#)中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會傳回指定值區的記錄狀態。

```
Get-S3BucketLogging -BucketName 's3testbucket'
```

輸出：

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程[GetBucketLogging](#)式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱[搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`GetBucketMetricsConfiguration`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`GetBucketMetricsConfiguration`。

CLI

AWS CLI

若要擷取具有特定 ID 的值區的指標組態

下列`get-bucket-metrics-configuration`範例顯示指定值區和 ID 的指標組態。

```
aws s3api get-bucket-metrics-configuration \
    --bucket my-bucket \
    --id 123
```

輸出：

```
{
    "MetricsConfiguration": {
        "Filter": {
            "Prefix": "logs"
        },
        "Id": "123"
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketMetricsConfiguration中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會針對指定的 S3 儲存貯體傳回名為 'testfilter' 的指標篩選器的詳細資料。

```
Get-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId
 'testfilter'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketMetricsConfiguration式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketNotification配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketNotification。

CLI

AWS CLI

下列命令會擷取名為的值區的通知組態my-bucket：

```
aws s3api get-bucket-notification --bucket my-bucket
```

輸出：

```
{
    "TopicConfiguration": {
        "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
        "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNl",
        "Event": "s3:ObjectCreated:*",
        "Events": [
            "s3:ObjectCreated:*"
        ]
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketNotification中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例擷取指定值區的通知組態

```
Get-S3BucketNotification -BucketName kt-tools | select -ExpandProperty
 TopicConfigurations
```

輸出：

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketNotification式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketPolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketPolicy。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,
                                 const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
            s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n"
 <<
                  policy_stream.str() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考GetBucketPolicy中的。

CLI

AWS CLI

下列命令會擷取名為的值區原則my-bucket：

```
aws s3api get-bucket-policy --bucket my-bucket
```

輸出：

```
{
    "Policy": "{\"Version\":\"2008-10-17\",\"Statement\":[{\"Sid\":\"\",\"Effect
\":\"Allow\",\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":
\"arn:aws:s3:::my-bucket/*\"},{\"Sid\":\"\",\"Effect\":\"Deny\",\"Principal\":
\"*\",\"Action\":\"s3:GetObject\",\"Resource\":\"arn:aws:s3:::my-bucket/secret/*
\"}]}"
}
```

取得並放置儲存貯體政策下列範例顯示如何下載 Amazon S3 儲存貯體政策、修改檔案，然後使用套用put-bucket-policy修改後的儲存貯體政策。要將存儲桶策略下載到文件中，您可以運行：

```
aws s3api get-bucket-policy --bucket mybucket --query Policy --output text >
 policy.json
```

然後，您可以視需要修改policy.json檔案。最後，您可以執行下列命令，將此修改後的政策套用回 S3 儲存貯體：

policy.json根據需要提供文件。最後，您可以執行下列命令，將此修改後的政策套用回 S3 儲存貯體：

根據需要提供文件。最後，您可以執行下列命令，將此修改後的政策套用回 S3 儲存貯體：

```
aws s3api put-bucket-policy --bucket mybucket --policy file://policy.json
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketPolicy中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to get the policy from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考GetBucketPolicy中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

取得儲存貯體政策。

```
import { GetBucketPolicyCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketPolicyCommand({
    Bucket: "test-bucket",
  });

  try {
    const { Policy } = await client.send(command);
    console.log(JSON.parse(Policy));
  } catch (err) {
    console.error(err);
  }
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考GetBucketPolicy中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK GetBucketPolicy中的 Kotlin API 參考。

PowerShell

適用的工具 PowerShell

範例 1：此命令會輸出與指定 S3 儲存貯體相關聯的儲存貯體政策。

```
Get-S3BucketPolicy -BucketName 's3testbucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketPolicy式參考中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_policy(self):
        """
        Get the security policy of the bucket.

        :return: The security policy of the specified bucket, in JSON format.
        """
        try:
            policy = self.bucket.Policy()
            logger.info(
                "Got policy %s for bucket '%s'.", policy.policy, self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get policy for bucket '%s'.",
 self.bucket.name)
            raise
        else:
            return json.loads(policy.policy)
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件GetBucketPolicy中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def get_policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    nil
  end

end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考GetBucketPolicy中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketPolicyStatus配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketPolicyStatus。

CLI

AWS CLI

若要擷取值區的政策狀態，指出值區是否為公開狀態

下列get-bucket-policy-status範例會擷取值區的政策狀態my-bucket。

```
aws s3api get-bucket-policy-status \
    --bucket my-bucket
```

輸出：

```
{
    "PolicyStatus": {
        "IsPublic": false
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketPolicyStatus中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會傳回指定 S3 儲存貯體的政策狀態，指出儲存貯體是否為公用。

```
Get-S3BucketPolicyStatus -BucketName 's3casetestbucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketPolicyStatus式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketReplication配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketReplication。

CLI

AWS CLI

下列命令會擷取名為的值區的複寫組態my-bucket：

```
aws s3api get-bucket-replication --bucket my-bucket
```

輸出：

```
{
    "ReplicationConfiguration": {
        "Rules": [
            {
                "Status": "Enabled",
                "Prefix": "",
                "Destination": {
                    "Bucket": "arn:aws:s3:::my-bucket-backup",
                    "StorageClass": "STANDARD"
                },
                "ID": "ZmUwNzE4ZmQ4tMjVhOS00MTlkLOGI4NDkzZTIWJjNTUtYTA1"
            }
        ],
        "Role": "arn:aws:iam::123456789012:role/s3-replication-role"
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketReplication中的。

PowerShell

適用的工具 PowerShell

範例 1：傳回在名為 'mybucket' 的值區上設定的複寫組態資訊。

```
Get-S3BucketReplication -BucketName mybucket
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketReplication式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketRequestPayment配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketRequestPayment。

CLI

AWS CLI

若要擷取值區的請求付款組態

下列get-bucket-request-payment範例會擷取指定值區的要求者付費組態。

```
aws s3api get-bucket-request-payment \
    --bucket my-bucket
```

輸出：

```
{
    "Payer": "BucketOwner"
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketRequestPayment中的。

PowerShell

適用的工具 PowerShell

範例 1：傳回名為「mybucket」之值區的請求付款組態。根據預設，值區擁有者會支付值區的下載費用。

```
Get-S3BucketRequestPayment -BucketName mybucket
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketRequestPayment式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketTagging配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketTagging。

CLI

AWS CLI

下列命令會擷取名為的值區的標記組態my-bucket：

```
aws s3api get-bucket-tagging --bucket my-bucket
```

輸出：

```
{
    "TagSet": [
        {
            "Value": "marketing",
            "Key": "organization"
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketTagging中的。

PowerShell

適用的工具 PowerShell

示例 1：此命令返回與給定存儲桶關聯的所有標籤。

```
Get-S3BucketTagging -BucketName 's3casetestbucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketTagging式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketVersioning配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketVersioning。

CLI

AWS CLI

下列指令會擷取名為的值區的版本設定my-bucket：

```
aws s3api get-bucket-versioning --bucket my-bucket
```

輸出：

```
{
    "Status": "Enabled"
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketVersioning中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會傳回與指定值區相關的版本化狀態。

```
Get-S3BucketVersioning -BucketName 's3testbucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketVersioning式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetBucketWebsite配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetBucketWebsite。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
                // Get the website configuration.
                GetBucketWebsiteRequest getRequest = new
GetBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                };
                GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine($"Index document:
{getResponse.WebsiteConfiguration.IndexDocumentSuffix}");
                Console.WriteLine($"Error document:
{getResponse.WebsiteConfiguration.ErrorDocument}");
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetBucketWebsite中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
```

```
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
            s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                  << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult =
 outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                  << std::endl << std::endl
                  << "For bucket '" << bucketName << "':"
                  << std::endl
                  << "Index page : "
                  << websiteResult.GetIndexDocument().GetSuffix()
                  << std::endl
                  << "Error page: "
                  << websiteResult.GetErrorDocument().GetKey()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考GetBucketWebsite中的。

CLI

AWS CLI

下列命令會擷取名為的值區的靜態網站組態my-bucket：

```
aws s3api get-bucket-website --bucket my-bucket
```

輸出：

```
{
    "IndexDocument": {
        "Suffix": "index.html"
    },
    "ErrorDocument": {
        "Key": "error.html"
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetBucketWebsite中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

取得網站組態。

```
import { GetBucketWebsiteCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketWebsiteCommand({
    Bucket: "test-bucket",
  });

  try {
    const { ErrorDocument, IndexDocument } = await client.send(command);
    console.log(
      `Your bucket is set up to host a website. It has an error document:`,
      `${ErrorDocument.Key}, and an index document: ${IndexDocument.Suffix}.`,
    );
  } catch (err) {
```

```
    console.error(err);
  }
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考GetBucketWebsite中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會傳回指定 S3 儲存貯體之靜態網站組態的詳細資料。

```
Get-S3BucketWebsite -BucketName 's3testbucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetBucketWebsite式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetObject配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetObject。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 從儲存貯體中取得物件 (如果其已修改的話)
- 從多區域存取點取得物件
- 開始使用儲存貯體和物件
- 開始使用加密
- 追蹤上傳和下載

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        /// <summary>
        /// Shows how to download an object from an Amazon S3 bucket to the
        /// local computer.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket where the object is
        /// currently stored.</param>
        /// <param name="objectName">The name of the object to download.</param>
        /// <param name="filePath">The path, including filename, where the
        /// downloaded object will be stored.</param>
        /// <returns>A boolean value indicating the success or failure of the
        /// download process.</returns>
        public static async Task<bool> DownloadObjectFromBucketAsync(
            IAmazonS3 client,
            string bucketName,
            string objectName,
            string filePath)
        {
            // Create a GetObject request
            var request = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
            };

            // Issue request and remember to dispose of the response
            using GetObjectResponse response = await
client.GetObjectAsync(request);

            try
            {
```

```
                // Save object to local file
                await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
                return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error saving {objectName}: {ex.Message}");
                return false;
            }
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetObject中的。

Bash

AWS CLI 與 Bash 腳本

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
```

```
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[GetObject](https://docs.aws.amazon.com/goto/aws-cli/s3api-2006-03-01/GetObject)中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/cpp/example_code/s3)。

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
```

```
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
            client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
                  << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考GetObject中的。

CLI

AWS CLI

下列範例使用get-object命令從 Amazon S3 下載物件：

```
aws s3api get-object --bucket text-content --key dir/my_images.tar.bz2
 my_images.tar.bz2
```

請注意，outfile 參數是不使用選項名稱指定的，例如「-outfile」。輸出檔案的名稱必須是指令中的最後一個參數。

下面的例子演示了如--range何使用從對象下載特定的字節範圍。請注意，字節範圍需要以「bytes=」前綴：

```
aws s3api get-object --bucket text-content --key dir/my_data --range
 bytes=8888-9999 my_data_range
```

如需有關擷取物件的詳細資訊，請參閱 Amazon S3 開發人員指南中的取得物件。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetObject中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(bucketName string, objectKey string,
 fileName string) error {
 result, err := basics.S3Client.GetObject(context.TODO(), &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
 objectKey, err)
  return err
 }
 defer result.Body.Close()
 file, err := os.Create(fileName)
 if err != nil {
```

```
  log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
  return err
 }
 defer file.Close()
 body, err := io.ReadAll(result.Body)
 if err != nil {
  log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
 err)
 }
 _, err = file.Write(body)
 return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考GetObject中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用 S3Client 將資料當作位元組陣列讀取。

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName> <path>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                    path - The path where the file is written to.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getObjectBytes(s3, bucketName, keyName, path);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
 keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
```

```
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

使用 S3 TransferManager 將 S3 儲存貯體中的物件下載到本機檔案。檢視完整檔案並測試。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;
```

```
    public Long downloadFile(S3TransferManager transferManager, String
 bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile =
 transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
 downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
 downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

使用 S3Client 讀取屬於某個物件的索引標籤。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    public static void listTags(S3Client s3, String bucketName, String keyName) {
        try {
            GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            GetObjectTaggingResponse tags =
s3.getObjectTagging(getTaggingRequest);
            List<Tag> tagSet = tags.tagSet();
            for (Tag tag : tagSet) {
                System.out.println(tag.key());
                System.out.println(tag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
}
```

使用 S3Client 取得物件的 URL。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
```

```
                .region(region)
                .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for  " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

使用 S3Client 透過使用 S3Presigner 用戶端物件取得物件。

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents a text file.\s
                """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
                .region(region)
                .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
 String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();
```

```
            GetObjectPresignRequest getObjectPresignRequest =
 GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(60))
                    .getObjectRequest(getObjectRequest)
                    .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
 presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
 presignedGetObjectRequest.url().openConnection();
            presignedGetObjectRequest.httpRequest().headers().forEach((header,
 values) -> {
                values.forEach(value -> {
                    connection.addRequestProperty(header, value);
                });
            });

            // Send any request payload that the service needs (not needed when
            // isBrowserExecutable is true).
            if (presignedGetObjectRequest.signedPayload().isPresent()) {
                connection.setDoOutput(true);

                try (InputStream signedPayload =
 presignedGetObjectRequest.signedPayload().get().asInputStream();
                        OutputStream httpOutputStream =
 connection.getOutputStream()) {
                    IoUtils.copy(signedPayload, httpOutputStream);
                }
            }

            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                System.out.println("Service returned response: ");
                IoUtils.copy(content, System.out);
            }

        } catch (S3Exception | IOException e) {
            e.getStackTrace();
        }
    }
}
```

通過使用對象和 S3 客戶端獲取 ResponseTransformer 對象。

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetDataResponseTransformer {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName> <path>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                    path - The path where the file is written to.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
 keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考GetObject中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

下載物件。

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetObjectCommand({
    Bucket: "test-bucket",
    Key: "hello-s3.txt",
  });

  try {
    const response = await client.send(command);
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log(str);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考GetObject中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK GetObject中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

取得物件。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考GetObject中的。

PowerShell

適用的工具 PowerShell

示例 1：此命令從存儲桶「測試文件」中檢索項目「sample.txt」，並將其保存到當前位置名為「local-sample.txt」的文件中。呼叫此命令之前，檔案「local-sample.txt」不一定要存在。

```
Read-S3Object -BucketName test-files -Key sample.txt -File local-sample.txt
```

示例 2：此命令從存儲桶「測試文件」中檢索虛擬目錄「DIR」，並將其保存到當前位置名為「本地目錄」的文件夾中。在調用此命令之前，文件夾「本地目錄」不必存在。

```
Read-S3Object -BucketName test-files -KeyPrefix DIR -Folder Local-DIR
```

範例 3：從值區名稱中具有「config」的值區中，將所有索引鍵以「.json」結尾的物件下載到指定資料夾中的檔案。物件索引鍵可用來設定檔案名稱。

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetObject式參考中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get(self):
        """
        Gets the object.
```

```
        :return: The object data in bytes.
        """
        try:
            body = self.object.get()["Body"].read()
            logger.info(
                "Got object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
        else:
            return body
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件GetObject中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

取得物件。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
```

```
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is
 downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes} downloaded to
 #{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

取得物件並報告其伺服器端的加密狀態。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
```

```
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? "no" :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考GetObject中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
async fn get_object(client: Client, opt: Opt) -> Result<usize, anyhow::Error> {
    trace!("bucket:      {}", opt.bucket);
    trace!("object:      {}", opt.object);
    trace!("destination: {}", opt.destination.display());

    let mut file = File::create(opt.destination.clone())?;

    let mut object = client
        .get_object()
        .bucket(opt.bucket)
        .key(opt.object)
        .send()
        .await?;

    let mut byte_count = 0_usize;
    while let Some(bytes) = object.body.try_next().await? {
        let bytes_len = bytes.len();
        file.write_all(&bytes)?;
        trace!("Intermediate write of {bytes_len}");
        byte_count += bytes_len;
    }

    Ok(byte_count)
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK GetObject中的 Rust API 參考資料。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   TRY.
       oo_result = lo_s3->getobject(           " oo_result is returned for
testing purposes. "
                   iv_bucket = iv_bucket_name
                   iv_key = iv_object_key
                 ).
       DATA(lv_object_data) = oo_result->get_body( ).
       MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
     CATCH /aws1/cx_s3_nosuchkey.
       MESSAGE 'Object key does not exist.' TYPE 'E'.
   ENDTRY.
```

- 如需 API 詳細資訊，請參閱 AWS SDK GetObject中的 SAP ABAP API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

從儲存貯體將物件下載至本機檔案。

```
    public func downloadFile(bucket: String, key: String, to: String) async
throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the data stream object. Return immediately if there isn't one.
        guard let body = output.body,
              let data = try await body.readData() else {
            return
        }
        try data.write(to: fileUrl)
    }
```

將物件讀取至 Swift 資料物件。

```
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the stream and return its contents in a `Data` object. If
        // there is no stream, return an empty `Data` object instead.
        guard let body = output.body,
              let data = try await body.readData() else {
            return "".data(using: .utf8)!
        }
```

```
        return data
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK GetObject中的斯威夫特 API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetObjectAcl配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetObjectAcl。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 管理存取控制清單 (ACL)

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
```

```
            s3Client.GetObjectAcl(request);

   if (!outcome.IsSuccess()) {
       const Aws::S3::S3Error &err = outcome.GetError();
       std::cerr << "Error: getObjectAcl: "
                 << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
   } else {
       Aws::Vector<Aws::S3::Model::Grant> grants =
               outcome.GetResult().GetGrants();

       for (auto it = grants.begin(); it != grants.end(); it++) {
           std::cout << "For object " << objectKey << ": "
                     << std::endl << std::endl;

           Aws::S3::Model::Grant grant = *it;
           Aws::S3::Model::Grantee grantee = grant.GetGrantee();

           if (grantee.TypeHasBeenSet()) {
               std::cout << "Type:          "
                         << getGranteeTypeString(grantee.GetType()) <<
std::endl;
           }

           if (grantee.DisplayNameHasBeenSet()) {
               std::cout << "Display name:  "
                         << grantee.GetDisplayName() << std::endl;
           }

           if (grantee.EmailAddressHasBeenSet()) {
               std::cout << "Email address: "
                         << grantee.GetEmailAddress() << std::endl;
           }

           if (grantee.IDHasBeenSet()) {
               std::cout << "ID:            "
                         << grantee.GetID() << std::endl;
           }

           if (grantee.URIHasBeenSet()) {
               std::cout << "URI:           "
                         << grantee.GetURI() << std::endl;
           }
```

```
            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string
*/
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string
*/
Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can read this object's data and its metadata, "
                   "and read/write this object's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
```

```
            return "Can read this object's data and its metadata";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this object's permissions";
            // case Aws::S3::Model::Permission::WRITE // Not applicable.
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this object's permissions";
        default:
            return "Permission unknown";
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考GetObjectAcl中的。

CLI

AWS CLI

下列命令會擷取值區中名為之物件的存取控制清單my-bucket：

```
aws s3api get-object-acl --bucket my-bucket --key index.html
```

輸出：

```
{
    "Owner": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
```

```
            },
            "Permission": "READ"
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetObjectAcl中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK GetObjectAcl中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get_acl(self):
        """
        Gets the ACL of the object.

        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()
            logger.info(
                "Got ACL for object %s owned by %s.",
                self.object.key,
                acl.owner["DisplayName"],
            )
        except ClientError:
            logger.exception("Couldn't get ACL for object %s.", self.object.key)
            raise
        else:
            return acl
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件GetObjectAcl中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetObjectLegalHold配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetObjectLegalHold。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 取得物件的合法保留組態
- 鎖定 Amazon S3 對象

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/// <summary>
/// Get the legal hold details for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object legal hold details.</returns>
public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
    string objectKey)
{
    try
    {
```

```
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
 {bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetObjectLegalHold中的。

CLI

AWS CLI

擷取物件的「合法保留」狀態

下列get-object-legal-hold範例會擷取指定物件的「合法保留」狀態。

```
aws s3api get-object-legal-hold \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf
```

輸出：

```
{
    "LegalHold": {
        "Status": "ON"
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetObjectLegalHold中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
```

```
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考GetObjectLegalHold中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   // Get the legal hold details for an S3 object.
   public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
       try {
           GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
               .bucket(bucketName)
               .key(objectKey)
               .build();

           GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
           System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
               ":\n\tStatus: " + response.legalHold().status());
```

```
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
        }

        return null;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考GetObjectLegalHold中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetObjectLockConfiguration配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetObjectLockConfiguration。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 鎖定 Amazon S3 對象

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Get the object lock configuration details for an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to get details.</param>
    /// <returns>The bucket's object lock configuration details.</returns>
```

```
   public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
   {
       try
       {
           var request = new GetObjectLockConfigurationRequest()
           {
               BucketName = bucketName
           };

           var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
           Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                             $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                             $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

           return response.ObjectLockConfiguration;
       }
       catch (AmazonS3Exception ex)
       {
           Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
           return new ObjectLockConfiguration();
       }
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考[GetObjectLockConfiguration](中的。

CLI

AWS CLI

若要擷取值區的物件鎖定組態

下列get-object-lock-configuration範例會擷取指定值區的物件鎖定組態。

```
aws s3api get-object-lock-configuration \
    --bucket my-bucket-with-object-lock
```

輸出：

```
{
    "ObjectLockConfiguration": {
        "ObjectLockEnabled": "Enabled",
        "Rule": {
            "DefaultRetention": {
                "Mode": "COMPLIANCE",
                "Days": 50
            }
        }
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetObjectLockConfiguration中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
```

```
 Bucket: aws.String(bucket),
}

output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
if err != nil {
 var noBucket *types.NoSuchBucket
 var apiErr *smithy.GenericAPIError
 if errors.As(err, &noBucket) {
  log.Printf("Bucket %s does not exist.\n", bucket)
  err = noBucket
 } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
"ObjectLockConfigurationNotFoundError" {
  log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
  err = nil
 }
} else {
 lockConfig = output.ObjectLockConfiguration
}

return lockConfig, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考GetObjectLockConfiguration中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   // Get the object lock configuration details for an S3 bucket.
   public void getBucketObjectLockConfiguration(String bucketName) {
       GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
           .bucket(bucketName)
           .build();
```

```
        GetObjectLockConfigurationResponse response =
 getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
 "+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
 response.objectLockConfiguration().rule().defaultRetention());
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[GetObjectLockConfiguration](中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫]。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import {
  GetObjectLockConfigurationCommand,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 */
export const main = async (client, bucketName) => {
  const command = new GetObjectLockConfigurationCommand({
    Bucket: bucketName,
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
```

```
  });

  try {
    const { ObjectLockConfiguration } = await client.send(command);
    console.log(`Object Lock Configuration: ${ObjectLockConfiguration}`);
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME");
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考GetObjectLockConfiguration中的。

PowerShell

適用的工具 PowerShell

範例 1：如果已為指定 S3 儲存貯體啟用物件鎖定組態，此命令會傳回值「已啟用」。

```
Get-S3ObjectLockConfiguration -BucketName 's3buckettesting' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

輸出：

```
Value
-----
Enabled
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetObjectLockConfiguration式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetObjectRetention配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetObjectRetention。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 鎖定 Amazon S3 對象

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in
{bucketName}: " +
                              $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
```

```
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock retention:
 '{ex.Message}'");
            return new ObjectLockRetention();
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetObjectRetention中的。

CLI

AWS CLI

若要擷取物件的物件保留組態

下列get-object-retention範例會擷取指定物件的物件保留組態。

```
aws s3api get-object-retention \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf
```

輸出：

```
{
    "Retention": {
        "Mode": "GOVERNANCE",
        "RetainUntilDate": "2025-01-01T00:00:00.000Z"
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetObjectRetention中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectRetention retrieves the object retention configuration for an S3
 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
```

```
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考GetObjectRetention中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            return null;
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考GetObjectRetention中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import { GetObjectRetentionCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new GetObjectRetentionCommand({
    Bucket: bucketName,
    Key: objectKey,
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const { Retention } = await client.send(command);
    console.log(`Object Retention Settings: ${Retention.Status}`);
  } catch (err) {
    console.error(err);
```

```
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME", "OBJECT_KEY");
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考GetObjectRetention中的。

PowerShell

適用的工具 PowerShell

示例 1：該命令返回模式和日期，直到對象將被保留。

```
Get-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetObjectRetention式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetObjectTagging配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetObjectTagging。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用索引標籤

CLI

AWS CLI

若要擷取附加至物件的標籤

下列get-object-tagging範例會從指定的物件擷取指定索引鍵的值。

```
aws s3api get-object-tagging \
    --bucket my-bucket \
    --key doc1.rtf
```

輸出：

```
{
    "TagSet": [
        {
            "Value": "confidential",
            "Key": "designation"
        }
    ]
}
```

下列get-object-tagging範例會嘗試擷取沒有標籤的物件doc2.rtf標籤組。

```
aws s3api get-object-tagging \
    --bucket my-bucket \
    --key doc2.rtf
```

輸出：

```
{
    "TagSet": []
}
```

下列get-object-tagging範例會擷取具有多個標籤之物件doc3.rtf的標籤組。

```
aws s3api get-object-tagging \
    --bucket my-bucket \
    --key doc3.rtf
```

輸出：

```
{
    "TagSet": [
```

```
        {
            "Value": "confidential",
            "Key": "designation"
        },
        {
            "Value": "finance",
            "Key": "department"
        },
        {
            "Value": "payroll",
            "Key": "team"
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetObjectTagging中的。

PowerShell

適用的工具 PowerShell

範例 1：範例會傳回與指定 S3 儲存貯體上存在的物件相關聯的標籤。

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'testbucket123'
```

輸出：

```
Key  Value
---  -----
test value
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetObjectTagging式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetPublicAccessBlock配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetPublicAccessBlock。

CLI

AWS CLI

設定或修改值區的區塊公用存取設定

下列get-public-access-block範例會顯示指定值區的區塊公用存取設定。

```
aws s3api get-public-access-block \
    --bucket my-bucket
```

輸出：

```
{
    "PublicAccessBlockConfiguration": {
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "BlockPublicAcls": true,
        "RestrictPublicBuckets": true
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetPublicAccessBlock中的。

PowerShell

適用的工具 PowerShell

範例 1：命令會傳回指定 S3 儲存貯體的公用存取區塊組態。

```
Get-S3PublicAccessBlock -BucketName 's3testbucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetPublicAccessBlock式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭HeadBucket配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用HeadBucket。

Bash

AWS CLI 與 Bash 腳本

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
##############################################################################
# function bucket_exists
#
# This function checks to see if the specified bucket already exists.
#
# Parameters:
#       $1 - The name of the bucket to check.
#
# Returns:
#       0 - If the bucket already exists.
#       1 - If the bucket doesn't exist.
##############################################################################
function bucket_exists() {
  local bucket_name
  bucket_name=$1

  # Check whether the bucket already exists.
  # We suppress all output - we're interested only in the return code.

  if aws s3api head-bucket \
    --bucket "$bucket_name" \
    >/dev/null 2>&1; then
    return 0 # 0 in Bash script means true.
  else
    return 1 # 1 in Bash script means false.
  fi
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考HeadBucket中的。

CLI

AWS CLI

下列指令會驗證儲存貯體的存取權：my-bucket

```
aws s3api head-bucket --bucket my-bucket
```

如果存在值區且您可以存取該值區，則不會傳回任何輸出。否則，將顯示錯誤消息。例如：

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考HeadBucket中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(bucketName string) (bool, error) {
 _, err := basics.S3Client.HeadBucket(context.TODO(), &s3.HeadBucketInput{
```

```
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
    log.Printf("Either you don't have access to bucket %v or another error
 occurred. "+
     "Here's what happened: %v\n", bucketName, err)
   }
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考HeadBucket中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
```

```
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def exists(self):
        """
        Determine whether the bucket exists and you have access to it.

        :return: True when the bucket exists; otherwise, False.
        """
        try:
            self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
            logger.info("Bucket %s exists.", self.bucket.name)
            exists = True
        except ClientError:
            logger.warning(
                "Bucket %s doesn't exist or you don't have access to it.",
                self.bucket.name,
            )
            exists = False
        return exists
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件HeadBucket中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭HeadObject配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用HeadObject。

CLI

AWS CLI

下列命令會擷取值區中名為之物件的中繼資料my-bucket：

```
aws s3api head-object --bucket my-bucket --key index.html
```

輸出：

```
{
    "AcceptRanges": "bytes",
    "ContentType": "text/html",
    "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
    "ContentLength": 77,
    "VersionId": "null",
    "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
    "Metadata": {}
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考HeadObject中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

判斷物件的內容類型。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    public static void getContentType(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

取得物件的還原狀態。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }
```

```
    public static void checkStatus(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
 response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考HeadObject中的。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end
```

```
  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object
 #{@object.bucket.name}:#{@object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考HeadObject中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListBucketAnalyticsConfigurations配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListBucketAnalyticsConfigurations。

CLI

AWS CLI

擷取值區的分析設定清單

以下內容list-bucket-analytics-configurations會擷取指定值區的分析組態清單。

```
aws s3api list-bucket-analytics-configurations \
    --bucket my-bucket
```

輸出：

```
{
    "AnalyticsConfigurationList": [
        {
            "StorageClassAnalysis": {},
            "Id": "1"
        }
    ],
    "IsTruncated": false
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListBucketAnalyticsConfigurations中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會傳回指定 S3 儲存貯體的前 100 個分析組態。

```
Get-S3BucketAnalyticsConfigurationList -BucketName 's3casetestbucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListBucketAnalyticsConfigurations式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListBucketInventoryConfigurations配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListBucketInventoryConfigurations。

CLI

AWS CLI

擷取值區的庫存組態清單

下列list-bucket-inventory-configurations範例會列出指定值區的庫存組態。

```
aws s3api list-bucket-inventory-configurations \
    --bucket my-bucket
```

輸出：

```
{
    "InventoryConfigurationList": [
        {
            "IsEnabled": true,
            "Destination": {
                "S3BucketDestination": {
                    "Format": "ORC",
                    "Bucket": "arn:aws:s3:::my-bucket",
                    "AccountId": "123456789012"
                }
            },
            "IncludedObjectVersions": "Current",
            "Id": "1",
            "Schedule": {
                "Frequency": "Weekly"
            }
        },
        {
            "IsEnabled": true,
            "Destination": {
                "S3BucketDestination": {
                    "Format": "CSV",
                    "Bucket": "arn:aws:s3:::my-bucket",
                    "AccountId": "123456789012"
                }
            },
            "IncludedObjectVersions": "Current",
            "Id": "2",
            "Schedule": {
                "Frequency": "Daily"
            }
        }
    ],
    "IsTruncated": false
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListBucketInventoryConfigurations中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會傳回指定 S3 儲存貯體的前 100 個庫存組態。

```
Get-S3BucketInventoryConfigurationList -BucketName 's3testbucket'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListBucketInventoryConfigurations式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListBuckets配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListBuckets。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
namespace ListBucketsExample
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example uses the AWS SDK for .NET to list the Amazon Simple Storage
```

```
    /// Service (Amazon S3) buckets belonging to the default account.
    /// </summary>
    public class ListBuckets
    {
        private static IAmazonS3 _s3Client;

        /// <summary>
        /// Get a list of the buckets owned by the default user.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <returns>The response from the ListingBuckets call that contains a
        /// list of the buckets owned by the default user.</returns>
        public static async Task<ListBucketsResponse> GetBuckets(IAmazonS3
 client)
        {
            return await client.ListBucketsAsync();
        }

        /// <summary>
        /// This method lists the name and creation date for the buckets in
        /// the passed List of S3 buckets.
        /// </summary>
        /// <param name="bucketList">A List of S3 bucket objects.</param>
        public static void DisplayBucketList(List<S3Bucket> bucketList)
        {
            bucketList
                .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
 created on: {b.CreationDate}"));
        }

        public static async Task Main()
        {
            // The client uses the AWS Region of the default user.
            // If the Region where the buckets were created is different,
            // pass the Region to the client constructor. For example:
            // _s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
            _s3Client = new AmazonS3Client();
            var response = await GetBuckets(_s3Client);
            DisplayBucketList(response.Buckets);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListBuckets中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
 buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考ListBuckets中的。

CLI

AWS CLI

以下命令使用命`list-buckets`令顯示所有 Amazon S3 儲存貯體 (跨所有區域) 的名稱：

```
aws s3api list-buckets --query "Buckets[].Name"
```

查詢選項會篩選儲存貯體名稱的`list-buckets`輸出。

如需有關儲存貯體的詳細資訊，請參閱 Amazon S3 開發人員指南中的使用 Amazon S3 儲存貯體。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListBuckets中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets() ([]types.Bucket, error) {
 result, err := basics.S3Client.ListBuckets(context.TODO(),
 &s3.ListBucketsInput{})
```

```
 var buckets []types.Bucket
 if err != nil {
  log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
 } else {
  buckets = result.Buckets
 }
 return buckets, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考ListBuckets中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
```

```
            .region(region)
            .build();

        listAllBuckets(s3);

    }
    public static void listAllBuckets(S3Client s3) {
        ListBucketsResponse response = s3.listBuckets();
        List<Bucket> bucketList = response.buckets();
        for (Bucket bucket: bucketList) {
            System.out.println("Bucket name "+bucket.name());
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考ListBuckets中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

列出儲存貯體。

```
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new ListBucketsCommand({});

  try {
    const { Owner, Buckets } = await client.send(command);
    console.log(
      `${Owner.DisplayName} owns ${Buckets.length} bucket${
        Buckets.length === 1 ? "" : "s"
```

```
      }:`,
    );
    console.log(`${Buckets.map((b) => ` • ${b.Name}`).join("\n")}`);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ListBuckets中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會傳回所有 S3 儲存貯體。

```
Get-S3Bucket
```

示例 2：此命令返回名為「測試文件」的存儲桶

```
Get-S3Bucket -BucketName test-files
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListBuckets式參考中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
```

```
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    @staticmethod
    def list(s3_resource):
        """
        Get the buckets in all Regions for the current account.

        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in
Boto3
                            that contains collections and factory methods to
create
                            other high-level S3 sub-resources.
        :return: The list of buckets.
        """
        try:
            buckets = list(s3_resource.buckets.all())
            logger.info("Got buckets: %s.", buckets)
        except ClientError:
            logger.exception("Couldn't get buckets.")
            raise
        else:
            return buckets
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListBuckets中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
require "aws-sdk-s3"

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts "Found these buckets:"
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考ListBuckets中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
async fn show_buckets(strict: bool, client: &Client, region: &str) -> Result<(),
 Error> {
    let resp = client.list_buckets().send().await?;
    let buckets = resp.buckets();
    let num_buckets = buckets.len();

    let mut in_region = 0;

    for bucket in buckets {
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;

            if r.location_constraint().unwrap().as_ref() == region {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
```

```
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK ListBuckets中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// Return an array containing information about every available bucket.
    ///
    /// - Returns: An array of ``S3ClientTypes.Bucket`` objects describing
    ///   each bucket.
    public func getAllBuckets() async throws -> [S3ClientTypes.Bucket] {
        let output = try await client.listBuckets(input: ListBucketsInput())

        guard let buckets = output.buckets else {
            return []
        }
        return buckets
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListBuckets中的斯威夫特 API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListMultipartUploads配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListMultipartUploads。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 刪除不完整的分段上傳

CLI

AWS CLI

下列指令會列出名my-bucket為的值區的所有作用中分段上傳：

```
aws s3api list-multipart-uploads --bucket my-bucket
```

輸出：

```
{
    "Uploads": [
        {
            "Initiator": {
                "DisplayName": "username",
                "ID": "arn:aws:iam::0123456789012:user/username"
            },
            "Initiated": "2015-06-02T18:01:30.000Z",
            "UploadId":
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
            "StorageClass": "STANDARD",
            "Key": "multipart/01",
            "Owner": {
                "DisplayName": "aws-account-name",
                "ID":
 "100719349fc3b6dcd7c820a124bf7aecd408092c3d7b51b38494939801fc248b"
            }
        }
    ],
```

```
    "CommonPrefixes": []
}
```

進行中的分段上傳會在 Amazon S3 產生儲存成本。完成或中止作用中的分段上傳，以便從您的帳戶中移除其部分。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListMultipartUploads中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s
```

```
                Where:
                    bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                System.out.println("Upload in progress: Key = \"" + upload.key()
 + "\", id = " + upload.uploadId());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考ListMultipartUploads中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListObjectVersions配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListObjectVersions。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 使用版本化物件

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example lists the versions of the objects in a version enabled
    /// Amazon Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class ListObjectVersions
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";

            // If the AWS Region where your bucket is defined is different from
            // the AWS Region where the Amazon S3 bucket is defined, pass the
constant
            // for the AWS Region to the client constructor like this:
```

```
            //      var client = new AmazonS3Client(RegionEndpoint.USWest2);
            IAmazonS3 client = new AmazonS3Client();
            await GetObjectListWithAllVersionsAsync(client, bucketName);
        }

        /// <summary>
        /// This method lists all versions of the objects within an Amazon S3
        /// version enabled bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// ListVersionsAsync.</param>
        /// <param name="bucketName">The name of the version enabled Amazon S3
 bucket
        /// for which you want to list the versions of the contained objects.</
param>
        public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3
 client, string bucketName)
        {
            try
            {
                // When you instantiate the ListVersionRequest, you can
                // optionally specify a key name prefix in the request
                // if you want a list of object versions of a specific object.

                // For this example we set a small limit in MaxKeys to return
                // a small list of versions.
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    MaxKeys = 2,
                };

                do
                {
                    ListVersionsResponse response = await
 client.ListVersionsAsync(request);

                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
                        Console.WriteLine($"key: {entry.Key} size:
 {entry.Size}");
                    }
```

```
                    // If response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.KeyMarker = response.NextKeyMarker;
                        request.VersionIdMarker = response.NextVersionIdMarker;
                    }
                    else
                    {
                        request = null;
                    }
                }
                while (request != null);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}'");
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListObjectVersions中的。

CLI

AWS CLI

下列命令會擷取名為的值區中物件的版本資訊my-bucket：

```
aws s3api list-object-versions --bucket my-bucket --prefix index.html
```

輸出：

```
{
    "DeleteMarkers": [
        {
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
```

```
            },
            "IsLatest": true,
            "VersionId": "B2VsEK5saUNNHKcOAJj7hIE86RozToyq",
            "Key": "index.html",
            "LastModified": "2015-11-10T00:57:03.000Z"
        },
        {
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
            "Key": "index.html",
            "LastModified": "2015-11-09T23:32:20.000Z"
        }
    ],
    "Versions": [
        {
            "LastModified": "2015-11-10T00:20:11.000Z",
            "VersionId": "Rb_l2T8UHDkFEwCgJjhlgPOZC0qJ.vpD",
            "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 38
        },
        {
            "LastModified": "2015-11-09T23:26:41.000Z",
            "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAFf",
            "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
```

```
            "IsLatest": false,
            "Size": 38
        },
        {
            "LastModified": "2015-11-09T22:50:50.000Z",
            "VersionId": "null",
            "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 533823
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[ListObjectVersions](https://docs.aws.amazon.com/goto/aws-cli/s3-2006-03-01/ListObjectVersions)中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/gov2)。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ListObjectVersions lists all versions of all objects in a bucket.
```

```
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考ListObjectVersions中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;

    for version in resp.versions() {
        println!("{}", version.key().unwrap_or_default());
```

```
        println!("  version ID: {}", version.version_id().unwrap_or_default());
        println!();
    }

    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK ListObjectVersions中的 Rust API 參考資料。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListObjects配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListObjects。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 建立列出 Amazon S3 物件的網頁

CLI

AWS CLI

下列範例會使用list-objects命令來顯示指定值區中所有物件的名稱：

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size:
 Size}'
```

此範例使用--query引數將輸出篩選為每個物件的list-objects索引鍵值和大小

如需有關物件的詳細資訊，請參閱 Amazon S3 開發人員指南中的使用 Amazon S3 物件。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListObjects中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會擷取值區「測試檔案」中所有項目的相關資訊。

```
Get-S3Object -BucketName test-files
```

示例 2：此命令從存儲桶「測試文件」中檢索有關項目「sample.txt」的信息。

```
Get-S3Object -BucketName test-files -Key sample.txt
```

示例 3：此命令從存儲桶「測試文件」中檢索有關前綴為「sample」的所有項目的信息。

```
Get-S3Object -BucketName test-files -KeyPrefix sample
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListObjects式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListObjectsV2配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListObjectsV2。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用儲存貯體和物件

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        /// <summary>
        /// Shows how to list the objects in an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket for which to list
        /// the contents.</param>
        /// <returns>A boolean value indicating the success or failure of the
        /// copy operation.</returns>
        public static async Task<bool> ListBucketContentsAsync(IAmazonS3 client,
 string bucketName)
        {
            try
            {
                var request = new ListObjectsV2Request
                {
                    BucketName = bucketName,
                    MaxKeys = 5,
                };

                Console.WriteLine("--------------------------------------");
                Console.WriteLine($"Listing the contents of {bucketName}:");
                Console.WriteLine("--------------------------------------");

                ListObjectsV2Response response;

                do
                {
                    response = await client.ListObjectsV2Async(request);

                    response.S3Objects
                        .ForEach(obj => Console.WriteLine($"{obj.Key,-35}
{obj.LastModified.ToShortDateString(),10}{obj.Size,10}"));

                    // If the response is truncated, set the request
 ContinuationToken
                    // from the NextContinuationToken property of the response.
                    request.ContinuationToken = response.NextContinuationToken;
                }
                while (response.IsTruncated);

                return true;
            }
```

```
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error encountered on server.
 Message:'{ex.Message}' getting list of objects.");
                return false;
            }
        }
```

使用分頁程式列出物件。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// The following example lists objects in an Amazon Simple Storage
    /// Service (Amazon S3) bucket.
    /// </summary>
    public class ListObjectsPaginator
    {
        private const string BucketName = "doc-example-bucket";

        public static async Task Main()
        {
            IAmazonS3 s3Client = new AmazonS3Client();

            Console.WriteLine($"Listing the objects contained in {BucketName}:
\n");
            await ListingObjectsAsync(s3Client, BucketName);
        }

        /// <summary>
        /// This method uses a paginator to retrieve the list of objects in an
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the S3 bucket whose objects
        /// you want to list.</param>
        public static async Task ListingObjectsAsync(IAmazonS3 client, string
 bucketName)
```

```
        {
            var listObjectsV2Paginator = client.Paginators.ListObjectsV2(new
 ListObjectsV2Request
            {
                BucketName = bucketName,
            });

            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                Console.WriteLine($"HttpStatusCode: {response.HttpStatusCode}");
                Console.WriteLine($"Number of Keys: {response.KeyCount}");
                foreach (var entry in response.S3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }
        }
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK for .NET API 參考中的 ListObjectsV2。

Bash

AWS CLI 與 Bash 腳本

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}
```

```
##############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的 ListObjectsV2。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3Client.ListObjectsV2(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                    outcome.GetResult().GetContents();

            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
    } while (!continuationToken.empty());

    std::cout << allObjects.size() << " object(s) found:" << std::endl;
```

```
    for (const auto &object: allObjects) {
        std::cout << "  " << object.GetKey() << std::endl;
    }

    return true;
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK for C++ API 參考中的 ListObjectsV2。

CLI

AWS CLI

若要取得值區中的物件清單

下列list-objects-v2範例會列出指定值區中的物件。

```
aws s3api list-objects-v2 \
    --bucket my-bucket
```

輸出：

```
{
    "Contents": [
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"621503c373607d548b37cff8778d992c\"",
            "StorageClass": "STANDARD",
            "Key": "doc1.rtf",
            "Size": 391
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"a2cecc36ab7c7fe3a71a273b9d45b1b5\"",
            "StorageClass": "STANDARD",
            "Key": "doc2.rtf",
            "Size": 373
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"08210852f65a2e9cb999972539a64d68\"",
```

```
            "StorageClass": "STANDARD",
            "Key": "doc3.rtf",
            "Size": 399
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"d1852dd683f404306569471af106988e\"",
            "StorageClass": "STANDARD",
            "Key": "doc4.rtf",
            "Size": 6225
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的 ListObjectsV2。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(bucketName string) ([]types.Object, error)
 {
```

```
 result, err := basics.S3Client.ListObjectsV2(context.TODO(),
 &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 })
 var contents []types.Object
 if err != nil {
  log.Printf("Couldn't list objects in bucket %v. Here's why: %v\n", bucketName,
 err)
 } else {
  contents = result.Contents
 }
 return contents, err
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK for Go API 參考中的 ListObjectsV2。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class ListObjects {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The Amazon S3 bucket from which objects are
 read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                    .builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
                System.out.print("\n The object is " + calKb(myValue.size()) + "
 KBs");
```

```
                System.out.print("\n The owner is " + myValue.owner());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // convert bytes to kbs.
    private static long calKb(Long val) {
        return val / 1024;
    }
}
```

使用分頁列出物件。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The Amazon S3 bucket from which objects are
 read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
```

```
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                    .bucket(bucketName)
                    .maxKeys(1)
                    .build();

            ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
            listRes.stream()
                    .flatMap(r -> r.contents().stream())
                    .forEach(content -> System.out.println(" Key: " +
 content.key() + " size = " + content.size()));

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK for Java 2.x API 參考中的 ListObjectsV2。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

條列儲存貯體中的所有物件。如果有多個對象， IsTruncated 並且 NextContinuationToken 將被用於遍歷完整列表。

```
import {
  S3Client,
  // This command supersedes the ListObjectsCommand and is the recommended way to
 list objects.
  ListObjectsV2Command,
} from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new ListObjectsV2Command({
    Bucket: "my-bucket",
    // The default and maximum number of keys returned is 1000. This limits it to
    // one for demonstration purposes.
    MaxKeys: 1,
  });

  try {
    let isTruncated = true;

    console.log("Your bucket contains the following objects:\n");
    let contents = "";

    while (isTruncated) {
      const { Contents, IsTruncated, NextContinuationToken } =
        await client.send(command);
      const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
      contents += contentsList + "\n";
      isTruncated = IsTruncated;
      command.input.ContinuationToken = NextContinuationToken;
    }
    console.log(contents);
  } catch (err) {
    console.error(err);
  }
};
```

- 有關 API 的詳細信息，請參閱 AWS SDK for JavaScript API 參考中的 ListObjectsV2。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的 ListObjectsV2 以獲取 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

列出儲存貯體中的物件。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }
```

- 有關 API 的詳細信息，請參閱 AWS SDK for PHP API 參考中的 ListObjectsV2。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
```

```
        self.key = self.object.key


    @staticmethod
    def list(bucket, prefix=None):
        """
        Lists the objects in a bucket, optionally filtered by a prefix.

        :param bucket: The bucket to query. This is a Boto3 Bucket resource.
        :param prefix: When specified, only objects that start with this prefix
are listed.
        :return: The list of objects.
        """
        try:
            if not prefix:
                objects = list(bucket.objects.all())
            else:
                objects = list(bucket.objects.filter(Prefix=prefix))
            logger.info(
                "Got objects %s from bucket '%s'", [o.key for o in objects],
bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get objects for bucket '%s'.",
bucket.name)
            raise
        else:
            return objects
```

- 如需 API 的詳細資訊，請參 AWS SDK for Python (Boto3) 發套件中的 ListObjectsV2。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why:
 #{e.message}"
    0
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 有關 API 的詳細信息，請參閱 AWS SDK for Ruby API 參考中的 ListObjectsV2。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
pub async fn list_objects(client: &Client, bucket: &str) -> Result<(), Error> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的 ListObjectsV2 以取得 Rust API 參考資料。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   TRY.
       oo_result = lo_s3->listobjectsv2(         " oo_result is returned for
testing purposes. "
         iv_bucket = iv_bucket_name
       ).
       MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
   ENDTRY.
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的 ListObjectsV2，瞭解 SAP ABAP API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public func listBucketFiles(bucket: String) async throws -> [String] {
        let input = ListObjectsV2Input(
            bucket: bucket
        )
        let output = try await client.listObjectsV2(input: input)
        var names: [String] = []

        guard let objList = output.contents else {
            return []
        }

        for obj in objList {
            if let objName = obj.key {
                names.append(objName)
            }
        }

        return names
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的 ListObjectsV2 以獲取 Swift API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutBucketAccelerateConfiguration配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketAccelerateConfiguration。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    using System;
```

```
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
    /// bucket-level feature that enables you to perform faster data transfers
    /// to Amazon S3. This example shows how to configure Transfer
    /// Acceleration.
    /// </summary>
    public class TransferAcceleration
    {
        /// <summary>
        /// The main method initializes the client object and sets the
        /// Amazon Simple Storage Service (Amazon S3) bucket name before
        /// calling EnableAccelerationAsync.
        /// </summary>
        public static async Task Main()
        {
            var s3Client = new AmazonS3Client();
            const string bucketName = "doc-example-bucket";

            await EnableAccelerationAsync(s3Client, bucketName);
        }

        /// <summary>
        /// This method sets the configuration to enable transfer acceleration
        /// for the bucket referred to in the bucketName parameter.
        /// </summary>
        /// <param name="client">An Amazon S3 client used to enable the
        /// acceleration on an Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
the
        /// method will be enabling acceleration.</param>
        private static async Task EnableAccelerationAsync(AmazonS3Client client,
string bucketName)
        {
            try
            {
                var putRequest = new PutBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                    AccelerateConfiguration = new AccelerateConfiguration
                    {
```

```
                        Status = BucketAccelerateStatus.Enabled,
                    },
                };
                await client.PutBucketAccelerateConfigurationAsync(putRequest);

                var getRequest = new GetBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                };
                var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

                Console.WriteLine($"Acceleration state = '{response.Status}' ");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error occurred. Message:'{ex.Message}' when
setting transfer acceleration");
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考[PutBucketAccelerateConfiguration](#)中的。

CLI

AWS CLI

若要設定值區的加速組態

下列`put-bucket-accelerate-configuration`範例會啟用指定值區的加速組態。

```
aws s3api put-bucket-accelerate-configuration \
    --bucket my-bucket \
    --accelerate-configuration Status=Enabled
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[PutBucketAccelerateConfiguration](#)中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令啟用指定 S3 儲存貯體的傳輸加速。

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 's3testbucket' -
AccelerateConfiguration_Status $statusVal
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutBucketAccelerateConfiguration式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutBucketAcl配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketAcl。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 管理存取控制清單 (ACL)

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        /// <summary>
        /// Creates an Amazon S3 bucket with an ACL to control access to the
        /// bucket and the objects stored in it.
```

```
        /// </summary>
        /// <param name="client">The initialized client object used to create
        /// an Amazon S3 bucket, with an ACL applied to the bucket.
        /// </param>
        /// <param name="region">The AWS Region where the bucket will be
created.</param>
        /// <param name="newBucketName">The name of the bucket to create.</param>
        /// <returns>A boolean value indicating success or failure.</returns>
        public static async Task<bool> CreateBucketUseCannedACLAsync(IAmazonS3
client, S3Region region, string newBucketName)
        {
            try
            {
                // Create a new Amazon S3 bucket with Canned ACL.
                var putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = region,
                    CannedACL = S3CannedACL.LogDeliveryWrite,
                };

                PutBucketResponse putBucketResponse = await
client.PutBucketAsync(putBucketRequest);

                return putBucketResponse.HttpStatusCode ==
System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Amazon S3 error: {ex.Message}");
            }

            return false;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考PutBucketAcl中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
 &ownerID,
                              const Aws::String &granteePermission,
                              const Aws::String &granteeType, const Aws::String
 &granteeID,
                              const Aws::String &granteeEmailAddress,
                              const Aws::String &granteeURI, const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));
```

```
    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
            s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
                  << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param access: Human readable string.
 \return Permission: A Permission enum.
*/

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
```

```
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration
*/

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考PutBucketAcl中的。

CLI

AWS CLI

此範例授`full control`予兩位 AWS 使用者 (user1@example.com 和 user2@example.com)，並授予所有人的`read`權限：

```
aws s3api put-bucket-acl --bucket MyBucket --grant-full-control
 emailaddress=user1@example.com,emailaddress=user2@example.com --grant-read
 uri=http://acs.amazonaws.com/groups/global/AllUsers
```

如需有關自訂 ACL (s3api ACL 命令，例如，使用相同的速記引數表示法) 的詳細資訊`put-bucket-acl`，請參閱 http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutBucketAcl中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <bucketName> <id>\s

                Where:
                  bucketName - The Amazon S3 bucket to grant permissions on.\s
                  id - The ID of the owner of this bucket (you can get this value
 from the AWS Management Console).
```

```
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }

    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                    .grantee(builder -> builder.id(id)
                            .type(Type.CANONICAL_USER))
                    .permission(Permission.FULL_CONTROL)
                    .build();

            List<Grant> grantList2 = new ArrayList<>();
            grantList2.add(ownerGrant);

            AccessControlPolicy acl = AccessControlPolicy.builder()
                    .owner(builder -> builder.id(id))
                    .grants(grantList2)
                    .build();

            PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
                    .bucket(bucketName)
                    .accessControlPolicy(acl)
                    .build();

            s3.putBucketAcl(putAclReq);
```

```
        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考PutBucketAcl中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

放置儲存貯體的 ACL。

```
import { PutBucketAclCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Most Amazon S3 use cases don't require the use of access control lists (ACLs).
// We recommend that you disable ACLs, except in unusual circumstances where
// you need to control access for each object individually.
// Consider a policy instead. For more information see https://
docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-policies.html.
export const main = async () => {
  // Grant a user READ access to a bucket.
  const command = new PutBucketAclCommand({
    Bucket: "test-bucket",
    AccessControlPolicy: {
      Grants: [
        {
          Grantee: {
            // The canonical ID of the user. This ID is an obfuscated form of
 your AWS account number.
            // It's unique to Amazon S3 and can't be found elsewhere.
```

```
            // For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/finding-canonical-user-id.html.
            ID: "canonical-id-1",
            Type: "CanonicalUser",
          },
          // One of FULL_CONTROL | READ | WRITE | READ_ACP | WRITE_ACP
          // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_Grant.html#AmazonS3-Type-Grant-Permission
          Permission: "FULL_CONTROL",
        },
      ],
      Owner: {
        ID: "canonical-id-2",
      },
    },
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutBucketAcl中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun setBucketAcl(
    bucketName: String,
```

```
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK PutBucketAcl中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def grant_log_delivery_access(self):
        """
        Grant the AWS Log Delivery group write access to the bucket so that
        Amazon S3 can deliver access logs to the bucket. This is the only
 recommended
        use of an S3 bucket ACL.
        """
        try:
            acl = self.bucket.Acl()
            # Putting an ACL overwrites the existing ACL. If you want to preserve
            # existing grants, append new grants to the list of existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {
                        "Type": "Group",
                        "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery",
                    },
                    "Permission": "WRITE",
```

```
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted log delivery access to bucket '%s'",
self.bucket.name)
        except ClientError:
            logger.exception("Couldn't add ACL to bucket '%s'.",
self.bucket.name)
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件PutBucketAcl中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutBucketCors配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketCors。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        /// <summary>
        /// Add CORS configuration to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to apply the CORS configuration to an Amazon S3 bucket.</param>
        /// <param name="configuration">The CORS configuration to apply.</param>
        private static async Task PutCORSConfigurationAsync(AmazonS3Client
client, CORSConfiguration configuration)
        {
```

```
            PutCORSConfigurationRequest request = new
 PutCORSConfigurationRequest()
            {
                BucketName = BucketName,
                Configuration = configuration,
            };

            _ = await client.PutCORSConfigurationAsync(request);
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考PutBucketCors中的。

CLI

AWS CLI

下列範例會啟用PUTPOST、和來自 www.example.com 的DELETE要求，並啟用來自任何網域的GET要求：

```
aws s3api put-bucket-cors --bucket MyBucket --cors-configuration file://cors.json

cors.json:
{
  "CORSRules": [
    {
      "AllowedOrigins": ["http://www.example.com"],
      "AllowedHeaders": ["*"],
      "AllowedMethods": ["PUT", "POST", "DELETE"],
      "MaxAgeSeconds": 3000,
      "ExposeHeaders": ["x-amz-server-side-encryption"]
    },
    {
      "AllowedOrigins": ["*"],
      "AllowedHeaders": ["Authorization"],
      "AllowedMethods": ["GET"],
      "MaxAgeSeconds": 3000
    }
  ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[PutBucketCors](#)中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <accountId>\s

                Where:
                    bucketName - The Amazon S3 bucket to upload an object into.
```

```
                    accountId - The id of the account that owns the Amazon S3
bucket.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    public static void deleteBucketCorsInformation(S3Client s3, String
bucketName, String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
                    .bucket(bucketName)
                    .expectedBucketOwner(accountId)
                    .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            GetBucketCorsRequest bucketCorsRequest =
GetBucketCorsRequest.builder()
```

```
                    .bucket(bucketName)
                    .expectedBucketOwner(accountId)
                    .build();

            GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.corsRules();
            for (CORSRule rule : corsRules) {
                System.out.println("allowOrigins: " + rule.allowedOrigins());
                System.out.println("AllowedMethod: " + rule.allowedMethods());
            }

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
        List<String> allowMethods = new ArrayList<>();
        allowMethods.add("PUT");
        allowMethods.add("POST");
        allowMethods.add("DELETE");

        List<String> allowOrigins = new ArrayList<>();
        allowOrigins.add("http://example.com");
        try {
            // Define CORS rules.
            CORSRule corsRule = CORSRule.builder()
                    .allowedMethods(allowMethods)
                    .allowedOrigins(allowOrigins)
                    .build();

            List<CORSRule> corsRules = new ArrayList<>();
            corsRules.add(corsRule);
            CORSConfiguration configuration = CORSConfiguration.builder()
                    .corsRules(corsRules)
                    .build();

            PutBucketCorsRequest putBucketCorsRequest =
PutBucketCorsRequest.builder()
                    .bucket(bucketName)
```

```
                    .corsConfiguration(configuration)
                    .expectedBucketOwner(accountId)
                    .build();

            s3.putBucketCors(putBucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考PutBucketCors中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

新增 CORS 規則。

```
import { PutBucketCorsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// By default, Amazon S3 doesn't allow cross-origin requests. Use this command
// to explicitly allow cross-origin requests.
export const main = async () => {
  const command = new PutBucketCorsCommand({
    Bucket: "test-bucket",
    CORSConfiguration: {
      CORSRules: [
        {
          // Allow all headers to be sent to this bucket.
          AllowedHeaders: ["*"],
```

```
          // Allow only GET and PUT methods to be sent to this bucket.
          AllowedMethods: ["GET", "PUT"],
          // Allow only requests from the specified origin.
          AllowedOrigins: ["https://www.example.com"],
          // Allow the entity tag (ETag) header to be returned in the response.
 The ETag header
          // The entity tag represents a specific version of the object. The ETag
 reflects
          // changes only to the contents of an object, not its metadata.
          ExposeHeaders: ["ETag"],
          // How long the requesting browser should cache the preflight response.
 After
          // this time, the preflight request will have to be made again.
          MaxAgeSeconds: 3600,
        },
      ],
    },
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutBucketCors中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_cors(self, cors_rules):
        """
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that
 are
        allowed from other domains.

        :param cors_rules: The CORS rules to apply.
        """
        try:
            self.bucket.Cors().put(CORSConfiguration={"CORSRules": cors_rules})
            logger.info(
                "Put CORS rules %s for bucket '%s'.", cors_rules,
 self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't put CORS rules for bucket %s.",
 self.bucket.name)
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件PutBucketCors中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
```

```
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考PutBucketCors中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutBucketEncryption配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketEncryption。

CLI

AWS CLI

設定值區的伺服器端加密

下列put-bucket-encryption範例會將 AES256 加密設定為指定值區的預設值。

```
aws s3api put-bucket-encryption \
    --bucket my-bucket \
    --server-side-encryption-configuration '{"Rules":
 [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutBucketEncryption中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令使用指定儲存貯體上的 Amazon S3 受管金鑰 (SSE-S3) 啟用預設 AES256 伺服器端加密。

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 's3testbucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutBucketEncryption式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutBucketLifecycleConfiguration配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketLifecycleConfiguration。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 刪除不完整的分段上傳
- 使用版本化物件

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        /// <summary>
        /// Adds lifecycle configuration information to the S3 bucket named in
        /// the bucketName parameter.
        /// </summary>
        /// <param name="client">The S3 client used to call the
        /// PutLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">A string representing the S3 bucket to
```

```
        /// which configuration information will be added.</param>
        /// <param name="configuration">A LifecycleConfiguration object that
        /// will be applied to the S3 bucket.</param>
        public static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
 string bucketName, LifecycleConfiguration configuration)
        {
            var request = new PutLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
                Configuration = configuration,
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考[PutBucketLifecycleConfiguration](#)中的。

CLI

AWS CLI

下列指令會將生命週期組態套用至名為的值區`my-bucket`：

```
aws s3api put-bucket-lifecycle-configuration --bucket my-bucket --lifecycle-
configuration  file://lifecycle.json
```

該文件`lifecycle.json`是指定兩個規則的當前文件夾中的 JSON 文檔：

```
{
    "Rules": [
        {
            "ID": "Move rotated logs to Glacier",
            "Prefix": "rotated/",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Date": "2015-11-10T00:00:00.000Z",
                    "StorageClass": "GLACIER"
                }
            ]
```

```
        },
        {
            "Status": "Enabled",
            "Prefix": "",
            "NoncurrentVersionTransitions": [
                {
                    "NoncurrentDays": 2,
                    "StorageClass": "GLACIER"
                }
            ],
            "ID": "Move old versions to Glacier"
        }
    ]
}
```

第一個規則會在指定日期將帶有前置詞的檔案移`rotated`至 Glacier。第二個規則會在舊物件版本不再是最新的時候，將它們移至 Glacier。如需有關可接受的時間戳記格式的資訊，請參閱 AWS CLI 使用者指南中的指定參數值。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[PutBucketLifecycleConfiguration](#)中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
```

```
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class LifecycleConfiguration {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                  <bucketName> <accountId>\s

                                Where:
                                  bucketName - The Amazon Simple Storage Service
 (Amazon S3) bucket to upload an object into.
                                  accountId - The id of the account that owns the
 Amazon S3 bucket.
                                """;

                if (args.length != 2) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String bucketName = args[0];
                String accountId = args[1];
                Region region = Region.US_EAST_1;
                S3Client s3 = S3Client.builder()
                                .region(region)
                                .build();
```

```
                setLifecycleConfig(s3, bucketName, accountId);
                getLifecycleConfig(s3, bucketName, accountId);
                deleteLifecycleConfig(s3, bucketName, accountId);
                System.out.println("You have successfully created, updated, and
deleted a Lifecycle configuration");
                s3.close();
        }

        public static void setLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
                try {
                        // Create a rule to archive objects with the
"glacierobjects/" prefix to Amazon
                        // S3 Glacier.
                        LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                                        .prefix("glacierobjects/")
                                        .build();

                        Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                                        .days(0)
                                        .build();

                        LifecycleRule rule1 = LifecycleRule.builder()
                                        .id("Archive immediately rule")
                                        .filter(ruleFilter)
                                        .transitions(transition)
                                        .status(ExpirationStatus.ENABLED)
                                        .build();

                        // Create a second rule.
                        Transition transition2 = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                                        .days(0)
                                        .build();

                        List<Transition> transitionList = new ArrayList<>();
                        transitionList.add(transition2);

                        LifecycleRuleFilter ruleFilter2 =
LifecycleRuleFilter.builder()
```

```
                                        .prefix("glacierobjects/")
                                        .build();

                        LifecycleRule rule2 = LifecycleRule.builder()
                                        .id("Archive and then delete rule")
                                        .filter(ruleFilter2)
                                        .transitions(transitionList)
                                        .status(ExpirationStatus.ENABLED)
                                        .build();

                        // Add the LifecycleRule objects to an ArrayList.
                        ArrayList<LifecycleRule> ruleList = new ArrayList<>();
                        ruleList.add(rule1);
                        ruleList.add(rule2);

                        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                                        .rules(ruleList)
                                        .build();

                        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                                        .builder()
                                        .bucket(bucketName)

.lifecycleConfiguration(lifecycleConfiguration)
                                        .expectedBucketOwner(accountId)
                                        .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }

        // Retrieve the configuration and add a new rule.
        public static void getLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
                try {
                        GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
```

```
                                        .builder()
                                        .bucket(bucketName)
                                        .expectedBucketOwner(accountId)
                                        .build();

                        GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
                        List<LifecycleRule> newList = new ArrayList<>();
                        List<LifecycleRule> rules = response.rules();
                        for (LifecycleRule rule : rules) {
                                newList.add(rule);
                        }

                        // Add a new rule with both a prefix predicate and a tag
predicate.
                        LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                                        .prefix("YearlyDocuments/")
                                        .build();

                        Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                                        .days(3650)
                                        .build();

                        LifecycleRule rule1 = LifecycleRule.builder()
                                        .id("NewRule")
                                        .filter(ruleFilter)
                                        .transitions(transition)
                                        .status(ExpirationStatus.ENABLED)
                                        .build();

                        // Add the new rule to the list.
                        newList.add(rule1);
                        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                                        .rules(newList)
                                        .build();

                        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                                        .builder()
```

```
                                        .bucket(bucketName)

    .lifecycleConfiguration(lifecycleConfiguration)
                                        .expectedBucketOwner(accountId)
                                        .build();


    s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }

        // Delete the configuration from the Amazon S3 bucket.
        public static void deleteLifecycleConfig(S3Client s3, String bucketName,
    String accountId) {
                try {
                        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest
    = DeleteBucketLifecycleRequest
                                        .builder()
                                        .bucket(bucketName)
                                        .expectedBucketOwner(accountId)
                                        .build();

                        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[PutBucketLifecycleConfiguration](#)中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_lifecycle_configuration(self, lifecycle_rules):
        """
        Apply a lifecycle configuration to the bucket. The lifecycle
 configuration can
        be used to archive or delete the objects in the bucket according to
 specified
        parameters, such as a number of days.

        :param lifecycle_rules: The lifecycle rules to apply.
        """
        try:
            self.bucket.LifecycleConfiguration().put(
                LifecycleConfiguration={"Rules": lifecycle_rules}
            )
            logger.info(
                "Put lifecycle rules %s for bucket '%s'.",
                lifecycle_rules,
                self.bucket.name,
            )
```

```
        except ClientError:
            logger.exception(
                "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件PutBucketLifecycleConfiguration中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutBucketLogging配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketLogging。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
/// logs.
```

```
    /// </summary>
    public class ServerAccessLogging
    {
        private static IConfiguration _configuration = null!;

        public static async Task Main()
        {
            LoadConfig();

            string bucketName = _configuration["BucketName"];
            string logBucketName = _configuration["LogBucketName"];
            string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
            string accountId = _configuration["AccountId"];

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Update bucket policy for target bucket to allow delivery of
logs to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);

                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }
```

```
        /// <summary>
        /// This method grants appropriate permissions for logging to the
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
        /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" +
logPrefix + @"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
                                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
                                ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3:::" + sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
accountId + @""" }
                                        }
                                    }]
                                }";
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
            Console.WriteLine(newPolicy);
```

```
            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
        /// This method enables logging for an Amazon S3 bucket. Logs will be
 stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
 used
        /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
 you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
 logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
```

```
                BucketName = bucketName,
                LoggingConfig = loggingConfig,
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
                .AddJsonFile("settings.local.json", true) // Optionally, load
 local settings.
                .Build();
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考[PutBucketLogging](#)中的。

CLI

AWS CLI

範例 1：設定值區政策記錄

下列put-bucket-logging範例會設定的記錄原則MyBucket。首先，使用put-bucket-policy命令授予值區政策中的記錄服務主體權限。

```
aws s3api put-bucket-policy \
    --bucket MyBucket \
    --policy file://policy.json
```

policy.json 的內容：

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {"Service": "logging.s3.amazonaws.com"},
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::MyBucket/Logs/*",
            "Condition": {
                "ArnLike": {"aws:SourceARN": "arn:aws:s3:::SOURCE-BUCKET-NAME"},
                "StringEquals": {"aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"}
            }
        }
    ]
}
```

若要套用記錄原則，請使用put-bucket-logging。

```
aws s3api put-bucket-logging \
    --bucket MyBucket \
    --bucket-logging-status file://logging.json
```

logging.json 的內容：

```
{
     "LoggingEnabled": {
         "TargetBucket": "MyBucket",
         "TargetPrefix": "Logs/"
     }
 }
```

需要此put-bucket-policy命令才能將s3:PutObject權限授與記錄服務主體。

如需詳細資訊，請參閱 Amazon S3 使用者指南中的 Amazon S3 伺服器存取日誌。

範例 2：若要設定值區政策，以便只記錄單一使用者的存取權

下列put-bucket-logging範例會設定的記錄原則MyBucket。 AWS 使用者 bob@example.com 將擁有對記錄檔的完全控制權，而且沒有其他人擁有任何存取權。首先，授與 S3 許可put-bucket-acl。

```
aws s3api put-bucket-acl \
```

```
    --bucket MyBucket \
    --grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery \
    --grant-read-acp URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

然後使用套用記錄原則put-bucket-logging。

```
aws s3api put-bucket-logging \
    --bucket MyBucket \
    --bucket-logging-status file://logging.json
```

logging.json 的內容：

```
{
    "LoggingEnabled": {
        "TargetBucket": "MyBucket",
        "TargetPrefix": "MyBucketLogs/",
        "TargetGrants": [
            {
                "Grantee": {
                    "Type": "AmazonCustomerByEmail",
                    "EmailAddress": "bob@example.com"
                },
                "Permission": "FULL_CONTROL"
            }
        ]
    }
}
```

需要該put-bucket-acl命令才能授予 S3 的日誌傳遞系統必要的權限（寫入和讀取 ACP 權限）。

如需詳細資訊，請參閱 Amazon S3 開發人員指南中的 Amazon S3 伺服器存取日誌。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutBucketLogging中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutBucketNotification配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketNotification。

CLI

AWS CLI

會將通知組態套用至名為my-bucket：

```
aws s3api put-bucket-notification --bucket my-bucket --notification-configuration
 file://notification.json
```

該檔案notification.json是指定要監視的 SNS 主題和事件類型的目前資料夾中的 JSON 文件：

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

SNS 主題必須附加 IAM 政策，以便讓 Amazon S3 發佈到該主題：

```
{
 "Version": "2008-10-17",
 "Id": "example-ID",
 "Statement": [
  {
   "Sid": "example-statement-ID",
   "Effect": "Allow",
   "Principal": {
     "Service": "s3.amazonaws.com"
   },
   "Action": [
    "SNS:Publish"
   ],
   "Resource": "arn:aws:sns:us-west-2:123456789012:my-bucket",
   "Condition": {
      "ArnLike": {
      "aws:SourceArn": "arn:aws:s3:*:*:my-bucket"
    }
   }
  }
 ]
```

```
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutBucketNotification中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例設定 S3 事件的 SNS 主題組態， ObjectRemovedDelete 並啟用指定 s3 儲存貯體的通知

```
$topic =  [Amazon.S3.Model.TopicConfiguration] @{
  Id = "delete-event"
  Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
  Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName kt-tools -TopicConfiguration $topic
```

範例 2：此範例會啟 ObjectCreatedAll 用將其傳送至 Lambda 函數的指定儲存貯體的通知。

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
  Id = "ObjectCreated-Lambda"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".pem"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName ssm-editor -LambdaFunctionConfiguration
 $lambdaConfig
```

範例 3：此範例會根據不同的索引鍵尾碼建立 2 個不同的 Lambda 組態，並在單一命令中進行設定。

```
#Lambda Config 1
```

```
$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
  Id = "ObjectCreated-dada-ps1"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".ps1"}
      )
    }
  }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName ssm-editor -LambdaFunctionConfiguration
 $firstLambdaConfig,$secondlambdaConfig
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutBucketNotification式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutBucketNotificationConfiguration配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketNotificationConfiguration。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to enable notifications for an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
    public class EnableNotifications
    {
        public static async Task Main()
        {
            const string bucketName = "doc-example-bucket1";
            const string snsTopic = "arn:aws:sns:us-east-2:0123456789ab:bucket-
notify";
            const string sqsQueue = "arn:aws:sqs:us-
east-2:0123456789ab:Example_Queue";

            IAmazonS3 client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2);
            await EnableNotificationAsync(client, bucketName, snsTopic,
 sqsQueue);
        }

        /// <summary>
        /// This method makes the call to the PutBucketNotificationAsync method.
```

```
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to call
        /// the PutBucketNotificationAsync method.</param>
        /// <param name="bucketName">The name of the bucket for which
        /// notifications will be turned on.</param>
        /// <param name="snsTopic">The ARN for the Amazon Simple Notification
        /// Service (Amazon SNS) topic associated with the S3 bucket.</param>
        /// <param name="sqsQueue">The ARN of the Amazon Simple Queue Service
        /// (Amazon SQS) queue to which notifications will be pushed.</param>
        public static async Task EnableNotificationAsync(
            IAmazonS3 client,
            string bucketName,
            string snsTopic,
            string sqsQueue)
        {
            try
            {
                // The bucket for which we are setting up notifications.
                var request = new PutBucketNotificationRequest()
                {
                    BucketName = bucketName,
                };

                // Defines the topic to use when sending a notification.
                var topicConfig = new TopicConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic,
                };
                request.TopicConfigurations = new List<TopicConfiguration>
                {
                    topicConfig,
                };
                request.QueueConfigurations = new List<QueueConfiguration>
                {
                    new QueueConfiguration()
                    {
                        Events = new List<EventType>
{ EventType.ObjectCreatedPut },
                        Queue = sqsQueue,
                    },
                };

                // Now apply the notification settings to the bucket.
```

```
                PutBucketNotificationResponse response = await
 client.PutBucketNotificationAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考[PutBucketNotificationConfiguration](中的。

CLI

AWS CLI

啟用值區的指定通知

下列put-bucket-notification-configuration範例會將通知組態套用至名為的值區my-bucket。該檔案notification.json是目前資料夾中的 JSON 文件，用於指定要監視的 SNS 主題和事件類型。

```
aws s3api put-bucket-notification-configuration \
    --bucket my-bucket \
    --notification-configuration file://notification.json
```

notification.json 的內容：

```
{
    "TopicConfigurations": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-
topic",
            "Events": [
                "s3:ObjectCreated:*"
            ]
        }
    ]
```

```
}
```

SNS 主題必須附加 IAM 政策，以允許 Amazon S3 向其發佈。

```
{
    "Version": "2008-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "arn:aws:sns:us-west-2:123456789012::s3-notification-
topic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:my-bucket"
                }
            }
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[PutBucketNotificationConfiguration](#)中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Event;
import software.amazon.awssdk.services.s3.model.NotificationConfiguration;
import
 software.amazon.awssdk.services.s3.model.PutBucketNotificationConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.TopicConfiguration;
import java.util.ArrayList;
import java.util.List;

public class SetBucketEventBridgeNotification {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The Amazon S3 bucket.\s
                    topicArn - The Simple Notification Service topic ARN.\s
                    id - An id value used for the topic configuration. This value
 is displayed in the AWS Management Console.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String topicArn = args[1];
        String id = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        setBucketNotification(s3Client, bucketName, topicArn, id);
        s3Client.close();
    }

    public static void setBucketNotification(S3Client s3Client, String
 bucketName, String topicArn, String id) {
        try {
```

```
            List<Event> events = new ArrayList<>();
            events.add(Event.S3_OBJECT_CREATED_PUT);

            TopicConfiguration config = TopicConfiguration.builder()
                    .topicArn(topicArn)
                    .events(events)
                    .id(id)
                    .build();

            List<TopicConfiguration> topics = new ArrayList<>();
            topics.add(config);

            NotificationConfiguration configuration =
 NotificationConfiguration.builder()
                    .topicConfigurations(topics)
                    .build();

            PutBucketNotificationConfigurationRequest configurationRequest =
 PutBucketNotificationConfigurationRequest
                    .builder()
                    .bucket(bucketName)
                    .notificationConfiguration(configuration)
                    .skipDestinationValidation(true)
                    .build();

            // Set the bucket notification configuration.
            s3Client.putBucketNotificationConfiguration(configurationRequest);
            System.out.println("Added bucket " + bucketName + " with EventBridge
 events enabled.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[PutBucketNotificationConfiguration](#)中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutBucketPolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketPolicy。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                 const Aws::String &policyBody,
                                 const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
            Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
            s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                  bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }
```

```
    return outcome.IsSuccess();
}


//! Build a policy JSON string.
/*!
  \param userArn: Aws user Amazon Resource Name (ARN).
      For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
  \param bucketName: Name of a bucket.
  \return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                            const Aws::String &bucketName) {
    return
            "{\n"
            "   \"Version\":\"2012-10-17\",\n"
            "   \"Statement\":[\n"
            "       {\n"
            "           \"Sid\": \"1\",\n"
            "           \"Effect\": \"Allow\",\n"
            "           \"Principal\": {\n"
            "               \"AWS\": \""
            + userArn +
            "\"\n""           },\n"
            "           \"Action\": [ \"s3:getObject\" ],\n"
            "           \"Resource\": [ \"arn:aws:s3:::"
            + bucketName +
            "/*\" ]\n"
            "       }\n"
            "   ]\n"
            "}";
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考[PutBucketPolicy](#)中的。

CLI

AWS CLI

此範例允許所有使用者擷取中的物件，MyBucket但中的物件除外MySecretFolder。它還授予put和delete權限給帳戶的 root 用 AWS 戶1234-5678-9012：

```
aws s3api put-bucket-policy --bucket MyBucket --policy file://policy.json

policy.json:
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": "*",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::MyBucket/*"
      },
      {
         "Effect": "Deny",
         "Principal": "*",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::MyBucket/MySecretFolder/*"
      },
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:root"
         },
         "Action": [
            "s3:DeleteObject",
            "s3:PutObject"
         ],
         "Resource": "arn:aws:s3:::MyBucket/*"
      }
   ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutBucketPolicy中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <polFile>

                Where:
                    bucketName - The Amazon S3 bucket to set the policy on.
                    polFile - A JSON file containing the policy (see the Amazon
 S3 Readme for an example).\s
```

```
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }

    public static void setPolicy(S3Client s3, String bucketName, String
policyText) {
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();

            s3.putBucketPolicy(policyReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }

    // Loads a JSON-formatted policy from a file
```

```
    public static String getBucketPolicyFromFile(String policyFile) {

        StringBuilder fileText = new StringBuilder();
        try {
            List<String> lines = Files.readAllLines(Paths.get(policyFile),
 StandardCharsets.UTF_8);
            for (String line : lines) {
                fileText.append(line);
            }

        } catch (IOException e) {
            System.out.format("Problem reading file: \"%s\"", policyFile);
            System.out.println(e.getMessage());
        }

        try {
            final JsonParser parser = new
 ObjectMapper().getFactory().createParser(fileText.toString());
            while (parser.nextToken() != null) {
            }

        } catch (IOException jpe) {
            jpe.printStackTrace();
        }
        return fileText.toString();
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考PutBucketPolicy中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

新增政策。

```
import { PutBucketPolicyCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new PutBucketPolicyCommand({
    Policy: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Sid: "AllowGetObject",
          // Allow this particular user to call GetObject on any object in this
 bucket.
          Effect: "Allow",
          Principal: {
            AWS: "arn:aws:iam::ACCOUNT-ID:user/USERNAME",
          },
          Action: "s3:GetObject",
          Resource: "arn:aws:s3:::BUCKET-NAME/*",
        },
      ],
    }),
    // Apply the preceding policy to this bucket.
    Bucket: "BUCKET-NAME",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutBucketPolicy中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_policy(self, policy):
        """
        Apply a security policy to the bucket. Policies control users' ability
        to perform specific actions, such as listing the objects in the bucket.

        :param policy: The policy to apply to the bucket.
        """
        try:
            self.bucket.Policy().put(Policy=json.dumps(policy))
            logger.info("Put policy %s for bucket '%s'.", policy,
 self.bucket.name)
        except ClientError:
            logger.exception("Couldn't apply policy to bucket '%s'.",
 self.bucket.name)
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件PutBucketPolicy中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def set_policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考PutBucketPolicy中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutBucketReplication配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketReplication。

CLI

AWS CLI

設定 S3 儲存貯體的複寫

下列put-bucket-replication範例會將複寫組態套用至指定的 S3 儲存貯體。

```
aws s3api put-bucket-replication \
    --bucket AWSDOC-EXAMPLE-BUCKET1 \
    --replication-configuration file://replication.json
```

replication.json 的內容：

```
{
    "Role": "arn:aws:iam::123456789012:role/s3-replication-role",
    "Rules": [
        {
            "Status": "Enabled",
            "Priority": 1,
            "DeleteMarkerReplication": { "Status": "Disabled" },
            "Filter" : { "Prefix": ""},
            "Destination": {
                "Bucket": "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET2"
            }
        }
    ]
}
```

目的地值區必須啟用版本控制。指定的角色必須具有寫入目標儲存貯體的權限，並具有允許 Amazon S3 擔任該角色的信任關係。

角色權限原則範例：

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetReplicationConfiguration",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET1/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ReplicateObject",
                "s3:ReplicateDelete",
                "s3:ReplicateTags"
            ],
            "Resource": "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET2/*"
        }
    ]
}
```

信任關係政策範例：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": "s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

此命令不會產生輸出。

如需詳細資訊，請參閱 Amazon 簡易儲存服務主控台使用者指南中的主題標題。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutBucketReplication中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例設定具有單一規則的複製組態，以便複製到 'exampletargetbucket' 儲存貯體，在值區 'examplebuket' 中以索引鍵名稱前置詞 "TaxDocs" 建立的任何新物件。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

範例 2：此範例會設定具有多個規則的複製組態，以便複寫至 'exampletargetbucket' 儲存貯體，以索引鍵名稱前置詞 "" 或 "TaxDocs「建立的任何新物件。OtherDocs關鍵字首碼不得重疊。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
```

```
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

範例 3：此範例會更新指定值區上的複寫組態，以停用控制將金鑰名稱前置詞為 "TaxDocs" 的物件複製到值區 'exampletargetbuket' 的規則。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutBucketReplication式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutBucketRequestPayment配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketRequestPayment。

CLI

AWS CLI

範例 1：啟用值區的「請求者付款」設定

下列put-bucket-request-payment範例會針requester pays對指定的值區啟用。

```
aws s3api put-bucket-request-payment \
    --bucket my-bucket \
    --request-payment-configuration '{"Payer":"Requester"}'
```

此命令不會產生輸出。

範例 2：若要停用值區的「請求者付款」設定

下列put-bucket-request-payment範例會requester pays針對指定值區停用。

```
aws s3api put-bucket-request-payment \
    --bucket my-bucket \
    --request-payment-configuration '{"Payer":"BucketOwner"}'
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutBucketRequestPayment中的。

PowerShell

適用的工具 PowerShell

範例 1：針對名為「mybucket」的值區更新請求付款組態，以便向值區要求下載的人員收取下載費用。根據預設，值區擁有者會支付下載費用。若要將請求付款設回預設值，請針對 RequestPaymentConfiguration _Payment 參數使用 BucketOwner "。

```
Write-S3BucketRequestPayment -BucketName mybucket -
RequestPaymentConfiguration_Payer Requester
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutBucketRequestPayment式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutBucketTagging配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketTagging。

CLI

AWS CLI

下列指令會將標記組態套用至名為的值區my-bucket：

```
aws s3api put-bucket-tagging --bucket my-bucket --tagging file://tagging.json
```

該文件tagging.json是指定標籤的當前文件夾中的 JSON 文檔：

```
{
   "TagSet": [
     {
       "Key": "organization",
       "Value": "marketing"
     }
   ]
}
```

或者my-bucket直接從命令行應用標記配置：

```
aws s3api put-bucket-tagging --bucket my-bucket --tagging
 'TagSet=[{Key=organization,Value=marketing}]'
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutBucketTagging中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會將兩個標記套用至名為的值區**cloudtrail-test-2018**：含有 Stage 索引鍵且值為 Test 的標記，以及含有環境鍵和 Alpha 值的標籤。若要確認標籤是否已新增至值區，請執行**Get-S3BucketTagging -BucketName bucket_name**。結果應顯示您在第一個命令中套用至值區的標籤。請注意，**Write-S3BucketTagging**會覆寫值區上的整個現有標記集。若要新增或刪除個別標籤，請執行 Resource Groups 和標記 API 指令程式，以**Add-RGTResourceTag**及**Remove-RGTResourceTag**。或者，使用 AWS 管理主控台中的標籤編輯器來管理 S3 儲存貯體標籤。

```
Write-S3BucketTagging -BucketName cloudtrail-test-2018 -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

範例 2：此命令會**cloudtrail-test-2018**將名為的值區傳送至**Write-S3BucketTagging**指令程式。它適用於標籤階段：生產和部門：財務桶。請注意，**Write-S3BucketTagging**會覆寫值區上的整個現有標記集。

```
Get-S3Bucket -BucketName cloudtrail-test-2018 | Write-S3BucketTagging
 -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
 Value="Finance" } )
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutBucketTagging式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**PutBucketVersioning**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketVersioning。

CLI

AWS CLI

下列指令會在名為的值區啟用版本控制my-bucket：

```
aws s3api put-bucket-versioning --bucket my-bucket --versioning-configuration
 Status=Enabled
```

以下命令啟用版本控制，並使用 mfa 代碼

```
aws s3api put-bucket-versioning --bucket my-bucket --versioning-configuration
 Status=Enabled --mfa "SERIAL 123456"
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutBucketVersioning中的。

PowerShell

適用的工具 PowerShell

範例 1：命令會為指定的 S3 儲存貯體啟用版本控制。

```
Write-S3BucketVersioning -BucketName 's3testbucket' -VersioningConfig_Status
 Enabled
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutBucketVersioning式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutBucketWebsite配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutBucketWebsite。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
                // Put the website configuration.
                PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument,
                    },
                };
                PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考PutBucketWebsite中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::String &indexPage, const Aws::String
 &errorPage,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
   Aws::S3::S3Client client(clientConfig);

   Aws::S3::Model::IndexDocument indexDocument;
   indexDocument.SetSuffix(indexPage);

   Aws::S3::Model::ErrorDocument errorDocument;
   errorDocument.SetKey(errorPage);
```

```
    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
            client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考PutBucketWebsite中的。

CLI

AWS CLI

會將靜態網站設定套用至名為my-bucket：

```
aws s3api put-bucket-website --bucket my-bucket --website-configuration file://
website.json
```

該文件website.json是指定網站索引和錯誤頁面的當前文件夾中的 JSON 文檔：

```
{
    "IndexDocument": {
        "Suffix": "index.html"
    },
    "ErrorDocument": {
        "Key": "error.html"
```

```
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[PutBucketWebsite](#)中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucketName> [indexdoc]\s

                Where:
                   bucketName   - The Amazon S3 bucket to set the website
 configuration on.\s
                   indexdoc - The index document, ex. 'index.html'
```

```
                                If not specified, 'index.html' will be set.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }

    public static void setWebsiteConfig(S3Client s3, String bucketName, String
 indexDoc) {
        try {
            WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()

 .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
                    .build();

            PutBucketWebsiteRequest pubWebsiteReq =
 PutBucketWebsiteRequest.builder()
                    .bucket(bucketName)
                    .websiteConfiguration(websiteConfig)
                    .build();

            s3.putBucketWebsite(pubWebsiteReq);
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考PutBucketWebsite中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

設定儲存貯體網站組態。

```
import { PutBucketWebsiteCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Set up a bucket as a static website.
// The bucket needs to be publicly accessible.
export const main = async () => {
  const command = new PutBucketWebsiteCommand({
    Bucket: "test-bucket",
    WebsiteConfiguration: {
      ErrorDocument: {
        // The object key name to use when a 4XX class error occurs.
        Key: "error.html",
      },
      IndexDocument: {
        // A suffix that is appended to a request that is for a directory.
        Suffix: "index.html",
      },
    },
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutBucketWebsite中的。

PowerShell

適用的工具 PowerShell

示例 1：該命令啟用給定存儲桶的網站託管，索引文檔為「index.html」，錯誤文檔為「錯誤 .html」。

```
Write-S3BucketWebsite -BucketName 's3testbucket' -
WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutBucketWebsite式參考中的。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
 configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end
```

```
  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the
 website.
  # @param error_document [String] The name of the error document to show for 4XX
 errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
 false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
 why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考PutBucketWebsite中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutObject配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutObject。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用儲存貯體和物件
- 追蹤上傳和下載

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
/// bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The Amazon S3 bucket to which the object
/// will be uploaded.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object
/// on the local computer to upload.</param>
/// <returns>A boolean value indicating the success or failure of the
/// upload procedure.</returns>
public static async Task<bool> UploadFileAsync(
    IAmazonS3 client,
    string bucketName,
    string objectName,
    string filePath)
{
    var request = new PutObjectRequest
    {
```

```
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await client.PutObjectAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully uploaded {objectName} to
{bucketName}.");
                return true;
            }
            else
            {
                Console.WriteLine($"Could not upload {objectName} to
{bucketName}.");
                return false;
            }
        }
```

使用伺服器端加密上傳物件。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket with server-side encryption enabled.
    /// </summary>
    public class ServerSideEncryption
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "samplefile.txt";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
```

```
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            await WritingAnObjectAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// Upload a sample object include a setting for encryption.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// to upload a file and apply server-side encryption.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// encrypted object will reside.</param>
        /// <param name="keyName">The name for the object that you want to
        /// create in the supplied bucket.</param>
        public static async Task WritingAnObjectAsync(IAmazonS3 client, string
bucketName, string keyName)
        {
            try
            {
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    ContentBody = "sample text",
                    ServerSideEncryptionMethod =
ServerSideEncryptionMethod.AES256,
                };

                var putResponse = await client.PutObjectAsync(putRequest);

                // Determine the encryption state of an object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
                ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;
```

```
                Console.WriteLine($"Encryption method used: {0}",
objectEncryption.ToString());
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}' when writing an
object");
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考PutObject中的。

Bash

AWS CLI 與 Bash 腳本

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
```

```
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3

  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutObject中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);
```

```
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval
 needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          fileName.c_str(),
                                          std::ios_base::in |
 std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << fileName << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3Client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Added object '" << fileName << "' to bucket '"
                  << bucketName << "'.";
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考PutObject中的。

CLI

AWS CLI

下列範例使用put-object命令將物件上傳到 Amazon S3：

```
aws s3api put-object --bucket text-content --key dir-1/my_images.tar.bz2 --body
 my_images.tar.bz2
```

下列範例顯示視訊檔案的上傳 (視訊檔案是使用 Windows 檔案系統語法指定的。):

```
aws s3api put-object --bucket text-content --key dir-1/big-video-file.mp4 --body
 e:\media\videos\f-sharp-3-data-services.mp4
```

如需有關上傳物件的詳細資訊，請參閱 Amazon S3 開發人員指南中的上傳物件。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutObject中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用低級 API 將對象放入存儲桶中。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(bucketName string, objectKey string,
 fileName string) error {
 file, err := os.Open(fileName)
```

```
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(context.TODO(), &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
    fileName, bucketName, objectKey, err)
  }
 }
 return err
}
```

使用傳輸管理員將物件上傳至值區。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
 string, contents string) (string, error) {
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
```

```
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
  }, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
 bucket)
  } else {
   outKey = *output.Key
  }
 }
 return outKey, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考PutObject中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用 S3Client 將文件上傳到儲存貯體。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
```

```
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class PutObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <bucketName> <objectKey> <objectPath>\s

                Where:
                  bucketName - The Amazon S3 bucket to upload an object into.
                  objectKey - The object to upload (for example, book.pdf).
                  objectPath - The path where the file is located (for example,
 C:/AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }
```

```
    // This example uses RequestBody.fromFile to avoid loading the whole file
 into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String
 objectKey, String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("x-amz-meta-myVal", "test");
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .metadata(metadata)
                    .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket
 " + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

使用 S3 TransferManager 將檔案上傳到儲存貯體。檢視完整檔案並測試。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String
 bucketName,
```

```
                                String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

使用 S3Client 將物件上傳至儲存貯體並設定索引標籤。

```
    public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
        try {
            Tag tag1 = Tag.builder()
                    .key("Tag 1")
                    .value("This is tag 1")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Tag 2")
                    .value("This is tag 2")
                    .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag1);
            tags.add(tag2);

            Tagging allTags = Tagging.builder()
                    .tagSet(tags)
                    .build();

            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .tagging(allTags)
                    .build();
```

```
            s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
        try {
            GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
            List<Tag> obTags = getTaggingRes.tagSet();
            for (Tag sinTag : obTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

            // Replace the object's tags with two new tags.
            Tag tag3 = Tag.builder()
                    .key("Tag 3")
                    .value("This is tag 3")
                    .build();

            Tag tag4 = Tag.builder()
                    .key("Tag 4")
                    .value("This is tag 4")
                    .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag3);
            tags.add(tag4);

            Tagging updatedTags = Tagging.builder()
                    .tagSet(tags)
                    .build();
```

```
            PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .tagging(updatedTags)
                    .build();

            s3.putObjectTagging(taggingRequest1);
            GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
            List<Tag> modTags = getTaggingRes2.tagSet();
            for (Tag sinTag : modTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Return a byte array.
    private static byte[] getObjectFile(String filePath) {
        FileInputStream fileInputStream = null;
        byte[] bytesArray = null;

        try {
            File file = new File(filePath);
            bytesArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(bytesArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
```

```
        }

        return bytesArray;
    }
}
```

使用 S3Client 將物件上傳至儲存貯體並設定中繼資料。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = """

                Usage:
                  <bucketName> <objectKey> <objectPath>\s

                Where:
                  bucketName - The Amazon S3 bucket to upload an object into.
                  objectKey - The object to upload (for example, book.pdf).
                  objectPath - The path where the file is located (for example,
 C:/AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(USAGE);
```

```
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
 bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file
 into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String
 objectKey, String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .metadata(metadata)
                    .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket
 " + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

使用 S3Client 將物件上傳至儲存貯體並設定物件保留值。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <key> <bucketName>\s

                Where:
                    key - The name of the object (for example, book.pdf).\s
                    bucketName - The Amazon S3 bucket name that contains the
 object (for example, bucket1).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
```

```
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }

    public static void setRentionPeriod(S3Client s3, String key, String bucket) {
        try {
            LocalDate localDate = LocalDate.parse("2020-07-17");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            ObjectLockRetention lockRetention = ObjectLockRetention.builder()
                    .mode("COMPLIANCE")
                    .retainUntilDate(instant)
                    .build();

            PutObjectRetentionRequest retentionRequest =
 PutObjectRetentionRequest.builder()
                    .bucket(bucket)
                    .key(key)
                    .bypassGovernanceRetention(true)
                    .retention(lockRetention)
                    .build();

            // To set Retention on an object, the Amazon S3 bucket must support
 object
            // locking, otherwise an exception is thrown.
            s3.putObjectRetention(retentionRequest);
            System.out.print("An object retention configuration was successfully
 placed on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考PutObject中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

上傳物件。

```
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new PutObjectCommand({
    Bucket: "test-bucket",
    Key: "hello-s3.txt",
    Body: "Hello S3!",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutObject中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK PutObject中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

將物件上傳至儲存貯體。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考PutObject中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會將單一檔案「local-sample.txt」上傳至 Amazon S3，並在儲存貯體「測試檔案」中建立含有金鑰「sample.txt」的物件。

```
Write-S3Object -BucketName test-files -Key "sample.txt" -File .\local-sample.txt
```

範例 2：此命令會將單一檔案「sample.txt」上傳至 Amazon S3，並在儲存貯體「測試檔案」中建立含有金鑰「sample.txt」的物件。如果未提供-Key 參數，檔案名稱會用作 S3 物件金鑰。

```
Write-S3Object -BucketName test-files -File .\sample.txt
```

範例 3：此命令會將單一檔案「local-sample.txt」上傳至 Amazon S3，並在儲存貯體「測試檔案」中建立含有金鑰「prefix/to/sample.txt」的物件。

```
Write-S3Object -BucketName test-files -Key "prefix/to/sample.txt" -File .\local-
sample.txt
```

範例 4：此命令會將子目錄「Script」中的所有檔案上傳到值區「test file」，並將通用 key prefix "SampleScripts" 套用至每個物件。每個上傳的文件將有一個鍵「SampleScripts/文件名」，其中「文件名」不同。

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\
```

範例 5：此命令會將本機主管「Script」中的所有 *.ps1 檔案上載至儲存貯體「test file」，並將通用 key prefix「」SampleScripts 套用至每個物件。每個上傳的文件將有一個鍵「SampleScripts/文件名 .ps1」，其中「文件名」不同。

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\
 -SearchPattern *.ps1
```

範例 6：此命令會建立包含指定內容字串的新 S3 物件，其中包含索引鍵為「sample.txt」。

```
Write-S3Object -BucketName test-files -Key "sample.txt" -Content "object
 contents"
```

範例 7：此指令會上傳指定的檔案 (檔案名稱用作索引鍵)，並將指定的標籤套用至新物件。

```
Write-S3Object -BucketName test-files -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

範例 8：此命令遞迴上傳指定的資料夾，並將指定的標籤套用至所有新物件。

```
Write-S3Object -BucketName test-files -Folder . -KeyPrefix "TaggedFiles" -Recurse
 -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutObject式參考中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string.
 When this
                     argument is a string, it is interpreted as a file name,
 which is
                     opened in read bytes mode.
        """
        put_data = data
        if isinstance(data, str):
            try:
                put_data = open(data, "rb")
            except IOError:
                logger.exception("Expected file name or binary data, got '%s'.",
 data)
```

```
                raise

        try:
            self.object.put(Body=put_data)
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
        finally:
            if getattr(put_data, "close", None):
                put_data.close()
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件PutObject中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用受管的上傳工具上傳檔案 (Object.upload_file)。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object
```

```
  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

使用 Object.put.上傳檔案。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object
```

```
  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, "rb") do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

使用 Object.Pet 上傳檔案並新增伺服器端加密。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
```

```
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考PutObject中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
pub async fn upload_object(
    client: &Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<PutObjectOutput, SdkError<PutObjectError>> {
    let body = ByteStream::from_path(Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK PutObject中的 Rust API 參考資料。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    "Get contents of file from application server."
    DATA lv_body TYPE xstring.
    OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.
    READ DATASET iv_file_name INTO lv_body.
    CLOSE DATASET iv_file_name.

    "Upload/put an object to an S3 bucket."
    TRY.
        lo_s3->putobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_file_name
```

```
            iv_body = lv_body
        ).
        MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- 如需 API 詳細資訊，請參閱 AWS SDK PutObject中的 SAP ABAP API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

將檔案從本機儲存體上傳至儲存貯體。

```
    public func uploadFile(bucket: String, key: String, file: String) async
throws {
        let fileUrl = URL(fileURLWithPath: file)
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.from(data: fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }
```

將 Swift 資料物件的內容上傳至儲存貯體。

```
    public func createFile(bucket: String, key: String, withData data: Data)
async throws {
        let dataStream = ByteStream.from(data: data)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK PutObject中的斯威夫特 API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutObjectAcl配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutObjectAcl。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 管理存取控制清單 (ACL)

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
 &objectKey, const Aws::String &ownerID,
                               const Aws::String &granteePermission, const
 Aws::String &granteeType,
                               const Aws::String &granteeID, const Aws::String
 &granteeEmailAddress,
                               const Aws::String &granteeURI, const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutObjectAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);
    request.SetKey(objectKey);
```

```
    Aws::S3::Model::PutObjectAclOutcome outcome =
            s3Client.PutObjectAcl(request);

    if (!outcome.IsSuccess()) {
        auto error = outcome.GetError();
        std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
                  << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param access: Human readable string.
 \return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration.
*/
Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
```

```
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考PutObjectAcl中的。

CLI

AWS CLI

下列命令會授`full control`予兩位 AWS 使用者 (user1@example.com 和 user2@example.com)，並授予所有人的`read`權限：

```
aws s3api put-object-acl --bucket MyBucket --key file.txt --grant-full-control
 emailaddress=user1@example.com,emailaddress=user2@example.com --grant-read
 uri=http://acs.amazonaws.com/groups/global/AllUsers
```

如需有關自訂 ACL (s3api ACL 命令，例如，使用相同的速記引數表示法) 的詳細資訊`put-object-acl`，請參閱 http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutObjectAcl中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class ObjectWrapper:
```

```
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put_acl(self, email):
        """
        Applies an ACL to the object that grants read access to an AWS user
identified
        by email address.

        :param email: The email address of the user to grant access.
        """
        try:
            acl = self.object.Acl()
            # Putting an ACL overwrites the existing ACL, so append new grants
            # if you want to preserve existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {"Type": "AmazonCustomerByEmail", "EmailAddress":
email},
                    "Permission": "READ",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted read access to %s.", email)
        except ClientError:
            logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件PutObjectAcl中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutObjectLegalHold配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutObjectLegalHold。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 鎖定 Amazon S3 對象

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/// <summary>
/// Set or modify a legal hold on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="holdStatus">The On or Off status for the legal hold.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectLegalHold(string bucketName,
    string objectKey, ObjectLockLegalHoldStatus holdStatus)
{
    try
    {
        var request = new PutObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            LegalHold = new ObjectLockLegalHold()
            {
                Status = holdStatus
```

```
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考PutObjectLegalHold中的。

CLI

AWS CLI

若要將「合法持有」套用至物件

下列put-object-legal-hold範例會對物件設定「合法保留」doc1.rtf。

```
aws s3api put-object-legal-hold \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf \
    --legal-hold Status=ON
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutObjectLegalHold中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
 {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }
```

```
 return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考PutObjectLegalHold中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   // Set or modify a legal hold on an object in an S3 bucket.
   public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
       ObjectLockLegalHold legalHold ;
       if (legalHoldOn) {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.ON)
               .build();
       } else {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.OFF)
               .build();
       }

       PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
           .bucket(bucketName)
           .key(objectKey)
           .legalHold(legalHold)
           .build();

       getClient().putObjectLegalHold(legalHoldRequest) ;
       System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
```

```
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考PutObjectLegalHold中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import { PutObjectLegalHoldCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new PutObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    LegalHold: {
      // Set the status to 'ON' to place a legal hold on the object.
      // Set the status to 'OFF' to remove the legal hold.
      Status: "ON",
    },
    // Optionally, you can provide additional parameters
    // ChecksumAlgorithm: "ALGORITHM",
    // ContentMD5: "MD5_HASH",
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // VersionId: "OBJECT_VERSION_ID",
  });
```

```
  try {
    const response = await client.send(command);
    console.log(
      `Object legal hold status: ${response.$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME", "OBJECT_KEY");
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutObjectLegalHold中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutObjectLockConfiguration配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutObjectLockConfiguration。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 鎖定 Amazon S3 對象

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

設定值區的物件鎖定組態。

```
    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

設定值區的預設保留期間。

```
   /// <summary>
   /// Set or modify a retention period on an S3 bucket.
   /// </summary>
   /// <param name="bucketName">The bucket to modify.</param>
   /// <param name="retention">The retention mode.</param>
   /// <param name="retainUntilDate">The date for retention until.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
   {
       var enabledString = enableObjectLock ? "Enabled" : "Disabled";
       var timeDifference = retainUntilDate.Subtract(DateTime.Now);
       try
       {
           // First, enable Versioning on the bucket.
           await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
           {
               BucketName = bucketName,
               VersioningConfig = new S3BucketVersioningConfig()
               {
                   EnableMfaDelete = false,
                   Status = VersionStatus.Enabled
               }
           });

           var request = new PutObjectLockConfigurationRequest()
           {
               BucketName = bucketName,
               ObjectLockConfiguration = new ObjectLockConfiguration()
               {
                   ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                   Rule = new ObjectLockRule()
                   {
                       DefaultRetention = new DefaultRetention()
                       {
                           Mode = retention,
                           Days = timeDifference.Days // Can be specified in
days or years but not both.
                       }
                   }
               }
```

```
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考[PutObjectLockConfiguration](#)中的。

CLI

AWS CLI

若要在值區上設定物件鎖定組態

下列`put-object-lock-configuration`範例會在指定的值區上設定 50 天的物件鎖定。

```
aws s3api put-object-lock-configuration \
    --bucket my-bucket-with-object-lock \
    --object-lock-configuration '{ "ObjectLockEnabled": "Enabled", "Rule":
 { "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[PutObjectLockConfiguration](#)中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

設定值區的物件鎖定組態。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
 string) error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
   Status:    types.BucketVersioningStatusEnabled,
  },
 }
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }

 input := &s3.PutObjectLockConfigurationInput{
```

```
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}
```

設定值區的預設保留期間。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ModifyDefaultBucketRetention modifies the default retention period of an
 existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
```

```
     Mode: retentionMode,
    },
   },
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考PutObjectLockConfiguration中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

設定值區的物件鎖定組態。

```
    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
 VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();
```

```
            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on
"+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
        }
    }
```

設定值區的預設保留期間。

```
    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();
```

```
        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();


getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName
+".");
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[PutObjectLockConfiguration](#)中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

設定值區的物件鎖定組態。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import {
  PutObjectLockConfigurationCommand,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 */
export const main = async (client, bucketName) => {
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
 bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
    },
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // Token: "OPTIONAL_TOKEN",
  });

  try {
    const response = await client.send(command);
    console.log(
      `Object Lock Configuration updated: ${response.$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME");
}
```

設定值區的預設保留期間。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import {
  PutObjectLockConfigurationCommand,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 */
export const main = async (client, bucketName) => {
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
 bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
      Rule: {
        DefaultRetention: {
          Mode: "GOVERNANCE",
          Years: 3,
        },
      },
    },
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // Token: "OPTIONAL_TOKEN",
  });

  try {
    const response = await client.send(command);
    console.log(
      `Default Object Lock Configuration updated: ${response.
$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};
```

```
// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME");
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutObjectLockConfiguration中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutObjectRetention配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutObjectRetention。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 鎖定 Amazon S3 對象

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
```

```
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考PutObjectRetention中的。

CLI

AWS CLI

若要設定物件的物件保留組態

下列put-object-retention範例會設定指定物件的物件保留組態，直到 2025-01-01 為止。

```
aws s3api put-object-retention \
```

```
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf \
    --retention '{ "Mode": "GOVERNANCE", "RetainUntilDate":
 "2025-01-01T00:00:00" }'
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutObjectRetention中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }
```

```
 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考PutObjectRetention中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   // Set or modify a retention period on an object in an S3 bucket.
   public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
{
       // Calculate the instant one day from now.
       Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

       // Convert the Instant to a ZonedDateTime object with a specific time
zone.
       ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

       // Define a formatter for human-readable output.
       DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

       // Format the ZonedDateTime object to a human-readable date string.
```

```
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考PutObjectRetention中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { fileURLToPath } from "url";
import { PutObjectRetentionCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
```

```
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new PutObjectRetentionCommand({
    Bucket: bucketName,
    Key: objectKey,
    BypassGovernanceRetention: false,
    // ChecksumAlgorithm: "ALGORITHM",
    // ContentMD5: "MD5_HASH",
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    Retention: {
      Mode: "GOVERNANCE", // or "COMPLIANCE"
      RetainUntilDate: new Date(new Date().getTime() + 24 * 60 * 60 * 1000),
    },
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const response = await client.send(command);
    console.log(
      `Object Retention settings updated: ${response.$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME", "OBJECT_KEY");
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutObjectRetention中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會針對指定 S3 儲存貯體中的 'testfile.txt' 物件啟用治理保留模式，直到 2019 年 12 月 31 日 00:00:00 '為止。

```
Write-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutObjectRetention式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭RestoreObject配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用RestoreObject。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to restore an archived object in an Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class RestoreArchivedObject
{
    public static void Main()
    {
        string bucketName = "doc-example-bucket";
        string objectKey = "archived-object.txt";
```

```
            // Specify your bucket region (an example region is shown).
            RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

            IAmazonS3 client = new AmazonS3Client(bucketRegion);
            RestoreObjectAsync(client, bucketName, objectKey).Wait();
        }

        /// <summary>
        /// This method restores an archived object from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// RestoreObjectAsync.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object was located before it was archived.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object to restore.</param>
        public static async Task RestoreObjectAsync(IAmazonS3 client, string
bucketName, string objectKey)
        {
            try
            {
                var restoreRequest = new RestoreObjectRequest
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Days = 2,
                };
                RestoreObjectResponse response = await
client.RestoreObjectAsync(restoreRequest);

                // Check the status of the restoration.
                await CheckRestorationStatusAsync(client, bucketName, objectKey);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Error: {amazonS3Exception.Message}");
            }
        }

        /// <summary>
        /// This method retrieves the status of the object's restoration.
        /// </summary>
```

```
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetObjectMetadataAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
        /// S3 bucket which contains the archived object.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object you want to restore.</param>
        public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
 string bucketName, string objectKey)
        {
            GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
            };

            GetObjectMetadataResponse response = await
 client.GetObjectMetadataAsync(metadataRequest);

            var restStatus = response.RestoreInProgress ? "in-progress" :
 "finished or failed";
            Console.WriteLine($"Restoration status: {restStatus}");
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考RestoreObject中的。

CLI

AWS CLI

若要建立物件的還原要求

下列restore-object範例會將儲存貯體的指定 Amazon S3 Glacier 物件還原 10 天。my-glacier-bucket

```
aws s3api restore-object \
    --bucket my-glacier-bucket \
    --key doc1.rtf \
```

```
    --restore-request Days=10
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考RestoreObject中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 *  For more information about restoring an object, see "Restoring an archived
 object" at
 *  https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 *  Before running this Java V2 code example, set up your development
 environment, including your credentials.
 *
 *  For more information, see the following documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <bucketName> <keyName> <expectedBucketOwner>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name of an object with a Storage class
value of Glacier.\s
                    expectedBucketOwner - The account that owns the bucket (you
can obtain this value from the AWS Management Console).\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }

    public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
        try {
            RestoreRequest restoreRequest = RestoreRequest.builder()
                    .days(10)

.glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
                    .build();

            RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
                    .expectedBucketOwner(expectedBucketOwner)
                    .bucket(bucketName)
                    .key(keyName)
                    .restoreRequest(restoreRequest)
                    .build();

            s3.restoreObject(objectRequest);
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考RestoreObject中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭SelectObjectContent配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用SelectObjectContent。

CLI

AWS CLI

根據 SQL 陳述式篩選 Amazon S3 物件的內容

下列select-object-content範例會my-data-file.csv使用指定的 SQL 陳述式篩選物件，並將輸出傳送至檔案。

```
aws s3api select-object-content \
    --bucket my-bucket \
    --key my-data-file.csv \
    --expression "select * from s3object limit 100" \
    --expression-type 'SQL' \
    --input-serialization '{"CSV": {}, "CompressionType": "NONE"}' \
    --output-serialization '{"CSV": {}}' "output.csv"
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考SelectObjectContent中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

下列範例顯示使用 JSON 物件的查詢。完整範例也會顯示 CSV 物件的使用方式。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import
 software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

```
public class SelectObjectContentExample {
    static final Logger logger =
 LoggerFactory.getLogger(SelectObjectContentExample.class);
    static final String BUCKET_NAME = "select-object-content-" +
 UUID.randomUUID();
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/
countries/master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
 SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where
 c.area < 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
                .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
                .compressionType(CompressionType.NONE)
                .build();

        OutputSerialization outputSerialization = OutputSerialization.builder()
                .json(JSONOutput.builder().recordDelimiter(null).build())
                .build();
```

```
        // Build the SelectObjectContentRequest.
        SelectObjectContentRequest select = SelectObjectContentRequest.builder()
                .bucket(BUCKET_NAME)
                .key(FILE_JSON)
                .expression(queryExpression)
                .expressionType(ExpressionType.SQL)
                .inputSerialization(inputSerialization)
                .outputSerialization(outputSerialization)
                .build();

        EventStreamInfo eventStreamInfo = new EventStreamInfo();
        // Call the selectObjectContent method with the request and a response
handler.
        // Supply an EventStreamInfo object to the response handler to gather
records and information from the response.
        s3AsyncClient.selectObjectContent(select,
buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record
->
                record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times",
fileType, eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler
buildResponseHandler(EventStreamInfo eventStreamInfo) {
        // Use a Visitor to process the response stream. This visitor logs
information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
SelectObjectContentResponseHandler.Visitor.builder()
                .onRecords(r -> {
                    logger.info("Record event received.");
                    eventStreamInfo.addRecord(r.payload().asUtf8String());
                    eventStreamInfo.incrementOnRecordsCalled();
                })
```

```
                .onCont(ce -> {
                    logger.info("Continuation event received.");
                    eventStreamInfo.incrementContinuationEvents();
                })
                .onProgress(pe -> {
                    Progress progress = pe.details();
                    logger.info("Progress event received:\n bytesScanned:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                            progress.bytesScanned(),
                            progress.bytesProcessed(),
                            progress.bytesReturned());
                })
                .onEnd(ee -> logger.info("End event received."))
                .onStats(se -> {
                    logger.info("Stats event received.");
                    eventStreamInfo.addStats(se.details());
                })
                .build();

        // Build the SelectObjectContentResponseHandler with the visitor that
 processes the stream.
        return SelectObjectContentResponseHandler.builder()
                .subscriber(visitor).build();
    }

    // The EventStreamInfo class is used to store information gathered while
 processing the response stream.
    static class EventStreamInfo {
        private final List<String> records = new ArrayList<>();
        private Integer countOnRecordsCalled = 0;
        private Integer countContinuationEvents = 0;
        private Stats stats;

        void incrementOnRecordsCalled() {
            countOnRecordsCalled++;
        }

        void incrementContinuationEvents() {
            countContinuationEvents++;
        }

        void addRecord(String record) {
            records.add(record);
        }
```

```
        void addStats(Stats stats) {
            this.stats = stats;
        }

        public List<String> getRecords() {
            return records;
        }

        public Integer getCountOnRecordsCalled() {
            return countOnRecordsCalled;
        }

        public Integer getCountContinuationEvents() {
            return countContinuationEvents;
        }

        public Stats getStats() {
            return stats;
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考SelectObjectContent中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭UploadPart配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用UploadPart。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 執行分段上傳
- 使用檢查總和

CLI

AWS CLI

下列指令會上傳使用指`create-multipart-upload`令起始的分段上傳中的第一部分：

```
aws s3api upload-part --bucket my-bucket --key 'multipart/01' --part-number 1 --
body part01 --upload-id
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
```

body此選項會使用本機檔案的名稱或路徑進行上載 (請勿使用 file://前置詞)。最小零件大小為 5 MB。上傳 ID 由傳回`create-multipart-upload`，也可以使用擷取`list-multipart-uploads`。當您建立多部分上傳時，會指定值區和索引鍵。

輸出：

```
{
    "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

儲存每個零件的 ETag 值以供日後使用。他們需要完成分段上傳。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考UploadPart中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
```

```
            .send()
            .await?;
        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );

   let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();
```

- 如需 API 的詳細資訊，請參閱 AWS SDK UploadPart中的 Rust API 參考資料。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS 開發套件的 Amazon S3 案例

下列程式碼範例說明如何使用 AWS 開發套件在 Amazon S3 中實作常見案例。這些案例會向您展示如何呼叫 Amazon S3 中的多個函數來完成特定任務。每個案例都包含一個連結 GitHub，您可以在其中找到如何設定和執行程式碼的指示。

範例

- 使用開發套件 AWS 為 Amazon S3 建立預先簽署的 URL
- 使用 AWS 開發套件列出 Amazon S3 物件的網頁
- 使用開發套件刪除不完整的分段上傳到 Amazon S3 AWS
- 將 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的所有物件下載至本機目錄
- 使用開發套件從多區域存取點取得 Amazon S3 物件 AWS
- 使用 AWS 開發套件從 Amazon S3 儲存貯體取得物件，並指定如果修改後的標頭
- 使用開 AWS 發套件開始使用 Amazon S3 儲存貯體和物件
- 使用開 AWS 發套件開始使用 Amazon S3 物件的加密功能
- 使用開 AWS 發套件開始使用 Amazon S3 物件的標籤
- 使用 AWS 開發套件取得 Amazon S3 物件的合法保留組態

- 使用 AWS 開發套件使用 Amazon S3 物件鎖定功能
- 使用開發套件 AWS 管理 Amazon S3 儲存貯體的存取控制清單 (ACL)
- 使用開發套件使用 Lambda 函數批次管理版本控制的 Amazon S3 物件 AWS
- 使用開發套件 AWS 剖析 Amazon S3 URI
- 使用開發套件執行 Amazon S3 物件的多部分複本 AWS
- 使用開發套件執行 Amazon S3 物件的多部分上傳 AWS
- 使用 AWS 開發套件追蹤 Amazon S3 物件上傳或下載
- 使用 AWS SDK 進行單元和集成測試的示例方法
- 以遞迴的方式將本機目錄上傳至 Amazon Simple Storage Service (Amazon S3) 儲存貯體
- 使用 AWS 開發套件在 Amazon S3 上傳或下載大型檔案
- 使用 AWS 開發套件將大小不明的串流上傳至 Amazon S3 物件
- 使用總和檢查碼搭配使用開發套件的 Amazon S3 物件 AWS
- 使用開發套件使用 Amazon S3 版本控制物件 AWS

使用開發套件 AWS 為 Amazon S3 建立預先簽署的 URL

下列程式碼範例示範如何建立適用於 Amazon S3 預先簽署的 URL 並上傳物件。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

產生可以於限定時間內執行 Amazon S3 動作的預先簽署的 URL。

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

public class GenPresignedUrl
```

```
    {
        public static void Main()
        {
            const string bucketName = "doc-example-bucket";
            const string objectKey = "sample.txt";

            // Specify how long the presigned URL lasts, in hours
            const double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
 KMS, you must specify Signature Version 4.
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);

            string urlString = GeneratePresignedURL(s3Client, bucketName,
 objectKey, timeoutDuration);
            Console.WriteLine($"The generated URL is: {urlString}.");
        }

        /// <summary>
        /// Generate a presigned URL that can be used to access the file named
        /// in the objectKey parameter for the amount of time specified in the
        /// duration parameter.
        /// </summary>
        /// <param name="client">An initialized S3 client object used to call
        /// the GetPresignedUrl method.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// object for which to create the presigned URL.</param>
        /// <param name="objectKey">The name of the object to access with the
        /// presigned URL.</param>
        /// <param name="duration">The length of time for which the presigned
        /// URL will be valid.</param>
        /// <returns>A string representing the generated presigned URL.</returns>
```

```
        public static string GeneratePresignedURL(IAmazonS3 client, string
bucketName, string objectKey, double duration)
        {
            string urlString = string.Empty;
            try
            {
                var request = new GetPreSignedUrlRequest()
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Expires = DateTime.UtcNow.AddHours(duration),
                };
                urlString = client.GetPreSignedURL(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:'{ex.Message}'");
            }

            return urlString;
        }
    }
```

產生預先簽章的 URL 並使用該 URL 執行上傳。

```
    using System;
    using System.IO;
    using System.Net.Http;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket using a presigned URL. The code first
    /// creates a presigned URL and then uses it to upload an object to an
    /// Amazon S3 bucket using that URL.
    /// </summary>
    public class UploadUsingPresignedURL
    {
```

```
        private static HttpClient httpClient = new HttpClient();

        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "samplefile.txt";
            string filePath = $"source\\{keyName}";

            // Specify how long the signed URL will be valid in hours.
            double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
 KMS, you must specify Signature Version 4.
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 client = new AmazonS3Client(RegionEndpoint.USEast1);

            var url = GeneratePreSignedURL(client, bucketName, keyName,
 timeoutDuration);
            var success = await UploadObject(filePath, url);

            if (success)
            {
                Console.WriteLine("Upload succeeded.");
            }
            else
            {
                Console.WriteLine("Upload failed.");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket using the presigned URL
 passed in
```

```
        /// the url parameter.
        /// </summary>
        /// <param name="filePath">The path (including file name) to the local
        /// file you want to upload.</param>
        /// <param name="url">The presigned URL that will be used to upload the
        /// file to the Amazon S3 bucket.</param>
        /// <returns>A Boolean value indicating the success or failure of the
        /// operation, based on the HttpWebResponse.</returns>
        public static async Task<bool> UploadObject(string filePath, string url)
        {
            using var streamContent = new StreamContent(
                new FileStream(filePath, FileMode.Open, FileAccess.Read));

            var response = await httpClient.PutAsync(url, streamContent);
            return response.IsSuccessStatusCode;
        }

        /// <summary>
        /// Generates a presigned URL which will be used to upload an object to
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetPreSignedURL.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which
 the
        /// presigned URL will point.</param>
        /// <param name="objectKey">The name of the file that will be uploaded.</
param>
        /// <param name="duration">How long (in hours) the presigned URL will
        /// be valid.</param>
        /// <returns>The generated URL.</returns>
        public static string GeneratePreSignedURL(
            IAmazonS3 client,
            string bucketName,
            string objectKey,
            double duration)
        {
            var request = new GetPreSignedUrlRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                Verb = HttpVerb.PUT,
                Expires = DateTime.UtcNow.AddHours(duration),
```

```
            };

            string url = client.GetPreSignedURL(request);
            return url;
        }
    }
```

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

產生預先簽署的 URL 以下載物件。

```
//! Routine which demonstrates creating a pre-signed URL to download an object
 from an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param expirationSeconds: Expiration in seconds for pre-signed URL.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String
 &bucketName,
                                                      const Aws::String &key,
                                                      uint64_t expirationSeconds,
                                                      const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_GET,
                                       expirationSeconds);
}
```

使用下載。

```
static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to get an object from a bucket.
  \param resultString: A string to hold the result.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedObjectUrl(const Aws::String &presignedURL,
                                                 Aws::String &resultString) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());
```

```
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    resultString = outWriteString.str();

    if (resultString.find("<?xml") == 0) {
        std::cerr << "Failed to get object, response:\n" << resultString <<
 std::endl;
        return false;
    }

    return true;
}
```

產生預先簽署的 URL 以上傳物件。

```
//! Routine which demonstrates creating a pre-signed URL to upload an object to
 an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String
 &bucketName,
                                                      const Aws::String &key,
                                                      uint64_t expirationSeconds,
                                                      const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
```

```
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_PUT,
                                       expirationSeconds);
}
```

使用庫曲線上傳。

```
static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test putObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to put an object in a bucket.
  \param data: Body of the putObject request.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                 const Aws::String &data) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);

    if (result != CURLE_OK) {
```

```
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

    if (result != CURLE_OK) {
```

```
        std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    std::string outString = outWriteString.str();
    if (outString.empty()) {
        std::cout << "Successfully put object." << std::endl;
        return true;
    } else {
        std::cout << "A server error was encountered, output:\n" << outString
                  << std::endl;
        return false;
    }
}
```

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

建立可包裝 S3 預先簽章動作的函數。

```
// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign
 actions
// used in the examples.
// It contains PresignClient, a client that is used to presign requests to Amazon
 S3.
```

```
// Presigned requests contain temporary credentials and can be made from any HTTP
 client.
type Presigner struct {
 PresignClient *s3.PresignClient
}



// GetObject makes a presigned request that can be used to get an object from a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) GetObject(
 bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignGetObject(context.TODO(),
 &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// PutObject makes a presigned request that can be used to put an object in a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
 bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignPutObject(context.TODO(),
 &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
```

```
  log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// DeleteObject makes a presigned request that can be used to delete an object
 from a bucket.
func (presigner Presigner) DeleteObject(bucketName string, objectKey string)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignDeleteObject(context.TODO(),
 &s3.DeleteObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to delete object %v. Here's why:
 %v\n", objectKey, err)
 }
 return request, err
}
```

執行互動式範例，產生並使用預先簽章的 URL 來上傳、下載和刪除 S3 物件。

```
// RunPresigningScenario is an interactive example that shows you how to get
 presigned
// HTTP requests that you can use to move data into and out of Amazon Simple
 Storage
// Service (Amazon S3). The presigned requests contain temporary credentials and
 can
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
// 2. Use the net/http package to use the presigned request to upload a local
 file to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the
 object to a local file.
```

```
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
// This example creates an Amazon S3 presign client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
 mocked
// during testing.
func RunPresigningScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner, httpRequester IHttpRequester) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 presigning demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}
 presignClient := s3.NewPresignClient(s3Client)
 presigner := actions.Presigner{PresignClient: presignClient}

 bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
  "you own or one you want to create:", demotools.NotEmpty{})
 bucketExists, err := bucketBasics.BucketExists(bucketName)
 if err != nil {
  panic(err)
 }
 if !bucketExists {
  err = bucketBasics.CreateBucket(bucketName, sdkConfig.Region)
  if err != nil {
   panic(err)
  } else {
   log.Println("Bucket created.")
  }
```

```
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
 demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
 panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedPutRequest.Method,
 presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
if err != nil {
 panic(err)
}
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
uploadFile)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedPutRequest.Method,
 uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
 presignedGetRequest.URL)
```

```
 log.Println("Using net/http to send the request...")
 getResponse, err := httpRequester.Get(presignedGetRequest.URL)
 if err != nil {
  panic(err)
 }
 log.Printf("%v object %v with presigned URL returned %v.",
 presignedGetRequest.Method,
  uploadKey, getResponse.StatusCode)
 defer getResponse.Body.Close()
 downloadBody, err := io.ReadAll(getResponse.Body)
 if err != nil {
  panic(err)
 }
 log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
 len(downloadBody))
 log.Println(strings.Repeat("-", 88))
 log.Println(string(downloadBody[:100]))
 log.Println(strings.Repeat("-", 88))

 log.Println("Let's presign a request to delete the object.")
 questioner.Ask("Press Enter when you're ready.")
 presignedDelRequest, err := presigner.DeleteObject(bucketName, uploadKey)
 if err != nil {
  panic(err)
 }
 log.Printf("Got a presigned %v request to URL:\n\t%v\n",
 presignedDelRequest.Method,
  presignedDelRequest.URL)
 log.Println("Using net/http to send the request...")
 delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
 if err != nil {
  panic(err)
 }
 log.Printf("%v object %v with presigned URL returned %v.\n",
 presignedDelRequest.Method,
  uploadKey, delResponse.StatusCode)
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

定義範例用來提出 HTTP 請求的 HTTP 請求包裝函式。

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
 during
// unit testing.
type IHttpRequester interface {
 Get(url string) (resp *http.Response, err error)
 Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
 error)
 Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
 scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
 return http.Get(url)
}
func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
 (resp *http.Response, err error) {
 putRequest, err := http.NewRequest("PUT", url, body)
 if err != nil {
  return nil, err
 }
 putRequest.ContentLength = contentLength
 return http.DefaultClient.Do(putRequest)
}
func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error)
 {
 delRequest, err := http.NewRequest("DELETE", url, nil)
 if err != nil {
  return nil, err
 }
 return http.DefaultClient.Do(delRequest)
}
```

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

為物件產生預先簽署的 URL，然後下載 (GET 要求)。

進口。

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
```

```
import java.time.Duration;
import java.util.UUID;
```

產生網址。

```
   /* Create a pre-signed URL to download an object in a subsequent GET request.
*/
   public String createPresignedGetUrl(String bucketName, String keyName) {
       try (S3Presigner presigner = S3Presigner.create()) {

           GetObjectRequest objectRequest = GetObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .build();

           GetObjectPresignRequest presignRequest =
GetObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))  // The URL will
expire in 10 minutes.
                   .getObjectRequest(objectRequest)
                   .build();

           PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
           logger.info("Presigned URL: [{}]",
presignedRequest.url().toString());
           logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

           return presignedRequest.url().toExternalForm();
       }
   }
```

使用下列三種方法中的任何一種下載物件。

使用 JDKHttpURLConnection（自 1.1 版以來）類進行下載。

```
   /* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
   public byte[] useHttpUrlConnectionToGet(String presignedUrlString) {
       ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
```

```
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
            connection.setRequestMethod("GET");
            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                IoUtils.copy(content, byteArrayOutputStream);
            }
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

使用 JDKHttpClient（自 v11 以來）類進行下載。

```
    /* Use the JDK HttpClient (since v11) class to do the download. */
    public byte[] useHttpClientToGet(String presignedUrlString) {
        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpResponse<InputStream> response = httpClient.send(requestBuilder
                            .uri(presignedUrl.toURI())
                            .GET()
                            .build(),
                    HttpResponse.BodyHandlers.ofInputStream());

            IoUtils.copy(response.body(), byteArrayOutputStream);

            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
```

```
        return byteArrayOutputStream.toByteArray();
    }
```

請使用適用於 Java SdkHttpClient 類別的 AWS SDK 來執行下載作業。

```
    /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
    public byte[] useSdkHttpClientToPut(String presignedUrlString) {

        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
        try {
            URL presignedUrl = new URL(presignedUrlString);
            SdkHttpRequest request = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.GET)
                    .uri(presignedUrl.toURI())
                    .build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                response.responseBody().ifPresentOrElse(
                        abortableInputStream -> {
                            try {
                                IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                            } catch (IOException e) {
                                throw new RuntimeException(e);
                            }
                        },
                        () -> logger.error("No response body."));

                logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
```

```
    }
```

為上傳產生預先簽署的 URL，然後上傳檔案 (PUT 要求)。

進口。

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

產生網址。

```
   /* Create a presigned URL to use in a subsequent PUT request */
   public String createPresignedUrl(String bucketName, String keyName,
Map<String, String> metadata) {
       try (S3Presigner presigner = S3Presigner.create()) {

           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .metadata(metadata)
                   .build();

           PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))  // The URL
expires in 10 minutes.
                   .putObjectRequest(objectRequest)
                   .build();


           PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
           String myURL = presignedRequest.url().toString();
           logger.info("Presigned URL to upload a file to: [{}]", myURL);
           logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

           return presignedRequest.url().toExternalForm();
       }
   }
```

使用下列三種方法中的任何一種上傳檔案物件。

使用 JDKHttpURLConnection（自 1.1 版以來）類進行上傳。

```
   /* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
   public void useHttpUrlConnectionToPut(String presignedUrlString, File
fileToPut, Map<String, String> metadata) {
       logger.info("Begin [{}] upload", fileToPut.toString());
       try {
           URL presignedUrl = new URL(presignedUrlString);
```

```
            HttpURLConnection connection = (HttpURLConnection)
 presignedUrl.openConnection();
            connection.setDoOutput(true);
            metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-
meta-" + k, v));
            connection.setRequestMethod("PUT");
            OutputStream out = connection.getOutputStream();

            try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
                 FileChannel inChannel = file.getChannel()) {
                ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is
 8k

                while (inChannel.read(buffer) > 0) {
                    buffer.flip();
                    for (int i = 0; i < buffer.limit(); i++) {
                        out.write(buffer.get());
                    }
                    buffer.clear();
                }
            } catch (IOException e) {
                logger.error(e.getMessage(), e);
            }

            out.close();
            connection.getResponseCode();
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

使用 JDKHttpClient（自 v11 以來）類進行上傳。

```
    /* Use the JDK HttpClient (since v11) class to do the upload. */
    public void useHttpClientToPut(String presignedUrlString, File fileToPut,
 Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));
```

```
        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            final HttpResponse<Void> response = httpClient.send(requestBuilder
                            .uri(new URL(presignedUrlString).toURI())

.PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
                            .build(),
                    HttpResponse.BodyHandlers.discarding());

            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

請使 AWS 用 Java V2 SdkHttpClient 類別來執行上傳作業。

```
    /* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
    public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        try {
            URL presignedUrl = new URL(presignedUrlString);

            SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.PUT)
                    .uri(presignedUrl.toURI());
            // Add headers
            metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" +
k, v));
            // Finish building the request.
            SdkHttpRequest request = requestBuilder.build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
                    .build();
```

```
            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
 sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}",
 response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

建立預先簽署的 URL 將物件上傳至儲存貯體。

```
import https from "https";
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
```

```
  });

  const signedUrlObject = await presigner.presign(
    new HttpRequest({ ...url, method: "PUT" }),
  );
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new PutObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

function put(url, data) {
  return new Promise((resolve, reject) => {
    const req = https.request(
      url,
      { method: "PUT", headers: { "Content-Length": new Blob([data]).size } },
      (res) => {
        let responseBody = "";
        res.on("data", (chunk) => {
          responseBody += chunk;
        });
        res.on("end", () => {
          resolve(responseBody);
        });
      },
    );
    req.on("error", (err) => {
      reject(err);
    });
    req.write(data);
    req.end();
  });
}

export const main = async () => {
  const REGION = "us-east-1";
  const BUCKET = "example_bucket";
  const KEY = "example_file.txt";

  // There are two ways to generate a presigned URL.
  // 1. Use createPresignedUrl without the S3 client.
```

```
  // 2. Use getSignedUrl in conjunction with the S3 client and GetObjectCommand.
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    const clientUrl = await createPresignedUrlWithClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    // After you get the presigned URL, you can provide your own file
    // data. Refer to put() above.
    console.log("Calling PUT using presigned URL without client");
    await put(noClientUrl, "Hello World");

    console.log("Calling PUT using presigned URL with client");
    await put(clientUrl, "Hello World");

    console.log("\nDone. Check your S3 console.");
  } catch (err) {
    console.error(err);
  }
};
```

建立預先簽署的 URL 從儲存貯體下載物件。

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
```

```
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(new HttpRequest(url));
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new GetObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

export const main = async () => {
  const REGION = "us-east-1";
  const BUCKET = "example_bucket";
  const KEY = "example_file.jpg";

  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    const clientUrl = await createPresignedUrlWithClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    console.log("Presigned URL without client");
    console.log(noClientUrl);
    console.log("\n");

    console.log("Presigned URL with client");
    console.log(clientUrl);
  } catch (err) {
    console.error(err);
  }
```

```
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

建立 GetObject 預先簽章的請求，並使用 URL 下載物件。

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET
 request to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()

    return objectContents
}
```

使用進階選項建立GetObject預先簽署的要求。

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be
 used 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

建立 PutObject 預先簽章的請求，並使用該請求上傳物件。

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)
```

```
    // Use the URL and any headers from the presigned HttpRequest in a subsequent
 HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }.put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- 如需詳細資訊，請參閱適用於 Kotlin 的AWS SDK 開發人員指南。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;

require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
```

```
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the
 S3 bucket to use: ");
        $key = $this->testable_readline("Next, provide the key of an object in
 the given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
 next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}

$runner = new PresignedURL();
$runner->run();
```

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

產生可以於限定時間內執行 S3 動作的預先簽署的 URL。使用 Requests package 搭配 URL 發出請求。

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method, Params=method_parameters,
 ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.", client_method
```

```
        )
        raise
    return url


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print("-" * 88)

    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key",
        help="For a GET operation, the key of the object in Amazon S3. For a "
        "PUT operation, the name of a file to upload.",
    )
    parser.add_argument("action", choices=("get", "put"), help="The action to
 perform.")
    args = parser.parse_args()

    s3_client = boto3.client("s3")
    client_action = "get_object" if args.action == "get" else "put_object"
    url = generate_presigned_url(
        s3_client, client_action, {"Bucket": args.bucket, "Key": args.key}, 1000
    )

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == "get":
        response = requests.get(url)
    elif args.action == "put":
        print("Putting data to the URL.")
        try:
            with open(args.key, "r") as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(
                f"Couldn't find {args.key}. For a PUT operation, the key must be
 the "
                f"name of a file that exists on your computer."
```

```
            )

    if response is not None:
        print("Got response:")
        print(f"Status: {response.status_code}")
        print(response.text)

    print("-" * 88)


if __name__ == "__main__":
    usage_demo()
```

產生預先簽署的 POST 請求以上傳檔案。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def generate_presigned_post(self, object_key, expires_in):
        """
        Generate a presigned Amazon S3 POST request to upload a file.
        A presigned POST can be used for a limited time to let someone without an
 AWS
        account upload a file to a bucket.

        :param object_key: The object key to identify the uploaded object.
        :param expires_in: The number of seconds the presigned POST is valid.
        :return: A dictionary that contains the URL and form fields that contain
                 required access data.
        """
        try:
            response = self.bucket.meta.client.generate_presigned_post(
```

```
                Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in
            )
            logger.info("Got presigned POST URL: %s", response["url"])
        except ClientError:
            logger.exception(
                "Couldn't get a presigned POST URL for bucket '%s' and object
 '%s'",
                self.bucket.name,
                object_key,
            )
            raise
        return response
```

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
require "aws-sdk-s3"
require "net/http"

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's
 why: #{e.message}"
```

```
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request("PUT", presigned_url.request_uri, object_content,
 "content_type" => "")
  end

  case response
  when Net::HTTPSuccess
    puts "Content uploaded!"
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

建立對 GET 和 PUT S3 物件的預先簽署請求。

```
async fn get_object(
    client: &Client,
```

```
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);
    let presigned_request = client
        .get_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in)?)
        .await?;

    println!("Object URI: {}", presigned_request.uri());

    Ok(())
}

async fn put_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);

    let presigned_request = client
        .put_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in)?)
        .await?;

    println!("Object URI: {}", presigned_request.uri());

    Ok(())
}
```

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS 開發套件列出 Amazon S3 物件的網頁

下列程式碼範例顯示如何在網頁中列出 Amazon S3 物件。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

下面的代碼是使調 AWS 用 SDK 的相關反應組件。可以在前面 GitHub 的鏈接中找到包含此組件的應用程序的可運行版本。

```
import { useEffect, useState } from "react";
import {
  ListObjectsCommand,
  ListObjectsCommandOutput,
  S3Client,
} from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import "./App.css";

function App() {
  const [objects, setObjects] = useState<
    Required<ListObjectsCommandOutput>["Contents"]
  >([]);

  useEffect(() => {
    const client = new S3Client({
      region: "us-east-1",
      // Unless you have a public bucket, you'll need access to a private bucket.
      // One way to do this is to create an Amazon Cognito identity pool, attach
 a role to the pool,
      // and grant the role access to the 's3:GetObject' action.
      //
      // You'll also need to configure the CORS settings on the bucket to allow
 traffic from
```

```
      // this example site. Here's an example configuration that allows all
 origins. Don't
      // do this in production.
      //[
      //  {
      //    "AllowedHeaders": ["*"],
      //    "AllowedMethods": ["GET"],
      //    "AllowedOrigins": ["*"],
      //    "ExposeHeaders": [],
      //  },
      //]
      //
      credentials: fromCognitoIdentityPool({
        clientConfig: { region: "us-east-1" },
        identityPoolId: "<YOUR_IDENTITY_POOL_ID>",
      }),
    });
    const command = new ListObjectsCommand({ Bucket: "bucket-name" });
    client.send(command).then(({ Contents }) => setObjects(Contents || []));
  }, []);

  return (
    <div className="App">
      {objects.map((o) => (
        <div key={o.ETag}>{o.Key}</div>
      ))}
    </div>
  );
}

export default App;
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ListObjects中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件刪除不完整的分段上傳到 Amazon S3 AWS

下列程式碼範例顯示如何刪除或停止不完整的 Amazon S3 多部分上傳。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

若要停止進行中或因任何原因不完整的分段上傳，您可以取得上傳的清單，然後將其刪除，如下列範例所示。

```
   public static void abortIncompleteMultipartUploadsFromList() {
       ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
               .bucket(bucketName)
               .build();

       ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
       List<MultipartUpload> uploads = response.uploads();

       AbortMultipartUploadRequest abortMultipartUploadRequest;
       for (MultipartUpload upload : uploads) {
           abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                   .bucket(bucketName)
                   .key(upload.key())
                   .expectedBucketOwner(accountId)
                   .uploadId(upload.uploadId())
                   .build();

           AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
           if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
               logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
           }
       }
   }
```

若要刪除日期之前或之後啟動的不完整分段上傳，您可以根據時間點選擇性地刪除多部分上傳，如下列範例所示。

```
   static void abortIncompleteMultipartUploadsOlderThan(Instant pointInTime) {
       ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
               .bucket(bucketName)
               .build();

       ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
       List<MultipartUpload> uploads = response.uploads();

       AbortMultipartUploadRequest abortMultipartUploadRequest;
       for (MultipartUpload upload : uploads) {
           logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
           if (upload.initiated().isBefore(pointInTime)) {
               abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                       .bucket(bucketName)
                       .key(upload.key())
                       .expectedBucketOwner(accountId)
                       .uploadId(upload.uploadId())
                       .build();

               AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
               if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                   logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
               }
           }
       }
   }
```

如果您在開始分段上傳之後可存取上傳 ID，您可以使用 ID 刪除進行中的上傳。

```
   static void abortMultipartUploadUsingUploadId() {
       String uploadId = startUploadReturningUploadId();
```

```
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
                .uploadId(uploadId)
                .bucket(bucketName)
                .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
 uploadId, bucketName);
        }
    }
```

若要持續刪除超過特定天數的未完整分段上傳，請為值區設定值區生命週期組態。下列範例會示範如何建立規則，以刪除超過 7 天的未完成上傳。

```
    static void abortMultipartUploadsUsingLifecycleConfig() {
        Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
                .abortIncompleteMultipartUpload(b -> b.
                        daysAfterInitiation(7))
                .status("Enabled")
                .filter(SdkBuilder::build) // Filter element is required.
                .build());

        // If the action is successful, the service sends back an HTTP 200
 response with an empty HTTP body.
        PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
                .bucket(bucketName)
                .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Rule to abort incomplete multipart uploads added to
 bucket.");
        } else {
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
 response.sdkHttpResponse().statusCode());
        }
    }
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。

- AbortMultipartUpload
- ListMultipartUploads
- PutBucketLifecycleConfiguration

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

將 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的所有物件下載至本機目錄

下列程式碼範例示範如何將 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的所有物件下載至本機目錄。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用 S3 下載TransferManager相同 S3 儲存貯體中的所有 S3 物件。檢視完整檔案並測試。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
```

```
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
            URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
 fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DownloadDirectory中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件從多區域存取點取得 Amazon S3 物件 AWS

下列程式碼範例顯示如何從多區域存取點取得物件。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

將 S3 用戶端設定為使用非對稱 SigV4 (SigV4a) 簽署演算法。

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric Sigv4 (Sigv4a)
signing algorithm.
            val sigV4AScheme = SigV4AsymmetricAuthScheme(CrtAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4AScheme)
            }
            return s3
        }
```

使用「多區域存取點」ARN 取代值區名稱來擷取物件。

```
    suspend fun getObjectFromMrap(
        s3: S3Client,
        mrapArn: String,
        keyName: String,
    ): String? {
        val request = GetObjectRequest {
            bucket = mrapArn // Use the ARN instead of the bucket name for object
operations.
            key = keyName
        }

        var stringObj: String? = null
        s3.getObject(request) { resp ->
            stringObj = resp.body?.decodeToString()
            if (stringObj != null) {
                println("Successfully read $keyName from $mrapArn")
            }
        }
        return stringObj
    }
```

- 如需詳細資訊，請參閱適用於 Kotlin 的AWS SDK 開發人員指南。
- 有關 API 的詳細信息，請參閱 AWS SDK GetObject中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS 開發套件從 Amazon S3 儲存貯體取得物件，並指定如果修改後的標頭

下列程式碼範例示範如何從 S3 儲存貯體中物件讀取資料，但僅在自上次擷取時後該儲存貯體尚未修改時才能進行此讀取。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
use aws_sdk_s3::{
    error::SdkError,
    operation::head_object::HeadObjectError,
    primitives::{ByteStream, DateTime, DateTimeFormat},
    Client, Error,
};
use tracing::{error, warn};

const KEY: &str = "key";
const BODY: &str = "Hello, world!";

/// Demonstrate how `if-modified-since` reports that matching objects haven't
/// changed.
///
/// # Steps
/// - Create a bucket.
/// - Put an object in the bucket.
/// - Get the bucket headers.
/// - Get the bucket headers again but only if modified.
/// - Delete the bucket.
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();

    // Get a new UUID to use when creating a unique bucket name.
```

```
    let uuid = uuid::Uuid::new_v4();

    // Load the AWS configuration from the environment.
    let client = Client::new(&aws_config::load_from_env().await);

    // Generate a unique bucket name using the previously generated UUID.
    // Then create a new bucket with that name.
    let bucket_name = format!("if-modified-since-{uuid}");
    client
        .create_bucket()
        .bucket(bucket_name.clone())
        .send()
        .await?;

    // Create a new object in the bucket whose name is `KEY` and whose
    // contents are `BODY`.
    let put_object_output = client
        .put_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .body(ByteStream::from_static(BODY.as_bytes()))
        .send()
        .await;

    // If the `PutObject` succeeded, get the eTag string from it. Otherwise,
    // report an error and return an empty string.
    let e_tag_1 = match put_object_output {
        Ok(put_object) => put_object.e_tag.unwrap(),
        Err(err) => {
            error!("{err:?}");
            String::new()
        }
    };

    // Request the object's headers.
    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await;

    // If the `HeadObject` request succeeded, create a tuple containing the
    // values of the headers `last-modified` and `etag`. If the request
```

```
    // failed, return the error in a tuple instead.
    let (last_modified, e_tag_2) = match head_object_output {
        Ok(head_object) => (
            Ok(head_object.last_modified().cloned().unwrap()),
            head_object.e_tag.unwrap(),
        ),
        Err(err) => (Err(err), String::new()),
    };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and first GetObject had differing eTags"
    );

    println!("First value of last_modified: {last_modified:?}");
    println!("First tag: {}\n", e_tag_1);

    // Send a second `HeadObject` request. This time, the `if_modified_since`
    // option is specified, giving the `last_modified` value returned by the
    // first call to `HeadObject`.
    //
    // Since the object hasn't been changed, and there are no other objects in
    // the bucket, there should be no matching objects.

    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .if_modified_since(last_modified.unwrap())
        .send()
        .await;

    // If the `HeadObject` request succeeded, the result is a typle containing
    // the `last_modified` and `e_tag_1` properties. This is _not_ the expected
    // result.
    //
    // The _expected_ result of the second call to `HeadObject` is an
    // `SdkError::ServiceError` containing the HTTP error response. If that's
    // the case and the HTTP status is 304 (not modified), the output is a
    // tuple containing the values of the HTTP `last-modified` and `etag`
    // headers.
    //
    // If any other HTTP error occurred, the error is returned as an
```

```
    // `SdkError::ServiceError`.

    let (last_modified, e_tag_2): (Result<DateTime, SdkError<HeadObjectError>>,
String) =
        match head_object_output {
            Ok(head_object) => (
                Ok(head_object.last_modified().cloned().unwrap()),
                head_object.e_tag.unwrap(),
            ),
            Err(err) => match err {
                SdkError::ServiceError(err) => {
                    // Get the raw HTTP response. If its status is 304, the
                    // object has not changed. This is the expected code path.
                    let http = err.raw();
                    match http.status().as_u16() {
                        // If the HTTP status is 304: Not Modified, return a
                        // tuple containing the values of the HTTP
                        // `last-modified` and `etag` headers.
                        304 => (
                            Ok(DateTime::from_str(
                                http.headers().get("last-modified").unwrap(),
                                DateTimeFormat::HttpDate,
                            )
                            .unwrap()),
                            http.headers().get("etag").map(|t|
t.into()).unwrap(),
                        ),
                        // Any other HTTP status code is returned as an
                        // `SdkError::ServiceError`.
                        _ => (Err(SdkError::ServiceError(err)), String::new()),
                    }
                }
                // Any other kind of error is returned in a tuple containing the
                // error and an empty string.
                _ => (Err(err), String::new()),
            },
        };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and second HeadObject had different eTags"
    );
```

```
    println!("Second value of last modified: {last_modified:?}");
    println!("Second tag: {}", e_tag_2);

    // Clean up by deleting the object and the bucket.
    client
        .delete_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await?;

    client
        .delete_bucket()
        .bucket(bucket_name.as_str())
        .send()
        .await?;

    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK GetObject中的 Rust API 參考資料。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開 AWS 發套件開始使用 Amazon S3 儲存貯體和物件

下列程式碼範例示範如何：

- 建立儲存貯體並上傳檔案到該儲存貯體。
- 從儲存貯體下載物件。
- 將物件複製至儲存貯體中的子文件夾。
- 列出儲存貯體中的物件。
- 刪除儲存貯體物件和該儲存貯體。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public class S3_Basics
    {
        public static async Task Main()
        {
            // Create an Amazon S3 client object. The constructor uses the
            // default user installed on the system. To work with Amazon S3
            // features in a different AWS Region, pass the AWS Region as a
            // parameter to the client constructor.
            IAmazonS3 client = new AmazonS3Client();
            string bucketName = string.Empty;
            string filePath = string.Empty;
            string keyName = string.Empty;

            var sepBar = new string('-', Console.WindowWidth);

            Console.WriteLine(sepBar);
            Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
            Console.WriteLine("procedures. This application will:");
            Console.WriteLine("\n\t1. Create a bucket");
            Console.WriteLine("\n\t2. Upload an object to the new bucket");
            Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
bucket");
            Console.WriteLine("\n\t4. List the items in the new bucket");
            Console.WriteLine("\n\t5. Delete all the items in the bucket");
            Console.WriteLine("\n\t6. Delete the bucket");
            Console.WriteLine(sepBar);

            // Create a bucket.
            Console.WriteLine($"\n{sepBar}");
            Console.WriteLine("\nCreate a new Amazon S3 bucket.\n");
            Console.WriteLine(sepBar);
```

```
            Console.Write("Please enter a name for the new bucket: ");
            bucketName = Console.ReadLine();

            var success = await S3Bucket.CreateBucketAsync(client, bucketName);
            if (success)
            {
                Console.WriteLine($"Successfully created bucket: {bucketName}.
\n");
            }
            else
            {
                Console.WriteLine($"Could not create bucket: {bucketName}.\n");
            }

            Console.WriteLine(sepBar);
            Console.WriteLine("Upload a file to the new bucket.");
            Console.WriteLine(sepBar);

            // Get the local path and filename for the file to upload.
            while (string.IsNullOrEmpty(filePath))
            {
                Console.Write("Please enter the path and filename of the file to
 upload: ");
                filePath = Console.ReadLine();

                // Confirm that the file exists on the local computer.
                if (!File.Exists(filePath))
                {
                    Console.WriteLine($"Couldn't find {filePath}. Try again.\n");
                    filePath = string.Empty;
                }
            }

            // Get the file name from the full path.
            keyName = Path.GetFileName(filePath);

            success = await S3Bucket.UploadFileAsync(client, bucketName, keyName,
 filePath);

            if (success)
            {
                Console.WriteLine($"Successfully uploaded {keyName} from
 {filePath} to {bucketName}.\n");
            }
```

```
            else
            {
                Console.WriteLine($"Could not upload {keyName}.\n");
            }

            // Set the file path to an empty string to avoid overwriting the
            // file we just uploaded to the bucket.
            filePath = string.Empty;

            // Now get a new location where we can save the file.
            while (string.IsNullOrEmpty(filePath))
            {
                // First get the path to which the file will be downloaded.
                Console.Write("Please enter the path where the file will be
downloaded: ");
                filePath = Console.ReadLine();

                // Confirm that the file exists on the local computer.
                if (File.Exists($"{filePath}\\{keyName}"))
                {
                    Console.WriteLine($"Sorry, the file already exists in that
location.\n");
                    filePath = string.Empty;
                }
            }

            // Download an object from a bucket.
            success = await S3Bucket.DownloadObjectFromBucketAsync(client,
bucketName, keyName, filePath);

            if (success)
            {
                Console.WriteLine($"Successfully downloaded {keyName}.\n");
            }
            else
            {
                Console.WriteLine($"Sorry, could not download {keyName}.\n");
            }

            // Copy the object to a different folder in the bucket.
            string folderName = string.Empty;

            while (string.IsNullOrEmpty(folderName))
            {
```

```
                Console.Write("Please enter the name of the folder to copy your
object to: ");
                folderName = Console.ReadLine();
            }

            while (string.IsNullOrEmpty(keyName))
            {
                // Get the name to give to the object once uploaded.
                Console.Write("Enter the name of the object to copy: ");
                keyName = Console.ReadLine();
            }

            await S3Bucket.CopyObjectInBucketAsync(client, bucketName, keyName,
folderName);

            // List the objects in the bucket.
            await S3Bucket.ListBucketContentsAsync(client, bucketName);

            // Delete the contents of the bucket.
            await S3Bucket.DeleteBucketContentsAsync(client, bucketName);

            // Deleting the bucket too quickly after deleting its contents will
            // cause an error that the bucket isn't empty. So...
            Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
            _ = Console.ReadLine();

            // Delete the bucket.
            await S3Bucket.DeleteBucketAsync(client, bucketName);
        }
    }
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2

- PutObject

Bash

AWS CLI 與 bash 腳本

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
###############################################################################
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function s3_getting_started() {
  {
    if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
      cd bucket-lifecycle-operations || exit

      source ./bucket_operations.sh
      cd ..
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the Amazon S3 getting started demo."
  echo_repeat "*" 88

  local bucket_name
  bucket_name=$(generate_random_name "doc-example-bucket")

  local region_code
  region_code=$(aws configure get region)
```

```
 if create_bucket -b "$bucket_name" -r "$region_code"; then
   echo "Created demo bucket named $bucket_name"
 else
   errecho "The bucket failed to create. This demo will exit."
   return 1
 fi

 local file_name
 while [ -z "$file_name" ]; do
   echo -n "Enter a file you want to upload to your bucket: "
   get_input
   file_name=$get_input_result

   if [ ! -f "$file_name" ]; then
     echo "Could not find file $file_name. Are you sure it exists?"
     file_name=""
   fi
 done

 local key
 key="$(basename "$file_name")"

 local result=0
 if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
   echo "Uploaded file $file_name into bucket $bucket_name with key $key."
 else
   result=1
 fi

 local destination_file
 destination_file="$file_name.download"
 if yes_no_input "Would you like to download $key to the file $destination_file?
(y/n) "; then
   if download_object_from_bucket "$bucket_name" "$destination_file" "$key";
then
     echo "Downloaded $key in the bucket $bucket_name to the file
$destination_file."
   else
     result=1
   fi
 fi

 if yes_no_input "Would you like to copy $key a new object key in your bucket?
(y/n) "; then
```

```
    local to_key
    to_key="demo/$key"
    if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
      echo "Copied $key in the bucket $bucket_name to the  $to_key."
    else
      result=1
    fi
  fi

  local bucket_items
  bucket_items=$(list_items_in_bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    result=1
  fi

  echo "Your bucket contains the following items."
  echo -e "Name\t\tSize"
  echo "$bucket_items"

  if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
 (y/n) "; then
    bucket_items=$(echo "$bucket_items" | cut -f 1)

    if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
      echo "The following items were deleted from the bucket $bucket_name"
      echo "$bucket_items"
    else
      result=1
    fi

    if delete_bucket "$bucket_name"; then
      echo "Deleted the bucket $bucket_name"
    else
      result=1
    fi
  fi

  return $result
}
```

這種情況下使用的 Amazon S3 函數。

```
##############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally
 unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi

  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}

################################################################################
```

```
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3

  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
```

```
################################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

################################################################################
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")
```

```
  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function delete_items_in_bucket
#
```

```
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}

###############################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
```

```
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
        /*!
          \param bucketName: The S3 bucket's name.
          \param client: An S3 client.
          \return bool: Function succeeded.
        */
        static bool
        deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

        //! Delete an object in an S3 bucket.
        /*!
          \param bucketName: The S3 bucket's name.
          \param key: The key for the object in the S3 bucket.
          \param client: An S3 client.
          \return bool: Function succeeded.
         */
        static bool
        deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
 &key,
                               Aws::S3::S3Client &client);
    }
}
```

```
//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
  \param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
  \param saveFilePath: Path for saving a downloaded S3 object.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &uploadFilePath,
                                           const Aws::String &saveFilePath,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: "doc-example-bucket-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String bucketName = "doc-example-bucket-" +
                             Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(

 Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                                clientConfig.region));
            request.WithCreateBucketConfiguration(createBucketConfiguration);
        }

        Aws::S3::Model::CreateBucketOutcome outcome =
 client.CreateBucket(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: createBucket: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
```

```
            return false;
        } else {
            std::cout << "Created the bucket, '" << bucketName <<
                      "', in the region, '" << clientConfig.region << "'." <<
std::endl;
        }
    }

    // 2. Upload a local file to the bucket.
    Aws::String key = "key-for-test";
    {
        Aws::S3::Model::PutObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        std::shared_ptr<Aws::FStream> input_data =
                Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                              uploadFilePath,
                                              std::ios_base::in |
                                              std::ios_base::binary);

        if (!input_data->is_open()) {
            std::cerr << "Error: unable to open file, '" << uploadFilePath <<
"'."
                      << std::endl;
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        }

        request.SetBody(input_data);

        Aws::S3::Model::PutObjectOutcome outcome =
                client.PutObject(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: putObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
            AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        } else {
            std::cout << "Added the object with the key, '" << key
                      << "', to the bucket, '"
                      << bucketName << "'." << std::endl;
```

```
        }
    }

    // 3. Download the object to a local file.
    {
        Aws::S3::Model::GetObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        Aws::S3::Model::GetObjectOutcome outcome =
                client.GetObject(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: getObject: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
        } else {
            std::cout << "Downloaded the object with the key, '" << key
                      << "', in the bucket, '"
                      << bucketName << "'." << std::endl;

            Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
                    GetBody();
            Aws::OFStream outStream(saveFilePath,
                                    std::ios_base::out | std::ios_base::binary);
            if (!outStream.is_open()) {
                std::cout << "Error: unable to open file, '" << saveFilePath <<
"'."
                          << std::endl;
            } else {
                outStream << ioStream.rdbuf();
                std::cout << "Wrote the downloaded object to the file '"
                          << saveFilePath << "'." << std::endl;
            }
        }
    }

    // 4. Copy the object to a different "folder" in the bucket.
    Aws::String copiedToKey = "test-folder/" + key;
    {
        Aws::S3::Model::CopyObjectRequest request;
        request.WithBucket(bucketName)
                .WithKey(copiedToKey)
```

```
                .WithCopySource(bucketName + "/" + key);

        Aws::S3::Model::CopyObjectOutcome outcome =
                client.CopyObject(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error: copyObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
        } else {
            std::cout << "Copied the object with the key, '" << key
                      << "', to the key, '" << copiedToKey
                      << ", in the bucket, '" << bucketName << "'." << std::endl;
        }
    }

    // 5. List objects in the bucket.
    {
        Aws::S3::Model::ListObjectsV2Request request;
        request.WithBucket(bucketName);

        Aws::String continuationToken;
        Aws::Vector<Aws::S3::Model::Object> allObjects;

        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
                    request);

            if (!outcome.IsSuccess()) {
                std::cerr << "Error: ListObjects: " <<
                          outcome.GetError().GetMessage() << std::endl;
                break;
            } else {
                Aws::Vector<Aws::S3::Model::Object> objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(),
objects.end());
                continuationToken = outcome.GetResult().GetContinuationToken();
            }
        } while (!continuationToken.empty());

        std::cout << allObjects.size() << " objects in the bucket, '" <<
bucketName
```

```
                  << "':" << std::endl;

        for (Aws::S3::Model::Object &object: allObjects) {
            std::cout << "     '" << object.GetKey() << "'" << std::endl;
        }
    }

    // 6. Delete all objects in the bucket.
    // All objects in the bucket must be deleted before deleting the bucket.
    AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
    AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

    // 7. Delete the bucket.
    return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                        const Aws::String &key,
                                        Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: deleteObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Deleted the object with the key, '" << key
                  << "', from the bucket, '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client
 &client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);
```

```
    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'." << std::endl;
    }
    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for C++ API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

定義包裝案例所使用之儲存貯體和物件動作的結構。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
```

```
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets() ([]types.Bucket, error) {
 result, err := basics.S3Client.ListBuckets(context.TODO(),
 &s3.ListBucketsInput{})
 var buckets []types.Bucket
 if err != nil {
  log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
 } else {
  buckets = result.Buckets
 }
 return buckets, err
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(bucketName string) (bool, error) {
 _, err := basics.S3Client.HeadBucket(context.TODO(), &s3.HeadBucketInput{
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
    log.Printf("Either you don't have access to bucket %v or another error
 occurred. "+
     "Here's what happened: %v\n", bucketName, err)
   }
```

```
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(name string, region string) error {
 _, err := basics.S3Client.CreateBucket(context.TODO(), &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  log.Printf("Couldn't create bucket %v in Region %v. Here's why: %v\n",
   name, region, err)
 }
 return err
}



// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(bucketName string, objectKey string,
 fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(context.TODO(), &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
    fileName, bucketName, objectKey, err)
  }
```

```
 }
 return err
}



// UploadLargeObject uses an upload manager to upload data to an object in a
 bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(bucketName string, objectKey string,
 largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }

 return err
}



// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(bucketName string, objectKey string,
 fileName string) error {
 result, err := basics.S3Client.GetObject(context.TODO(), &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
 objectKey, err)
  return err
 }
```

```
 defer result.Body.Close()
 file, err := os.Create(fileName)
 if err != nil {
  log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
  return err
 }
 defer file.Close()
 body, err := io.ReadAll(result.Body)
 if err != nil {
  log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
 err)
 }
 _, err = file.Write(body)
 return err
}



// DownloadLargeObject uses a download manager to download an object from a
 bucket.
// The download manager gets the data in parts and writes them to a buffer until
 all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(bucketName string, objectKey
 string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
 {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(context.TODO(), buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

```
// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(bucketName string, objectKey string,
 folderName string) error {
 _, err := basics.S3Client.CopyObject(context.TODO(), &s3.CopyObjectInput{
  Bucket:     aws.String(bucketName),
  CopySource: aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
  Key:        aws.String(fmt.Sprintf("%v/%v", folderName, objectKey)),
 })
 if err != nil {
  log.Printf("Couldn't copy object from %v:%v to %v:%v/%v. Here's why: %v\n",
   bucketName, objectKey, bucketName, folderName, objectKey, err)
 }
 return err
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(sourceBucket string, destinationBucket
 string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(context.TODO(), &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't copy object from %v:%v to %v:%v. Here's why: %v\n",
   sourceBucket, objectKey, destinationBucket, objectKey, err)
 }
 return err
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(bucketName string) ([]types.Object, error)
 {
 result, err := basics.S3Client.ListObjectsV2(context.TODO(),
 &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 })
 var contents []types.Object
 if err != nil {
```

```
  log.Printf("Couldn't list objects in bucket %v. Here's why: %v\n", bucketName,
 err)
 } else {
  contents = result.Contents
 }
 return contents, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(bucketName string, objectKeys []string)
 error {
 var objectIds []types.ObjectIdentifier
 for _, key := range objectKeys {
  objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
 }
 output, err := basics.S3Client.DeleteObjects(context.TODO(),
 &s3.DeleteObjectsInput{
  Bucket: aws.String(bucketName),
  Delete: &types.Delete{Objects: objectIds},
 })
 if err != nil {
  log.Printf("Couldn't delete objects from bucket %v. Here's why: %v\n",
 bucketName, err)
 } else {
  log.Printf("Deleted %v objects.\n", len(output.Deleted))
 }
 return err
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is
 returned.
func (basics BucketBasics) DeleteBucket(bucketName string) error {
 _, err := basics.S3Client.DeleteBucket(context.TODO(), &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
  log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
 }
 return err
}
```

執行互動式案例，說明如何使用 S3 儲存貯體和物件。

```
// RunGetStartedScenario is an interactive example that shows you how to use
 Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
 objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Upload a large object to the bucket by using an upload manager.
// 4. Download an object to a local file.
// 5. Download a large object by using a download manager.
// 6. Copy an object to a different folder in the bucket.
// 7. List objects in the bucket.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunGetStartedScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Println("Something went wrong with the demo.\n", r)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 getting started demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}

 count := 10
```

```
log.Printf("Let's list up to %v buckets for your account:", count)
buckets, err := bucketBasics.ListBuckets()
if err != nil {
 panic(err)
}
if len(buckets) == 0 {
 log.Println("You don't have any buckets!")
} else {
 if count > len(buckets) {
  count = len(buckets)
 }
 for _, bucket := range buckets[:count] {
  log.Printf("\t%v\n", *bucket.Name)
 }
}

bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
 demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(bucketName)
if err != nil {
 panic(err)
}
if !bucketExists {
 err = bucketBasics.CreateBucket(bucketName, sdkConfig.Region)
 if err != nil {
  panic(err)
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
const smallKey = "doc-example-key"
err = bucketBasics.UploadFile(bucketName, smallKey, smallFile)
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))
```

```
mibs := 30
log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your
bucket. ", mibs)
questioner.Ask("Press Enter when you're ready.")
largeBytes := make([]byte, 1024*1024*mibs)
rand.Seed(time.Now().Unix())
rand.Read(largeBytes)
largeKey := "doc-example-large"
log.Println("Uploading...")
err = bucketBasics.UploadLargeObject(bucketName, largeKey, largeBytes)
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download %v to a file.", smallKey)
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
demotools.NotEmpty{})
err = bucketBasics.DownloadFile(bucketName, smallKey, downloadFileName)
if err != nil {
 panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download the %v MiB object.", mibs)
questioner.Ask("Press Enter when you're ready.")
log.Println("Downloading...")
largeDownload, err := bucketBasics.DownloadLargeObject(bucketName, largeKey)
if err != nil {
 panic(err)
}
log.Printf("Downloaded %v bytes.", len(largeDownload))
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(bucketName, smallKey, folderName)
if err != nil {
 panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))
```

```
 log.Println("Let's list the objects in your bucket.")
 questioner.Ask("Press Enter when you're ready.")
 objects, err := bucketBasics.ListObjects(bucketName)
 if err != nil {
  panic(err)
 }
 log.Printf("Found %v objects.\n", len(objects))
 var objKeys []string
 for _, object := range objects {
  objKeys = append(objKeys, *object.Key)
  log.Printf("\t%v\n", *object.Key)
 }
 log.Println(strings.Repeat("-", 88))

 if questioner.AskBool("Do you want to delete your bucket and all of its "+
  "contents? (y/n)", "y") {
  log.Println("Deleting objects.")
  err = bucketBasics.DeleteObjects(bucketName, objKeys)
  if err != nil {
   panic(err)
  }
  log.Println("Deleting bucket.")
  err = bucketBasics.DeleteBucket(bucketName)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleting downloaded file %v.\n", downloadFileName)
  err = os.Remove(downloadFileName)
  if err != nil {
   panic(err)
  }
 } else {
  log.Println("Okay. Don't forget to delete objects from your bucket to avoid
 charges.")
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Go API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Deletes the object from the Amazon S3 bucket.
 * 8. Deletes the Amazon S3 bucket.
 */
```

```
public class S3Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <bucketName> <key> <objectPath> <savePath> <toBucket>

                Where:
                    bucketName - The Amazon S3 bucket to create.
                    key - The key to use.
                    objectPath - The path where the file is located (for example,
 C:/AWS/book2.pdf).
                    savePath - The path where the file is saved after it's
 downloaded (for example, C:/AWS/book2.pdf).
                    toBucket - An Amazon S3 bucket to where an object is copied
 to (for example, C:/AWS/book2.pdf).\s
                    """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an Amazon S3 bucket.");
        createBucket(s3, bucketName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update a local file to the Amazon S3 bucket.");
        uploadLocalFile(s3, bucketName, key, objectPath);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Download the object to another local file.");
        getObjectBytes(s3, bucketName, key, savePath);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Perform a multipart upload.");
        String multipartKey = "multiPartKey";
        multipartUpload(s3, toBucket, multipartKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List all objects located in the Amazon S3
bucket.");
        listAllObjects(s3, bucketName);
        anotherListExample(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Copy the object to another Amazon S3 bucket.");
        copyBucketObject(s3, bucketName, key, toBucket);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the object from the Amazon S3 bucket.");
        deleteObjectFromBucket(s3, bucketName, key);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Delete the Amazon S3 bucket.");
        deleteBucket(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon S3 operations were successfully
performed");
        System.out.println(DASHES);
```

```
        s3.close();
    }

    // Create a bucket by using a S3Waiter object.
    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteBucket(S3Client client, String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucket)
                .build();

        client.deleteBucket(deleteBucketRequest);
        System.out.println(bucket + " was deleted.");
    }

    /**
     * Upload an object in parts.
     */
    public static void multipartUpload(S3Client s3, String bucketName, String
key) {
        int mB = 1024 * 1024;
        // First create a multipart upload and get the upload id.
```

```
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object.
        UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(1).build();

        String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
                .eTag();
        CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
                .uploadId(uploadId)
                .partNumber(2).build();
        String etag2 = s3.uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
                .eTag();
        CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

        // Call completeMultipartUpload operation to tell S3 to merge all
uploaded
        // parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                .parts(part1, part2)
                .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
```

```
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .multipartUpload(completedMultipartUpload)
                .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        byte[] b = new byte[size];
        new Random().nextBytes(b);
        return ByteBuffer.wrap(b);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static void uploadLocalFile(S3Client s3, String bucketName, String
key, String objectPath) {
       PutObjectRequest objectRequest = PutObjectRequest.builder()
               .bucket(bucketName)
               .key(key)
               .build();

       s3.putObject(objectRequest, RequestBody.fromFile(new File(objectPath)));
   }

   public static void listAllObjects(S3Client s3, String bucketName) {
       ListObjectsV2Request listObjectsReqManual =
ListObjectsV2Request.builder()
               .bucket(bucketName)
               .maxKeys(1)
               .build();

       boolean done = false;
       while (!done) {
           ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
           for (S3Object content : listObjResponse.contents()) {
               System.out.println(content.key());
           }

           if (listObjResponse.nextContinuationToken() == null) {
               done = true;
           }

           listObjectsReqManual = listObjectsReqManual.toBuilder()
                   .continuationToken(listObjResponse.nextContinuationToken())
                   .build();
       }
   }

   public static void anotherListExample(S3Client s3, String bucketName) {
       ListObjectsV2Request listReq = ListObjectsV2Request.builder()
               .bucket(bucketName)
               .maxKeys(1)
               .build();

       ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

       // Process response pages.
```

```
        listRes.stream()
                .flatMap(r -> r.contents().stream())
                .forEach(content -> System.out.println(" Key: " + content.key() +
" size = " + content.size()));

        // Helper method to work with paginated collection of items directly.
        listRes.contents().stream()
                .forEach(content -> System.out.println(" Key: " + content.key() +
" size = " + content.size()));

        for (S3Object content : listRes.contents()) {
            System.out.println(" Key: " + content.key() + " size = " +
content.size());
        }
    }

    public static void deleteObjectFromBucket(S3Client s3, String bucketName,
String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        s3.deleteObject(deleteObjectRequest);
        System.out.println(key + " was deleted");
    }

    public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
        String encodedUrl = null;
        try {
            encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
StandardCharsets.UTF_8.toString());
        } catch (UnsupportedEncodingException e) {
            System.out.println("URL could not be encoded: " + e.getMessage());
        }
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
                .copySource(encodedUrl)
                .destinationBucket(toBucket)
                .destinationKey(objectKey)
                .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
```

```
            System.out.println("The " + objectKey + " was copied to " +
 toBucket);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

首先，匯入所有必要模組。

```
// Used to check if currently running file is this file.
import { fileURLToPath } from "url";
import { readdirSync, readFileSync, writeFileSync } from "fs";

// Local helper utils.
```

```
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

import {
  S3Client,
  CreateBucketCommand,
  PutObjectCommand,
  ListObjectsCommand,
  CopyObjectCommand,
  GetObjectCommand,
  DeleteObjectsCommand,
  DeleteBucketCommand,
} from "@aws-sdk/client-s3";
```

前面的匯入動作參考了一些協助公用程式。這些公用程式位於本節開頭所連結之 GitHub 儲存庫的本機公用程式。如需相關內容，請參閱下列公用程式的實作方式。

```
export const dirnameFromMetaUrl = (metaUrl) =>
  fileURLToPath(new URL(".", metaUrl));

import { select, input, confirm, checkbox } from "@inquirer/prompts";

export class Prompter {
  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}}
 options
   */
  select(options) {
    return select(options);
  }

  /**
   * @param {{ message: string }} options
   */
  input(options) {
    return input(options);
  }

  /**
   * @param {string} prompt
   */
```

```
  checkContinue = async (prompt = "") => {
    const prefix = prompt && prompt + " ";
    let ok = await this.confirm({
      message: `${prefix}Continue?`,
    });
    if (!ok) throw new Error("Exiting...");
  };

  /**
   * @param {{ message: string }} options
   */
  confirm(options) {
    return confirm(options);
  }

  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}}
 options
   */
  checkbox(options) {
    return checkbox(options);
  }
}

export const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n    ${text}\n${rule}\n`;
};
```

S3 的物件儲存在「儲存貯體」。接下來，來定義一個建立新儲存貯體的函數。

```
export const createBucket = async () => {
  const bucketName = await prompter.input({
    message: "Enter a bucket name. Bucket names must be globally unique:",
  });
  const command = new CreateBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log("Bucket created successfully.\n");
  return bucketName;
};
```

儲存貯體包含「物件」。此功能會將儲存庫的內容做為物件上傳至您的儲存貯體。

```
export const uploadFilesToBucket = async ({ bucketName, folderPath }) => {
  console.log(`Uploading files from ${folderPath}\n`);
  const keys = readdirSync(folderPath);
  const files = keys.map((key) => {
    const filePath = `${folderPath}/${key}`;
    const fileContent = readFileSync(filePath);
    return {
      Key: key,
      Body: fileContent,
    };
  });

  for (let file of files) {
    await s3Client.send(
      new PutObjectCommand({
        Bucket: bucketName,
        Body: file.Body,
        Key: file.Key,
      }),
    );
    console.log(`${file.Key} uploaded successfully.`);
  }
};
```

上傳物件之後，請檢查確認已正確上傳物件。您可以使 ListObjects 用的。您將使用「金鑰」屬性，但在回應中也有其他有用的屬性。

```
export const listFilesInBucket = async ({ bucketName }) => {
  const command = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(command);
  const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
  console.log("\nHere's a list of files in the bucket:");
  console.log(contentsList + "\n");
};
```

有時您可能想從儲存貯體複製物件到另一個儲存貯體。為此使用 CopyObject 命令。

```
export const copyFileFromBucket = async ({ destinationBucket }) => {
```

```
  const proceed = await prompter.confirm({
    message: "Would you like to copy an object from another bucket?",
  });

  if (!proceed) {
    return;
  } else {
    const copy = async () => {
      try {
        const sourceBucket = await prompter.input({
          message: "Enter source bucket name:",
        });
        const sourceKey = await prompter.input({
          message: "Enter source key:",
        });
        const destinationKey = await prompter.input({
          message: "Enter destination key:",
        });

        const command = new CopyObjectCommand({
          Bucket: destinationBucket,
          CopySource: `${sourceBucket}/${sourceKey}`,
          Key: destinationKey,
        });
        await s3Client.send(command);
        await copyFileFromBucket({ destinationBucket });
      } catch (err) {
        console.error(`Copy error.`);
        console.error(err);
        const retryAnswer = await prompter.confirm({ message: "Try again?" });
        if (retryAnswer) {
          await copy();
        }
      }
    };
    await copy();
  }
};
```

沒有從儲存貯體中取得多個物件的 SDK 方法。反之，您會建立一份待下載的物件清單，並重複執行這些物件。

```
export const downloadFilesFromBucket = async ({ bucketName }) => {
  const { Contents } = await s3Client.send(
    new ListObjectsCommand({ Bucket: bucketName }),
  );
  const path = await prompter.input({
    message: "Enter destination path for files:",
  });

  for (let content of Contents) {
    const obj = await s3Client.send(
      new GetObjectCommand({ Bucket: bucketName, Key: content.Key }),
    );
    writeFileSync(
      `${path}/${content.Key}`,
      await obj.Body.transformToByteArray(),
    );
  }
  console.log("Files downloaded successfully.\n");
};
```

該清除資源了。儲存貯體在刪除之前必須先清空。這兩個函數會清空並刪除儲存貯體。

```
export const emptyBucket = async ({ bucketName }) => {
  const listObjectsCommand = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(listObjectsCommand);
  const keys = Contents.map((c) => c.Key);

  const deleteObjectsCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: keys.map((key) => ({ Key: key })) },
  });
  await s3Client.send(deleteObjectsCommand);
  console.log(`${bucketName} emptied successfully.\n`);
};

export const deleteBucket = async ({ bucketName }) => {
  const command = new DeleteBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log(`${bucketName} deleted successfully.\n`);
};
```

「主要」函數會將所有內容放在一起。若你直接執行這個檔案，主要函數將受到叫用。

```
const main = async () => {
  const OBJECT_DIRECTORY = `${dirnameFromMetaUrl(
    import.meta.url,
  )}../../../../resources/sample_files/.sample_media`;

  try {
    console.log(wrapText("Welcome to the Amazon S3 getting started example."));
    console.log("Let's create a bucket.");
    const bucketName = await createBucket();
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("File upload."));
    console.log(
      "I have some default files ready to go. You can edit the source code to
 provide your own.",
    );
    await uploadFilesToBucket({
      bucketName,
      folderPath: OBJECT_DIRECTORY,
    });

    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Copy files."));
    await copyFileFromBucket({ destinationBucket: bucketName });
    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Download files."));
    await downloadFilesFromBucket({ bucketName });

    console.log(wrapText("Clean up."));
    await emptyBucket({ bucketName });
    await deleteBucket({ bucketName });
  } catch (err) {
    console.error(err);
  }
};
```

- 如需 API 詳細資訊，請參閱《AWS SDK for JavaScript API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <bucketName> <key> <objectPath> <savePath> <toBucket>

    Where:
        bucketName - The Amazon S3 bucket to create.
        key - The key to use.
        objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).
        savePath - The path where the file is saved after it's downloaded (for
 example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to (for
 example, C:/AWS/book2.pdf).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }
```

```
    val bucketName = args[0]
    val key = args[1]
    val objectPath = args[2]
    val savePath = args[3]
    val toBucket = args[4]

    // Create an Amazon S3 bucket.
    createBucket(bucketName)

    // Update a local file to the Amazon S3 bucket.
    putObject(bucketName, key, objectPath)

    // Download the object to another local file.
    getObjectFromMrap(bucketName, key, savePath)

    // List all objects located in the Amazon S3 bucket.
    listBucketObs(bucketName)

    // Copy the object to another Amazon S3 bucket
    copyBucketOb(bucketName, key, toBucket)

    // Delete the object from the Amazon S3 bucket.
    deleteBucketObs(bucketName, key)

    // Delete the Amazon S3 bucket.
    deleteBucket(bucketName)
    println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
```

```
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
```

```
        }

    S3Client { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }
```

```
    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS 適用於 Kotlin 的 SDK API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
        echo("\n");
        echo("---------------------------------------\n");
        print("Welcome to the Amazon S3 getting started demo using PHP!\n");
        echo("---------------------------------------\n");

        $region = 'us-west-2';

        $this->s3client = new S3Client([
                'region' => $region,
        ]);
        /* Inline declaration example
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
        */

        $this->bucketName = "doc-example-bucket-" . uniqid();

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
```

```
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with object copying before continuing.");
        }

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
```

```
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (count($check) <= 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

```
        echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- 如需 API 詳細資訊，請參閱《AWS SDK for PHP API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError


def do_scenario(s3_resource):
    print("-" * 88)
    print("Welcome to the Amazon S3 getting started demo!")
    print("-" * 88)
```

```
    bucket_name = f"doc-example-bucket-{uuid.uuid4()}"
    bucket = s3_resource.Bucket(bucket_name)
    try:
        bucket.create(
            CreateBucketConfiguration={
                "LocationConstraint": s3_resource.meta.client.meta.region_name
            }
        )
        print(f"Created demo bucket named {bucket.name}.")
    except ClientError as err:
        print(f"Tried and failed to create demo bucket {bucket_name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")
        print(f"\nCan't continue the demo without a bucket!")
        return

    file_name = None
    while file_name is None:
        file_name = input("\nEnter a file you want to upload to your bucket: ")
        if not os.path.exists(file_name):
            print(f"Couldn't find file {file_name}. Are you sure it exists?")
            file_name = None

    obj = bucket.Object(os.path.basename(file_name))
    try:
        obj.upload_file(file_name)
        print(
            f"Uploaded file {file_name} into bucket {bucket.name} with key
 {obj.key}."
        )
    except S3UploadFailedError as err:
        print(f"Couldn't upload file {file_name} to {bucket.name}.")
        print(f"\t{err}")

    answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
    if answer.lower() == "y":
        data = io.BytesIO()
        try:
            obj.download_fileobj(data)
            data.seek(0)
            print(f"Got your object. Here are the first 20 bytes:\n")
            print(f"\t{data.read(20)}")
        except ClientError as err:
            print(f"Couldn't download {obj.key}.")
```

```
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? "
    )
    if answer.lower() == "y":
        dest_obj = bucket.Object(f"demo-folder/{obj.key}")
        try:
            dest_obj.copy({"Bucket": bucket.name, "Key": obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("\nYour bucket contains the following objects:")
    try:
        for o in bucket.objects.all():
            print(f"\t{o.key}")
    except ClientError as err:
        print(f"Couldn't list the objects in bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")

    answer = input(
        "\nDo you want to delete all of the objects as well as the bucket (y/n)?
 "
    )
    if answer.lower() == "y":
        try:
            bucket.objects.delete()
            bucket.delete()
            print(f"Emptied and deleted bucket {bucket.name}.\n")
        except ClientError as err:
            print(f"Couldn't empty and delete bucket {bucket.name}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )
```

```
    print("Thanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    do_scenario(boto3.resource("s3"))
```

- 如需 API 的詳細資訊，請參閱《適用於 Python (Boto3) 的AWS SDK API 參考資料》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
require "aws-sdk-s3"

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
```

```
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "doc-example-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: "us-east-1" # Note: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts("Tried and failed to create demo bucket.")
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
    bucket
  end

  # Requests a file name from the user.
  #
  # @return The name of the file.
  def create_file
    File.open("demo.txt", w) { |f| f.write("This is a demo file.") }
  end

  # Uploads a file to an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
destination
  # @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded
file.
  def upload_file(bucket)
    File.open("demo.txt", "w+") { |f| f.write("This is a demo file.") }
    s3_object = bucket.object(File.basename("demo.txt"))
    s3_object.upload_file("demo.txt")
    puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't upload file demo.txt to #{bucket.name}.")
```

```
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    s3_object
  end

  # Downloads an Amazon S3 object to a file.
  #
  # @param s3_object [Aws::S3::Object] The object to download.
  def download_file(s3_object)
    puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
    answer = gets.chomp.downcase
    if answer == "y"
      puts("Enter a name for the downloaded file: ")
      file_name = gets.chomp
      s3_object.download_file(file_name)
      puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't download #{s3_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Copies an Amazon S3 object to a subfolder within the same bucket.
  #
  # @param source_object [Aws::S3::Object] The source object to copy.
  # @return [Aws::S3::Object, nil] The destination object.
  def copy_object(source_object)
    dest_object = nil
    puts("\nDo you want to copy #{source_object.key} to a subfolder in your
 bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == "y"
      dest_object = source_object.bucket.object("demo-folder/
#{source_object.key}")
      dest_object.copy_from(source_object)
      puts("Copied #{source_object.key} to #{dest_object.key}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't copy #{source_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
```

```
    dest_object
  end

  # Lists the objects in an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to query.
  def list_objects(bucket)
    puts("\nYour bucket contains the following objects:")
    bucket.objects.each do |obj|
      puts("\t#{obj.key}")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list the objects in bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
 ")
    answer = gets.chomp.downcase
    if answer == "y"
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the Amazon S3 getting started demo!")
  puts("-" * 88)

  bucket = scenario.create_bucket
  s3_object = scenario.upload_file(bucket)
```

```
  scenario.download_file(s3_object)
  scenario.copy_object(s3_object)
  scenario.list_objects(bucket)
  scenario.delete_bucket(bucket)

  puts("Thanks for watching!")
  puts("-" * 88)
rescue Aws::Errors::ServiceError
  puts("Something went wrong with the demo!")
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME
 == __FILE__
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Ruby API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

案例中執行的二進制文件的程式碼。

```
use aws_config::meta::region::RegionProviderChain;
```

```
use aws_sdk_s3::{config::Region, Client};
use s3_service::error::Error;
use uuid::Uuid;


#[tokio::main]
async fn main() -> Result<(), Error> {
    let (region, client, bucket_name, file_name, key, target_key) =
 initialize_variables().await;

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name,
 key, target_key).await
    {
        println!("{:?}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (Region, Client, String, String, String,
 String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);

    let bucket_name = format!("doc-example-bucket-{}", Uuid::new_v4());

    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    (region, client, bucket_name, file_name, key, target_key)
}

async fn run_s3_operations(
    region: Region,
    client: Client,
    bucket_name: String,
    file_name: String,
    key: String,
```

```
    target_key: String,
) -> Result<(), Error> {
    s3_service::create_bucket(&client, &bucket_name, region.as_ref()).await?;
    s3_service::upload_object(&client, &bucket_name, &file_name, &key).await?;
    let _object = s3_service::download_object(&client, &bucket_name, &key).await;
    s3_service::copy_object(&client, &bucket_name, &key, &target_key).await?;
    s3_service::list_objects(&client, &bucket_name).await?;
    s3_service::delete_objects(&client, &bucket_name).await?;
    s3_service::delete_bucket(&client, &bucket_name).await?;

    Ok(())
}
```

帶有常見動作的庫文件稱為二進制文件。

```
use aws_sdk_s3::operation::{
    copy_object::{CopyObjectError, CopyObjectOutput},
    create_bucket::{CreateBucketError, CreateBucketOutput},
    get_object::{GetObjectError, GetObjectOutput},
    list_objects_v2::ListObjectsV2Output,
    put_object::{PutObjectError, PutObjectOutput},
};
use aws_sdk_s3::types::{
    BucketLocationConstraint, CreateBucketConfiguration, Delete,
 ObjectIdentifier,
};
use aws_sdk_s3::{error::SdkError, primitives::ByteStream, Client};
use error::Error;
use std::path::Path;
use std::str;

pub mod error;

pub async fn delete_bucket(client: &Client, bucket_name: &str) -> Result<(),
 Error> {
    client.delete_bucket().bucket(bucket_name).send().await?;
    println!("Bucket deleted");
    Ok(())
}
```

```
pub async fn delete_objects(client: &Client, bucket_name: &str) ->
 Result<Vec<String>, Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    let mut delete_objects: Vec<ObjectIdentifier> = vec![];
    for obj in objects.contents() {
        let obj_id = ObjectIdentifier::builder()
            .set_key(Some(obj.key().unwrap().to_string()))
            .build()
            .map_err(Error::from)?;
        delete_objects.push(obj_id);
    }

    let return_keys = delete_objects.iter().map(|o| o.key.clone()).collect();

    if !delete_objects.is_empty() {
        client
            .delete_objects()
            .bucket(bucket_name)
            .delete(
                Delete::builder()
                    .set_objects(Some(delete_objects))
                    .build()
                    .map_err(Error::from)?,
            )
            .send()
            .await?;
    }

    let objects: ListObjectsV2Output =
 client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(Error::unhandled(
            "There were still objects left in the bucket.",
        )),
    }
}

pub async fn list_objects(client: &Client, bucket: &str) -> Result<(), Error> {
    let mut response = client
```

```
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}

pub async fn copy_object(
    client: &Client,
    bucket_name: &str,
    object_key: &str,
    target_key: &str,
) -> Result<CopyObjectOutput, SdkError<CopyObjectError>> {
    let mut source_bucket_and_object: String = "".to_owned();
    source_bucket_and_object.push_str(bucket_name);
    source_bucket_and_object.push('/');
    source_bucket_and_object.push_str(object_key);

    client
        .copy_object()
        .copy_source(source_bucket_and_object)
        .bucket(bucket_name)
        .key(target_key)
        .send()
        .await
}

pub async fn download_object(
    client: &Client,
```

```
    bucket_name: &str,
    key: &str,
) -> Result<GetObjectOutput, SdkError<GetObjectError>> {
    client
        .get_object()
        .bucket(bucket_name)
        .key(key)
        .send()
        .await
}

pub async fn upload_object(
    client: &Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<PutObjectOutput, SdkError<PutObjectError>> {
    let body = ByteStream::from_path(Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
}

pub async fn create_bucket(
    client: &Client,
    bucket_name: &str,
    region: &str,
) -> Result<CreateBucketOutput, SdkError<CreateBucketError>> {
    let constraint = BucketLocationConstraint::from(region);
    let cfg = CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await
}
```

- 如需 API 詳細資訊，請參閱《適用於 Rust 的AWS SDK API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

" Create an Amazon Simple Storage Service (Amazon S3) bucket. "
TRY.
    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
    ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
  CATCH /aws1/cx_s3_bucketalrdyexists.
    MESSAGE 'Bucket name already exists.' TYPE 'E'.
  CATCH /aws1/cx_s3_bktalrdyownedbyyou.
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.


"Upload an object to an S3 bucket."
```

```
    TRY.
        "Get contents of file from application server."
        DATA lv_file_content TYPE xstring.
        OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
        READ DATASET iv_key INTO lv_file_content.
        CLOSE DATASET iv_key.

        lo_s3->putobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_key
            iv_body = lv_file_content
        ).
        MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.

    " Get an object from a bucket. "
    TRY.
        DATA(lo_result) = lo_s3->getobject(
                    iv_bucket = iv_bucket_name
                    iv_key = iv_key
                ).
        DATA(lv_object_data) = lo_result->get_body( ).
        MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.

    " Copy an object to a subfolder in a bucket. "
    TRY.
        lo_s3->copyobject(
          iv_bucket = iv_bucket_name
          iv_key = |{ iv_copy_to_folder }/{ iv_key }|
          iv_copysource = |{ iv_bucket_name }/{ iv_key }|
        ).
        MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.
```

```
" List objects in the bucket. "
TRY.
    DATA(lo_list) = lo_s3->listobjects(
       iv_bucket = iv_bucket_name
     ).
    MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
DATA text TYPE string VALUE 'Object List - '.
DATA lv_object_key TYPE /aws1/s3_objectkey.
LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
  lv_object_key = lo_object->get_key( ).
  CONCATENATE lv_object_key ', ' INTO text.
ENDLOOP.
MESSAGE text TYPE'I'.

" Delete the objects in a bucket. "
TRY.
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
    ).
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }|
    ).
    MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.


" Delete the bucket. "
TRY.
    lo_s3->deletebucket(
        iv_bucket = iv_bucket_name
    ).
    MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的AWS SDK API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

一個用於處理對適用於 Swift 的 SDK 呼叫的 Swift 類別。

```
import Foundation
import AWSS3
import ClientRuntime
import AWSClientRuntime

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
    let client: S3Client
```

```
    /// Initialize and return a new ``ServiceHandler`` object, which is used to
drive the AWS calls
    /// used for the example.
    ///
    /// - Returns: A new ``ServiceHandler`` object, ready to be called to
    ///            execute AWS operations.
    public init() async {
        do {
            client = try S3Client(region: "us-east-2")
        } catch {
            print("ERROR: ", dump(error, name: "Initializing S3 client"))
            exit(1)
        }
    }

    /// Create a new user given the specified name.
    ///
    /// - Parameters:
    ///   - name: Name of the bucket to create.
    /// Throws an exception if an error occurs.
    public func createBucket(name: String) async throws {
        let config = S3ClientTypes.CreateBucketConfiguration(
            locationConstraint: .usEast2
        )
        let input = CreateBucketInput(
            bucket: name,
            createBucketConfiguration: config
        )
        _ = try await client.createBucket(input: input)
    }

    /// Delete a bucket.
    /// - Parameter name: Name of the bucket to delete.
    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
        _ = try await client.deleteBucket(input: input)
    }

    /// Upload a file from local storage to the bucket.
    /// - Parameters:
    ///   - bucket: Name of the bucket to upload the file to.
    ///   - key: Name of the file to create.
```

```
    ///   - file: Path name of the file to upload.
    public func uploadFile(bucket: String, key: String, file: String) async
throws {
        let fileUrl = URL(fileURLWithPath: file)
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.from(data: fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }

    /// Create a file in the specified bucket with the given name. The new
    /// file's contents are uploaded from a `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket to create a file in.
    ///   - key: Name of the file to create.
    ///   - data: A `Data` object to write into the new file.
    public func createFile(bucket: String, key: String, withData data: Data)
async throws {
        let dataStream = ByteStream.from(data: data)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }

    /// Download the named file to the given directory on the local device.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket that contains the file to be copied.
    ///   - key: The name of the file to copy from the bucket.
    ///   - to: The path of the directory on the local device where you want to
    ///     download the file.
    public func downloadFile(bucket: String, key: String, to: String) async
throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)
```

```
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the data stream object. Return immediately if there isn't one.
        guard let body = output.body,
              let data = try await body.readData() else {
            return
        }
        try data.write(to: fileUrl)
    }

    /// Read the specified file from the given S3 bucket into a Swift
    /// `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to read.
    ///   - key: Name of the file within the bucket to read.
    ///
    /// - Returns: A `Data` object containing the complete file data.
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the stream and return its contents in a `Data` object. If
        // there is no stream, return an empty `Data` object instead.
        guard let body = output.body,
              let data = try await body.readData() else {
            return "".data(using: .utf8)!
        }

        return data
    }

    /// Copy a file from one bucket to another.
    ///
    /// - Parameters:
    ///   - sourceBucket: Name of the bucket containing the source file.
```

```
    ///   - name: Name of the source file.
    ///   - destBucket: Name of the bucket to copy the file into.
    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        _ = try await client.copyObject(input: input)
    }

    /// Deletes the specified file from Amazon S3.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to delete.
    ///   - key: Name of the file to delete.
    ///
    public func deleteFile(bucket: String, key: String) async throws {
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.deleteObject(input: input)
        } catch {
            throw error
        }
    }

    /// Returns an array of strings, each naming one file in the
    /// specified bucket.
    ///
    /// - Parameter bucket: Name of the bucket to get a file listing for.
    /// - Returns: An array of `String` objects, each giving the name of
    ///            one file contained in the bucket.
    public func listBucketFiles(bucket: String) async throws -> [String] {
        let input = ListObjectsV2Input(
            bucket: bucket
        )
```

```
        let output = try await client.listObjectsV2(input: input)
        var names: [String] = []

        guard let objList = output.contents else {
            return []
        }

        for obj in objList {
            if let objName = obj.key {
                names.append(objName)
            }
        }

        return names
    }
}
```

一個管理 SDK 呼叫的 Swift 命令列程式。

```
import Foundation
import ServiceHandler
import ArgumentParser

/// The command-line arguments and options available for this
/// example command.
struct ExampleCommand: ParsableCommand {
    @Argument(help: "Name of the S3 bucket to create")
    var bucketName: String

    @Argument(help: "Pathname of the file to upload to the S3 bucket")
    var uploadSource: String

    @Argument(help: "The name (key) to give the file in the S3 bucket")
    var objName: String

    @Argument(help: "S3 bucket to copy the object to")
    var destBucket: String

    @Argument(help: "Directory where you want to download the file from the S3
 bucket")
    var downloadDir: String
```

```
    static var configuration = CommandConfiguration(
        commandName: "s3-basics",
        abstract: "Demonstrates a series of basic AWS S3 functions.",
        discussion: """
        Performs the following Amazon S3 commands:

        * `CreateBucket`
        * `PutObject`
        * `GetObject`
        * `CopyObject`
        * `ListObjects`
        * `DeleteObjects`
        * `DeleteBucket`
        """
    )

    /// Called by ``main()`` to do the actual running of the AWS
    /// example.
    func runAsync() async throws {
        let serviceHandler = await ServiceHandler()

        // 1. Create the bucket.
        print("Creating the bucket \(bucketName)...")
        try await serviceHandler.createBucket(name: bucketName)

        // 2. Upload a file to the bucket.
        print("Uploading the file \(uploadSource)...")
        try await serviceHandler.uploadFile(bucket: bucketName, key: objName,
file: uploadSource)

        // 3. Download the file.
        print("Downloading the file \(objName) to \(downloadDir)...")
        try await serviceHandler.downloadFile(bucket: bucketName, key: objName,
to: downloadDir)

        // 4. Copy the file to another bucket.
        print("Copying the file to the bucket \(destBucket)...")
        try await serviceHandler.copyFile(from: bucketName, name: objName, to:
destBucket)

        // 5. List the contents of the bucket.

        print("Getting a list of the files in the bucket \(bucketName)")
```

```
        let fileList = try await serviceHandler.listBucketFiles(bucket:
 bucketName)
        let numFiles = fileList.count
        if numFiles != 0 {
            print("\(numFiles) file\((numFiles > 1) ? "s" : "") in bucket
 \(bucketName):")
            for name in fileList {
                print("  \(name)")
            }
        } else {
            print("No files found in bucket \(bucketName)")
        }

        // 6. Delete the objects from the bucket.

        print("Deleting the file \(objName) from the bucket \(bucketName)...")
        try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
        print("Deleting the file \(objName) from the bucket \(destBucket)...")
        try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

        // 7. Delete the bucket.
        print("Deleting the bucket \(bucketName)...")
        try await serviceHandler.deleteBucket(name: bucketName)

        print("Done.")
    }
}

//
// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Swift 的AWS SDK API 參考》中的下列主題。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開 AWS 發套件開始使用 Amazon S3 物件的加密功能

下列程式碼範例示範如何開始使用 Amazon S3 物件的加密。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to apply client encryption to an object in an
/// Amazon Simple Storage Service (Amazon S3) bucket.
```

```
    /// </summary>
    public class SSEClientEncryption
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "exampleobject.txt";
            string copyTargetKeyName = "examplecopy.txt";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // Upload the object.
                PutObjectRequest putObjectRequest = await
UploadObjectAsync(client, bucketName, keyName, base64Key);

                // Download the object and verify that its contents match what
you uploaded.
                await DownloadObjectAsync(client, bucketName, keyName, base64Key,
putObjectRequest);

                // Get object metadata and verify that the object uses AES-256
encryption.
                await GetObjectMetadataAsync(client, bucketName, keyName,
base64Key);

                // Copy both the source and target objects using server-side
encryption with
                // an encryption key.
                await CopyObjectAsync(client, bucketName, keyName,
copyTargetKeyName, aesEncryption, base64Key);
            }
            catch (AmazonS3Exception ex)
```

```
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PutObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which
the
        /// object will be uploaded.</param>
        /// <param name="keyName">The name of the object to upload to the Amazon
S3
        /// bucket.</param>
        /// <param name="base64Key">The encryption key.</param>
        /// <returns>The PutObjectRequest object for use by
DownloadObjectAsync.</returns>
        public static async Task<PutObjectRequest> UploadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            PutObjectRequest putObjectRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };
            PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
            return putObjectRequest;
        }

        /// <summary>
        /// Downloads an encrypted object from an Amazon S3 bucket.
        /// </summary>
```

```
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// is located.</param>
        /// <param name="keyName">The name of the Amazon S3 object to download.</
param>
        /// <param name="base64Key">The encryption key used to encrypt the
        /// object.</param>
        /// <param name="putObjectRequest">The PutObjectRequest used to upload
        /// the object.</param>
        public static async Task DownloadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key,
            PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // Provide encryption information for the object stored in Amazon
 S3.
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            using (GetObjectResponse getResponse = await
 client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
 StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (string.Compare(putObjectRequest.ContentBody, content) == 0)
                {
                    Console.WriteLine("Object content is same as we uploaded");
                }
                else
                {
                    Console.WriteLine("Error...Object content is not same.");
```

```
                }

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
                {
                    Console.WriteLine("Object encryption method is AES256, same
as we set");
                }
                else
                {
                    Console.WriteLine("Error...Object encryption method is not
the same as AES256 we set");
                }
            }
        }

        /// <summary>
        /// Retrieves the metadata associated with an Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to call GetObjectMetadataAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket containing
the
        /// object for which we want to retrieve metadata.</param>
        /// <param name="keyName">The name of the object for which we wish to
        /// retrieve the metadata.</param>
        /// <param name="base64Key">The encryption key associated with the
        /// object.</param>
        public static async Task GetObjectMetadataAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
```

```
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
            Console.WriteLine("The object metadata show encryption method used
is: {0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }

        /// <summary>
        /// Copies an encrypted object from one Amazon S3 bucket to another.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// CopyObjectAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket containing the object
        /// to copy.</param>
        /// <param name="keyName">The name of the object to copy.</param>
        /// <param name="copyTargetKeyName">The Amazon S3 bucket to which the
object
        /// will be copied.</param>
        /// <param name="aesEncryption">The encryption type to use.</param>
        /// <param name="base64Key">The encryption key to use.</param>
        public static async Task CopyObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string copyTargetKeyName,
            Aes aesEncryption,
            string base64Key)
        {
            aesEncryption.GenerateKey();
            string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

            CopyObjectRequest copyRequest = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = keyName,
                DestinationBucket = bucketName,
                DestinationKey = copyTargetKeyName,

                // Information about the source object's encryption.
                CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
```

```
                CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,

                // Information about the target object's encryption.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
            };
            await client.CopyObjectAsync(copyRequest);
        }
    }
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。
  - CopyObject
  - GetObject
  - GetObjectMetadata

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開 AWS 發套件開始使用 Amazon S3 物件的標籤

下列程式碼範例示範如何開始使用 Amazon S3 物件的索引標籤。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
```

```
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to work with tags in Amazon Simple Storage
    /// Service (Amazon S3) objects.
    /// </summary>
    public class ObjectTag
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "newobject.txt";
            string filePath = @"*** file path ***";

            // Specify your bucket region (an example region is shown).
            RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

            var client = new AmazonS3Client(bucketRegion);
            await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
        }

        /// <summary>
        /// This method uploads an object with tags. It then shows the tag
        /// values, changes the tags, and shows the new tags.
        /// </summary>
        /// <param name="client">The Initialized Amazon S3 client object used
        /// to call the methods to create and change an objects tags.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object will be stored.</param>
        /// <param name="keyName">A string representing the key name of the
        /// object to be tagged.</param>
        /// <param name="filePath">The directory location and file name of the
        /// object to be uploaded to the Amazon S3 bucket.</param>
        public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
bucketName, string keyName, string filePath)
        {
            try
            {
                // Create an object with tags.
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    FilePath = filePath,
```

```
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Keyx1", Value = "Value1" },
                        new Tag { Key = "Keyx2", Value = "Value2" },
                    },
                };

                PutObjectResponse response = await
client.PutObjectAsync(putRequest);

                // Now retrieve the new object's tags.
                GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };

                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

                // Display the tag values.
                objectTags.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

                Tagging newTagSet = new Tagging()
                {
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Key3", Value = "Value3" },
                        new Tag { Key = "Key4", Value = "Value4" },
                    },
                };

                PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    Tagging = newTagSet,
                };
```

```
                PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

                // Retrieve the tags again and show the values.
                GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

                objectTags2.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine(
                        $"Error: '{ex.Message}'");
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetObjectTagging中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS 開發套件取得 Amazon S3 物件的合法保留組態

下列程式碼範例說明如何取得 S3 儲存貯體的合法保留組態。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
{bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetObjectLegalHold中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   // Get the legal hold details for an S3 object.
   public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
       try {
           GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
               .bucket(bucketName)
               .key(objectKey)
               .build();

           GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
           System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
               ":\n\tStatus: " + response.legalHold().status());
           return response.legalHold();

       } catch (S3Exception ex) {
           System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
       }

       return null;
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考GetObjectLegalHold中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import { fileURLToPath } from "url";
import { GetObjectLegalHoldCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new GetObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const response = await client.send(command);
    console.log(`Legal Hold Status: ${response.LegalHold.Status}`);
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "DOC-EXAMPLE-BUCKET", "OBJECT_KEY");
```

```
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考GetObjectLegalHold中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS 開發套件使用 Amazon S3 物件鎖定功能

下列程式碼範例顯示如何使用 S3 物件鎖定功能。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

執行展示 Amazon S3 物件鎖定功能的互動式案例。

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ObjectLockScenario;

public static class S3ObjectLockWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.
```

```
   This .NET example performs the following tasks:
       1. Create test Amazon Simple Storage Service (S3) buckets with different
lock policies.
       2. Upload sample objects to each bucket.
       3. Set some Legal Hold and Retention Periods on objects and buckets.
       4. Investigate lock policies by viewing settings or attempting to delete
or overwrite objects.
       5. Clean up objects and buckets.
  */

   public static S3ActionsWrapper _s3ActionsWrapper = null!;
   public static IConfiguration _configuration = null!;
   private static string _resourcePrefix = null!;
   private static string noLockBucketName = null!;
   private static string lockEnabledBucketName = null!;
   private static string retentionAfterCreationBucketName = null!;
   private static List<string> bucketNames = new List<string>();
   private static List<string> fileNames = new List<string>();

   public static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon service.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
               services.AddAWSService<IAmazonS3>()
                   .AddTransient<S3ActionsWrapper>()
           )
           .Build();

       _configuration = new ConfigurationBuilder()
           .SetBasePath(Directory.GetCurrentDirectory())
           .AddJsonFile("settings.json") // Load settings from .json file.
           .AddJsonFile("settings.local.json",
               true) // Optionally, load local settings.
           .Build();

       ConfigurationSetup();
```

```
        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Workflow Scenario.");
            Console.WriteLine(new string('-', 80));
            await Setup(true);

            await DemoActionChoices();

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Cleaning up resources.");
            Console.WriteLine(new string('-', 80));
            await Cleanup(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon S3 Object Locking Workflow is complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await Cleanup(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>
    /// Any setup operations needed.
    /// </summary>
    public static void ConfigurationSetup()
    {
```

```
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

        noLockBucketName = _resourcePrefix + "-no-lock";
        lockEnabledBucketName = _resourcePrefix + "-lock-enabled";
        retentionAfterCreationBucketName = _resourcePrefix + "-retention-after-
creation";

        bucketNames.Add(noLockBucketName);
        bucketNames.Add(lockEnabledBucketName);
        bucketNames.Add(retentionAfterCreationBucketName);
    }

    // <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Setup(bool interactive)
    {
        Console.WriteLine(
            "\nFor this workflow, we will use the AWS SDK for .NET to create
 several S3\n" +
            "buckets and files to demonstrate working with S3 locking features.
\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nS3 buckets can be created either with or without
 object lock enabled.");
        await _s3ActionsWrapper.CreateBucketWithObjectLock(noLockBucketName,
 false);
        await _s3ActionsWrapper.CreateBucketWithObjectLock(lockEnabledBucketName,
 true);
        await
 _s3ActionsWrapper.CreateBucketWithObjectLock(retentionAfterCreationBucketName,
 false);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();
```

```
        Console.WriteLine("\nA bucket can be configured to use object locking
with a default retention period.");
        await
_s3ActionsWrapper.ModifyBucketDefaultRetention(retentionAfterCreationBucketName,
true,
            ObjectLockRetentionMode.Governance, DateTime.UtcNow.AddDays(1));

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nObject lock policies can also be added to existing
buckets.");
        await _s3ActionsWrapper.EnableObjectLockOnBucket(lockEnabledBucketName);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        // Upload some files to the buckets.
        Console.WriteLine("\nNow let's add some test files:");
        var fileName = _configuration["exampleFileName"] ?? "exampleFile.txt";
        int fileCount = 2;
        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a bucket.");
        }

        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileCount; i++)
            {
                var numberedFileName = Path.GetFileNameWithoutExtension(fileName)
+ i + Path.GetExtension(fileName);
                fileNames.Add(numberedFileName);
                await _s3ActionsWrapper.UploadFileAsync(bucketName,
numberedFileName, fileName);
            }
        }
        Console.WriteLine("Press Enter to continue.");
        if (interactive)
```

```
            Console.ReadLine();

        if (!interactive)
            return true;
        Console.WriteLine("\nNow we can set some object lock policies on
individual files:");
        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileNames.Count; i++)
            {
                // No modifications to the objects in the first bucket.
                if (bucketName != bucketNames[0])
                {
                    var exampleFileName = fileNames[i];
                    switch (i)
                    {
                        case 0:
                            {
                                var question =
                                    $"\nWould you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a legal hold.
                                    await
_s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
ObjectLockLegalHoldStatus.On);

                                }
                                break;
                            }
                        case 1:
                            {
                                var question =
                                    $"\nWould you like to add a 1 day Governance
retention period to {exampleFileName} in {bucketName}? (y/n)" +
                                    "\nReminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a Governance mode retention period for
1 day.
```

```
                                    await
_s3ActionsWrapper.ModifyObjectRetentionPeriod(
                                        bucketName, exampleFileName,
                                        ObjectLockRetentionMode.Governance,
                                        DateTime.UtcNow.AddDays(1));
                                }
                                break;
                            }
                        }
                    }
                }
            }
            Console.WriteLine(new string('-', 80));
            return true;
        }

        // <summary>
        /// List all of the current buckets and objects.
        /// </summary>
        /// <param name="interactive">True to run as interactive.</param>
        /// <returns>The list of buckets and objects.</returns>
        public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)
        {
            var allObjects = new List<S3ObjectVersion>();
            foreach (var bucketName in bucketNames)
            {
                var objectsInBucket = await
_s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
                foreach (var objectKey in objectsInBucket.Versions)
                {
                    allObjects.Add(objectKey);
                }
            }

            if (interactive)
            {
                Console.WriteLine("\nCurrent buckets and objects:\n");
                int i = 0;
                foreach (var bucketObject in allObjects)
                {
                    i++;
                    Console.WriteLine(
```

```
                    $"{i}: {bucketObject.Key} \n\tBucket:
{bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
            }
        }

        return allObjects;
    }

    /// <summary>
    /// Present the user with the demo action choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<bool> DemoActionChoices()
    {
        var choices = new string[]{
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."};

        var choice = 0;
        // Keep asking the user until they choose to move on.
        while (choice != 6)
        {
            Console.WriteLine(new string('-', 80));
            choice = GetChoiceResponse(
                "\nExplore the S3 locking features by selecting one of the
following choices:"
                , choices);
            Console.WriteLine(new string('-', 80));
            switch (choice)
            {
                case 0:
                    {
                        await ListBucketsAndObjects(true);
                        break;
                    }
                case 1:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
```

```
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 2:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 3:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
overwrite:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        // Create the file if it does not already exist.
                        if (!File.Exists(allFiles[fileChoice].Key))
                        {
                            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
                            await sw.WriteLineAsync(
                                "This is a sample file for uploading to a
bucket.");
                        }
                        await
_s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
                        break;
                    }
                case 4:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
```

```
                        Console.WriteLine("\nEnter the number of the object and
bucket to view:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        await
_s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
                        break;
                    }
                case 5:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
view:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        break;
                    }
            }
        }
        return true;
    }

    // <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));

        if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
        {
            // Remove all locks and delete all buckets and objects.
            var allFiles = await ListBucketsAndObjects(false);
            foreach (var fileInfo in allFiles)
            {
```

```
                // Check for a legal hold.
                var legalHold = await
_s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
                if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
                {
                    await
_s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
ObjectLockLegalHoldStatus.Off);
                }

                // Check for a retention period.
                var retention = await
_s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);
                var hasRetentionPeriod = retention?.Mode ==
ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
DateTime.UtcNow.Date;
                await
_s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key,
hasRetentionPeriod, fileInfo.VersionId);
            }

            foreach (var bucketName in bucketNames)
            {
                await _s3ActionsWrapper.DeleteBucketByName(bucketName);
            }

        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
```

```
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {choices[i]}");
            }
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }
}
```

S3 函數的包裝類。

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
 bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
 enableObjectLock)
    {
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
 {enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
```

```
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };
```

```
            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
```

```
        {
            Console.WriteLine($"\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
```

```
                            Days = timeDifference.Days // Can be specified in
days or years but not both.
                        }
                    }
                }
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in
{bucketName}: " +
                              $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock retention:
'{ex.Message}'");
            return new ObjectLockRetention();
        }
    }

    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the legal hold details for an S3 object.
```

```
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
{bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }

    /// <summary>
    /// Get the object lock configuration details for an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to get details.</param>
    /// <returns>The bucket's object lock configuration details.</returns>
    public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
    {
        try
        {
            var request = new GetObjectLockConfigurationRequest()
            {
                BucketName = bucketName
            };
```

```
            var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                              $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                              $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

            return response.ObjectLockConfiguration;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
            return new ObjectLockConfiguration();
        }
    }

    /// <summary>
    /// Upload a file from the local computer to an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object to
upload.</param>
    /// <returns>True if success.<returns>
    public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
            ChecksumAlgorithm = ChecksumAlgorithm.SHA256
        };

        var response = await _amazonS3.PutObjectAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"\tSuccessfully uploaded {objectName} to
{bucketName}.");
            return true;
```

```
        }
        else
        {
            Console.WriteLine($"\tCould not upload {objectName} to
 {bucketName}.");
            return false;
        }
    }

    /// <summary>
    /// List bucket objects and versions.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <returns>The list of objects and versions.</returns>
    public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
 bucketName)
    {
        var request = new ListVersionsRequest()
        {
            BucketName = bucketName
        };

        var response = await _amazonS3.ListVersionsAsync(request);
        return response;
    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="hasRetention">True if the object has retention settings.</
param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
 objectKey, bool hasRetention, string? versionId = null)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
```

```
                VersionId = versionId,
            };
            if (hasRetention)
            {
                // Set the BypassGovernanceRetention header
                // if the file has retention settings.
                request.BypassGovernanceRetention = true;
            }
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine(
                $"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteBucketByName(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete bucket {bucketName}: " +
ex.Message);
            return false;
        }
```

```
    }

}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

執行展示 Amazon S3 物件鎖定功能的互動式案例。

```
// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
 questioner demotools.IQuestioner
 resources  Resources
 s3Actions  *actions.S3Actions
 sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner) ObjectLockScenario {
 scenario := ObjectLockScenario{
  questioner: questioner,
```

```
  resources:  Resources{},
  s3Actions:  &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
  sdkConfig:  sdkConfig,
 }
 scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
 scenario.resources.init(scenario.s3Actions, questioner)
 return scenario
}

type nameLocked struct {
 name   string
 locked bool
}

var createInfo = []nameLocked{
 {"standard-bucket", false},
 {"lock-bucket", true},
 {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
 log.Println("Let's create some S3 buckets to use for this workflow.")
 success := false
 for !success {
  prefix := scenario.questioner.Ask(
   "This example creates three buckets. Enter a prefix to name your buckets
 (remember bucket names must be globally unique):")

  for _, info := range createInfo {
   bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx,
 fmt.Sprintf("%s.%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
   if err != nil {
    switch err.(type) {
    case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
     log.Printf("Couldn't create bucket %s.\n", bucketName)
    default:
     panic(err)
    }
    break
   }
   scenario.resources.demoBuckets[info.name] = &DemoBucket{
    name:       bucketName,
    objectKeys: []string{},
```

```
    }
    log.Printf("Created bucket %s.\n", bucketName)
   }

   if len(scenario.resources.demoBuckets) < len(createInfo) {
    scenario.resources.deleteBuckets(ctx)
   } else {
    success = true
   }
  }

  log.Println("S3 buckets created.")
  log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
  log.Println("\nA bucket can be configured to use object locking.")
  scenario.questioner.Ask("Press Enter to continue.")

  var err error
  bucket := scenario.resources.demoBuckets["retention-bucket"]
  err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
   default:
    panic(err)
   }
  } else {
   log.Printf("Object locking enabled on bucket %s.", bucket.name)
  }

  log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a
  bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx
  context.Context) {
  log.Println("\nA bucket can be configured to use object locking with a default
  retention period.")
```

```
 bucket := scenario.resources.demoBuckets["retention-bucket"]
 retentionPeriod := scenario.questioner.AskInt("Enter the default retention
 period in days: ")
 err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
 bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
 types.ObjectLockRetentionModeGovernance)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't configure a default retention period on bucket %s.\n",
 bucket.name)
  default:
   panic(err)
  }
 } else {
  log.Printf("Default retention policy set on bucket %s with %d day retention
 period.", bucket.name, retentionPeriod)
  bucket.retentionEnabled = true
 }

 log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
 log.Println("Uploading test objects to S3 buckets.")

 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  for i := 0; i < 2; i++ {
   key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
 fmt.Sprintf("example-%d", i),
    fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
   if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
     log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
    default:
     panic(err)
    }
   } else {
    log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
    bucket.objectKeys = append(bucket.objectKeys, key)
   }
```

```
  }
 }

 scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
 log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test
 objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx
 context.Context) {
 log.Println("Now let's set object lock configurations on individual objects.")

 buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
 scenario.resources.demoBuckets["retention-bucket"]}
 for _, bucket := range buckets {
  for index, objKey := range bucket.objectKeys {
   switch index {
   case 0:
    if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold
 to %s in %s (y/n)? ", objKey, bucket.name), "y") {
     err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
 types.ObjectLockLegalHoldStatusOn)
     if err != nil {
      switch err.(type) {
      case *types.NoSuchKey:
       log.Printf("Couldn't set legal hold on %s.\n", objKey)
      default:
       panic(err)
      }
     } else {
      log.Printf("Legal hold set on %s.\n", objKey)
     }
    }
   case 1:
    q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to
 %s in %s?\n"+
     "Reminder: Only a user with the s3:BypassGovernanceRetention permission is
 able to delete this object\n"+
     "or its bucket until the retention period has expired. (y/n) ", objKey,
 bucket.name)
    if scenario.questioner.AskBool(q, "y") {
     err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
 types.ObjectLockRetentionModeGovernance, 1)
```

```
     if err != nil {
      switch err.(type) {
      case *types.NoSuchKey:
       log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
 bucket.name)
      default:
       panic(err)
      }
     } else {
      log.Printf("Retention period set to 1 for %s.", objKey)
      bucket.retentionEnabled = true
     }
    }
   }
  }
 }
 log.Println(strings.Repeat("-", 88))
}

const (
 ListAll = iota
 DeleteObject
 DeleteRetentionObject
 OverwriteObject
 ViewRetention
 ViewLegalHold
 Finish
)

// InteractWithObjects allows the user to interact with the objects and test the
 object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
 log.Println("Now you can interact with the objects to explore the object lock
 configurations.")
 interactiveChoices := []string{
  "List all objects and buckets.",
  "Attempt to delete an object.",
  "Attempt to delete an object with retention period bypass.",
  "Attempt to overwrite a file.",
  "View the retention settings for an object.",
  "View the legal hold settings for an object.",
  "Finish the workflow."}

 choice := ListAll
```

```
for choice != Finish {
 objList := scenario.GetAllObjects(ctx)
 objChoices := scenario.makeObjectChoiceList(objList)
 choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
interactiveChoices)
 switch choice {
 case ListAll:
  log.Println("The current objects in the example buckets are:")
  for _, objChoice := range objChoices {
   log.Println("\t", objChoice)
  }
 case DeleteObject, DeleteRetentionObject:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
delete:\n", objChoices)
  obj := objList[objChoice]
  deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
obj.versionId, choice == DeleteRetentionObject)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Println("Nothing to delete.")
   default:
    panic(err)
   }
  } else if deleted {
   log.Printf("Object %s deleted.\n", obj.key)
  }
 case OverwriteObject:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
  obj := objList[objChoice]
  _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Println("Couldn't upload to nonexistent bucket.")
   default:
    panic(err)
   }
  } else {
   log.Printf("Uploaded new content to object %s.\n", obj.key)
  }
 case ViewRetention:
```

```
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
  view:\n", objChoices)
    obj := objList[objChoice]
    retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket,
  obj.key)
    if err != nil {
     switch err.(type) {
     case *types.NoSuchKey:
      log.Printf("Can't get retention configuration for %s.\n", obj.key)
     default:
      panic(err)
     }
    } else if retention != nil {
     log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
  retention.Mode, retention.RetainUntilDate)
    } else {
     log.Printf("Object %s does not have object retention configured.\n", obj.key)
    }
   case ViewLegalHold:
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
  view:\n", objChoices)
    obj := objList[objChoice]
    legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket,
  obj.key, obj.versionId)
    if err != nil {
     switch err.(type) {
     case *types.NoSuchKey:
      log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
     default:
      panic(err)
     }
    } else if legalHold != nil {
     log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
    } else {
     log.Printf("Object %s does not have legal hold configured.", obj.key)
    }
   case Finish:
    log.Println("Let's clean up.")
   }
   log.Println(strings.Repeat("-", 88))
  }
}

type BucketKeyVersionId struct {
```

```
 bucket    string
 key       string
 versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns
 them in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
 []BucketKeyVersionId {
 var objectList []BucketKeyVersionId
 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Printf("Couldn't get object versions for %s.\n", bucket.name)
   default:
    panic(err)
   }
  } else {
   for _, version := range versions {
    objectList = append(objectList,
     BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
 *version.VersionId})
   }
  }
 }
 return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
 are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
 []BucketKeyVersionId) []string {
 choices := make([]string, len(bucketObjects))
 for i := 0; i < len(bucketObjects); i++ {
  choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
   bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
 }
 return choices
}
```

```
// Run runs the S3 Object Lock workflow scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := scenario.questioner.(*demotools.MockQuestioner)
   if isMock || scenario.questioner.AskBool("Do you want to see the full error
 message (y/n)?", "y") {
    log.Println(r)
   }
   scenario.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 Object Lock Workflow Scenario.")
 log.Println(strings.Repeat("-", 88))

 scenario.CreateBuckets(ctx)
 scenario.EnableLockOnBucket(ctx)
 scenario.SetDefaultRetentionPolicy(ctx)
 scenario.UploadTestObjects(ctx)
 scenario.SetObjectLockConfigurations(ctx)
 scenario.InteractWithObjects(ctx)

 scenario.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

定義包裝此範例中使用的 S3 動作的結構。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}
```

```
// CreateBucketWithLock creates a new S3 bucket with optional object locking
 enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
```

```
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}



// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }
```

```
 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
 "ObjectLockConfigurationNotFoundError" {
   log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
   err = nil
  }
 } else {
  lockConfig = output.ObjectLockConfiguration
 }

 return lockConfig, err
}



// GetObjectRetention retrieves the object retention configuration for an S3
 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
```

```
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
 {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}



// ModifyDefaultBucketRetention modifies the default retention period of an
 existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
```

```
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
 string) error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
   Status:    types.BucketVersioningStatusEnabled,
  },
 }
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
```

```
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}



// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
```

```
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
 string, contents string) (string, error) {
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
  }, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
 bucket)
  } else {
   outKey = *output.Key
  }
 }
 return outKey, err
}
```

```
// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
 string, versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
```

```
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
    }
   }
  }
 } else {
  deleted = true
 }
 return deleted, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
```

```
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
  } else if len(delOut.Errors) > 0 {
   for _, outErr := range delOut.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
   }
   err = fmt.Errorf("%s", *delOut.Errors[0].Message)
  }
 }
 return err
}
```

清理資源。

```
// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
 name             string
 legalHold        bool
 retentionEnabled bool
 objectKeys       []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario
 and handles
// cleanup when the scenario finishes.
type Resources struct {
 demoBuckets map[string]*DemoBucket

 s3Actions  *actions.S3Actions
 questioner demotools.IQuestioner
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
 demotools.IQuestioner) {
```

```
 resources.s3Actions = s3Actions
 resources.questioner = questioner
 resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 " +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if !wantDelete {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
  return
 }

 log.Println("Removing objects from S3 buckets and deleting buckets...")
 resources.deleteBuckets(ctx)
 //resources.deleteRetentionObjects(resources.retentionBucket,
 resources.retentionObjects)

 log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
 ObjectLockScenario.
func (resources *Resources) deleteBuckets(ctx context.Context) {
 for _, info := range createInfo {
  bucket := resources.demoBuckets[info.name]
  resources.deleteObjects(ctx, bucket)
  _, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
   Bucket: aws.String(bucket.name),
  })
  if err != nil {
   panic(err)
```

```
  }
 }
 resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket
 *DemoBucket) {
 lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx,
 bucket.name)
 if err != nil {
  panic(err)
 }
 versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("No objects to get from %s.\n", bucket.name)
  default:
   panic(err)
  }
 }
 delObjects := make([]types.ObjectIdentifier, len(versions))
 for i, version := range versions {
  if lockConfig != nil && lockConfig.ObjectLockEnabled ==
 types.ObjectLockEnabledEnabled {
   status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
 *version.Key, *version.VersionId)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Couldn't determine legal hold status for %s in %s.\n",
 *version.Key, bucket.name)
    default:
     panic(err)
    }
   } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
    err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
 *version.VersionId, types.ObjectLockLegalHoldStatusOff)
    if err != nil {
     switch err.(type) {
     case *types.NoSuchKey:
      log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
 bucket.name)
```

```
     default:
      panic(err)
     }
    }
   }
  }
  delObjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
 version.VersionId}
 }
 err = resources.s3Actions.DeleteObjects(ctx, bucket.name, delObjects,
 bucket.retentionEnabled)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Println("Nothing to delete.")
  default:
   panic(err)
  }
 }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Go API 參考》中的下列主題。
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

執行展示 Amazon S3 物件鎖定功能的互動式案例。

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
 Before running this Java V2 code example, set up your development
 environment, including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

 This Java example performs the following tasks:
    1. Create test Amazon Simple Storage Service (S3) buckets with different lock
 policies.
    2. Upload sample objects to each bucket.
    3. Set some Legal Hold and Retention Periods on objects and buckets.
    4. Investigate lock policies by viewing settings or attempting to delete or
 overwrite objects.
    5. Clean up objects and buckets.
 */
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        // Get the current date and time to ensure bucket name is unique.
        LocalDateTime currentTime = LocalDateTime.now();

        // Format the date and time as a string.
```

```
        DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyyMMddHHmmss");
        String timeStamp = currentTime.format(formatter);

        s3LockActions = new S3LockActions();
        bucketName = "bucket"+timeStamp;
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Workflow Scenario.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
```

```
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by
selecting one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
                }

                case 1 -> {
                    System.out.println("Enter the number of the object to
delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
                }

                case 2 -> {
```

```
                    System.out.println("Enter the number of the object to
delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
                }

                case 3 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();

                    // Attempt to overwrite the file.
                    try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
                        writer.write("This is a modified text.");

                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    s3LockActions.uploadFile(bucketName, objectKey, objectKey);
                }

                case 4 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
```

```
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectRetention(bucketName, objectKey);
                }

                case 5 -> {
                    System.out.println("Enter the number of the object to
view:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectLegalHold(bucketName, objectKey);
                    s3LockActions.getBucketObjectLockConfiguration(bucketName);
                }

                case 6 -> {
                    System.out.println("Exiting the workflow...");
                    return;
                }

                default -> {
                    System.out.println("Invalid choice. Please select again.");
                }
            }
        }
    }

    // Clean up the resources from the scenario.
    private static void cleanup() {
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
        for (S3InfoObject fileInfo : allFiles) {
            String bucketName = fileInfo.getBucketName();
            String key = fileInfo.getKeyName();
            String version = fileInfo.getVersion();
```

```
            if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
                ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
                if (legalHold != null) {
                    String holdStatus = legalHold.status().name();
                    System.out.println(holdStatus);
                    if (holdStatus.compareTo("ON") == 0) {
                        s3LockActions.modifyObjectLegalHold(bucketName, key,
false);
                    }
                }
                // Check for a retention period.
                ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
                boolean hasRetentionPeriod ;
                hasRetentionPeriod = retention != null;
                s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

            } else {
                System.out.println(bucketName +" objects do not have a legal
lock");
                s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
            }
        }

        // Delete the buckets.
        System.out.println("Delete "+bucketName);
        for (String bucket : bucketNames){
            s3LockActions.deleteBucketByName(bucket);
        }
    }

    private static void setup() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                For this workflow, we will use the AWS SDK for Java to create
several S3
                buckets and files to demonstrate working with S3 locking
features.
                """);
```

```
        System.out.println("S3 buckets can be created either with or without
object lock enabled.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();

        // Create three S3 buckets.
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
        s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Bucket "+bucketNames.get(2) +" will be configured to
use object locking with a default retention period.");
        s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
        s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        // Upload some files to the buckets.
        System.out.println("Now let's add some test files:");
        String fileName = "exampleFile.txt";
        int fileCount = 2;
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
            writer.write("This is a sample file for uploading to a bucket.");

        } catch (IOException e) {
            e.printStackTrace();
        }

        for (String bucketName : bucketNames){
            for (int i = 0; i < fileCount; i++) {
                // Get the file name without extension.
                String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
                int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
                if (extensionIndex > 0) {
```

```
                    fileNameWithoutExtension =
fileNameWithoutExtension.substring(0, extensionIndex);
                }

                // Create the numbered file names.
                String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
                fileNames.add(numberedFileName);
                s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
            }
        }

        String question = null;
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println("Now we can set some object lock policies on
individual files:");
        for (String bucketName : bucketNames) {
            for (int i = 0; i < fileNames.size(); i++){

                // No modifications to the objects in the first bucket.
                if (!bucketName.equals(bucketNames.get(0))) {
                    String exampleFileName = fileNames.get(i);
                    switch (i) {
                        case 0 -> {
                            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                            System.out.println(question);
                            String ans = scanner.nextLine().trim();
                            if (ans.equalsIgnoreCase("y")) {
                                System.out.println("**** You have selected to put
a legal hold " + exampleFileName);

                                // Set a legal hold.
                                s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
                            }
                        }
                        case 1 -> {
                            """
                                Would you like to add a 1 day Governance
retention period to %s in %s (y/n)?
```

```
                                Reminder: Only a user with the
 s3:BypassGovernanceRetention permission will be able to delete this file or its
 bucket until the retention period has expired.
                                """.formatted(exampleFileName, bucketName);
                            System.out.println(question);
                            String ans2 = scanner.nextLine().trim();
                            if (ans2.equalsIgnoreCase("y")) {

 s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                            }
                        }
                    }
                }
            }
        }

    // Get file extension.
    private static String getFileExtension(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        if (dotIndex > 0) {
            return fileName.substring(dotIndex);
        }
        return "";
    }

    public static void configurationSetup() {
        String noLockBucketName = bucketName + "-no-lock";
        String lockEnabledBucketName = bucketName + "-lock-enabled";
        String retentionAfterCreationBucketName = bucketName + "-retention-after-
creation";
        bucketNames.add(noLockBucketName);
        bucketNames.add(lockEnabledBucketName);
        bucketNames.add(retentionAfterCreationBucketName);
    }

    public static int getChoiceResponse(String question, String[] choices) {
        Scanner scanner = new Scanner(System.in);
        if (question != null) {
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }
```

```
        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid
 number.");
            }
        }

        return choiceNumber - 1;
    }
}
```

S3 函數的包裝類。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import
 software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
```

```
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import
 software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
 {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time
 zone.
        ZonedDateTime zonedDateTime =
 futureInstant.atZone(ZoneId.systemDefault());
```

```
        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
    }

    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
```

```
            System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
        }

        return null;
    }

    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
        return bucketNames.stream()
            .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
                .map(version -> {
                    S3InfoObject s3InfoObject = new S3InfoObject();
                    s3InfoObject.setBucketName(bucketName);
                    s3InfoObject.setVersion(version.versionId());
                    s3InfoObject.setKeyName(version.key());
                    return s3InfoObject;
                }))
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the
updated value.
                if (interactive) {
                    System.out.println(i + ": "+ s3InfoObject.getKeyName());
```

```
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
            .bucket(bucketName)
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();
```

```
       ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
           .objectLockEnabled(ObjectLockEnabled.ENABLED)
           .rule(lockRule)
           .build();

       PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
           .bucket(bucketName)
           .objectLockConfiguration(objectLockConfiguration)
           .build();


getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
       System.out.println("Added a default retention to bucket "+bucketName
+".");
   }

   // Enable object lock on an existing bucket.
   public void enableObjectLockOnBucket(String bucketName) {
       try {
           VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
               .status(BucketVersioningStatus.ENABLED)
               .build();

           PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
               .bucket(bucketName)
               .versioningConfiguration(versioningConfiguration)
               .build();

           // Enable versioning on the bucket.
           getClient().putBucketVersioning(putBucketVersioningRequest);
           PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
               .bucket(bucketName)
               .objectLockConfiguration(ObjectLockConfiguration.builder()
                   .objectLockEnabled(ObjectLockEnabled.ENABLED)
                   .build())
               .build();

           getClient().putObjectLockConfiguration(request);
```

```
            System.out.println("Successfully enabled object lock on
"+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
        }
    }

    public void uploadFile(String bucketName, String objectName, String filePath)
{
        Path file = Paths.get(filePath);
        PutObjectRequest request = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectName)
            .checksumAlgorithm(ChecksumAlgorithm.SHA256)
            .build();

        PutObjectResponse response = getClient().putObject(request, file);
        if (response != null) {
            System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
        } else {
            System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
        }
    }

    // Set or modify a legal hold on an object in an S3 bucket.
    public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
        ObjectLockLegalHold legalHold ;
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.OFF)
                .build();
        }

        PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
```

```
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
    }

    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey,
boolean hasRetention, String versionId) {
        try {
            DeleteObjectRequest objectRequest;
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
            System.out.println("The object was successfully deleted");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
```

```
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }

    public void deleteBucketByName(String bucketName) {
        try {
            DeleteBucketRequest request = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

            getClient().deleteBucket(request);
            System.out.println(bucketName +" was deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
"+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
response.objectLockConfiguration().rule().defaultRetention());
```

```
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

index.js-工作流程的入口點。這會協調所有的步驟。請造訪 GitHub 以查看 Scenario、ScenarioInput、 ScenarioOutput和的實作詳細資訊 ScenarioAction。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "./welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  confirmSetLegalHoldFileEnabled,
```

```
  confirmSetLegalHoldFileRetention,
  confirmSetRetentionPeriodFileEnabled,
  confirmSetRetentionPeriodFileRetention,
  confirmUpdateLockPolicy,
  confirmUpdateRetention,
  createBuckets,
  createBucketsAction,
  populateBuckets,
  populateBucketsAction,
  setLegalHoldFileEnabledAction,
  setLegalHoldFileRetentionAction,
  setRetentionPeriodFileEnabledAction,
  setRetentionPeriodFileRetentionAction,
  updateLockPolicy,
  updateLockPolicyAction,
  updateRetention,
  updateRetentionAction,
} from "./setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Object Locking - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        updateRetention(scenarios),
        confirmUpdateRetention(scenarios),
        exitOnFalse(scenarios, "confirmUpdateRetention"),
        updateRetentionAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
```

```
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
      ],
      initialState,
    ),
    demo: new scenarios.Scenario(
      "S3 Object Locking - Demo",
      [loadState, replAction(scenarios, client)],
      initialState,
    ),
    clean: new scenarios.Scenario(
      "S3 Object Locking - Destroy",
      [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
      ],
      initialState,
    ),
  };
};

// Call function if run directly
import { fileURLToPath } from "url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios);
```

```
}
```

welcome.steps.js-將歡迎消息輸出到控制台。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    `Welcome to the Amazon Simple Storage Service (S3) Object Locking Workflow
 Scenario. For this workflow, we will use the AWS SDK for JavaScript to create
 several S3 buckets and files to demonstrate working with S3 locking features.`,
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

setup.steps.js-部署存儲桶，對象和文件設置。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  BucketVersioningStatus,
  ChecksumAlgorithm,
```

```
  CreateBucketCommand,
  MFADeleteStatus,
  PutBucketVersioningCommand,
  PutObjectCommand,
  PutObjectLockConfigurationCommand,
  PutObjectLegalHoldCommand,
  PutObjectRetentionCommand,
  ObjectLockLegalHoldStatus,
  ObjectLockRetentionMode,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const bucketPrefix = "js-object-locking";

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    `The following buckets will be created:
         ${bucketPrefix}-no-lock with object lock False.
         ${bucketPrefix}-lock-enabled with object lock True.
         ${bucketPrefix}-retention-after-creation with object lock False.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
```

```
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const noLockBucketName = `${bucketPrefix}-no-lock`;
    const lockEnabledBucketName = `${bucketPrefix}-lock-enabled`;
    const retentionBucketName = `${bucketPrefix}-retention-after-creation`;

    await client.send(new CreateBucketCommand({ Bucket: noLockBucketName }));
    await client.send(
      new CreateBucketCommand({
        Bucket: lockEnabledBucketName,
        ObjectLockEnabledForBucket: true,
      }),
    );
    await client.send(new CreateBucketCommand({ Bucket: retentionBucketName }));

    state.noLockBucketName = noLockBucketName;
    state.lockEnabledBucketName = lockEnabledBucketName;
    state.retentionBucketName = retentionBucketName;
  });

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    `The following test files will be created:
         file0.txt in ${bucketPrefix}-no-lock.
         file1.txt in ${bucketPrefix}-no-lock.
         file0.txt in ${bucketPrefix}-lock-enabled.
         file1.txt in ${bucketPrefix}-lock-enabled.
         file0.txt in ${bucketPrefix}-retention-after-creation.
         file1.txt in ${bucketPrefix}-retention-after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
```

```
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    await client.send(
      new PutObjectCommand({
        Bucket: state.noLockBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.noLockBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.lockEnabledBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.lockEnabledBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
```

```
      new PutObjectCommand({
        Bucket: state.retentionBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.retentionBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 */
const updateRetention = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateRetention",
    `A bucket can be configured to use object locking with a default retention
 period.
   A default retention period will be configured for ${bucketPrefix}-retention-
after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateRetention",
    "Configure default retention period?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
```

```
const updateRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateRetentionAction", async (state) => {
    await client.send(
      new PutBucketVersioningCommand({
        Bucket: state.retentionBucketName,
        VersioningConfiguration: {
          MFADelete: MFADeleteStatus.Disabled,
          Status: BucketVersioningStatus.Enabled,
        },
      }),
    );

    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.retentionBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
          Rule: {
            DefaultRetention: {
              Mode: "GOVERNANCE",
              Years: 1,
            },
          },
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 */
const updateLockPolicy = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateLockPolicy",
    `Object lock policies can also be added to existing buckets.
   An object lock policy will be added to ${bucketPrefix}-lock-enabled.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateLockPolicy = (scenarios) =>
  new scenarios.ScenarioInput(
```

```
    "confirmUpdateLockPolicy",
    "Add object lock policy?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateLockPolicyAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateLockPolicyAction", async (state) => {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.lockEnabledBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileEnabled",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.lockEnabledBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileEnabledAction",
    async (state) => {
```

```
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.lockEnabledBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetRetentionPeriodFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileEnabled",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.lockEnabledBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
 able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileEnabledAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
```

```
        new PutObjectRetentionCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.lockEnabledBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileRetention",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.retentionBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileRetentionAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
```

```
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.retentionBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileRetention },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmSetRetentionPeriodFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileRetention",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.retentionBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
 able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileRetentionAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
```

```
            RetainUntilDate: retentionDate,
          },
          BypassGovernanceRetention: true,
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.retentionBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileRetention },
  );

export {
  createBuckets,
  confirmCreateBuckets,
  createBucketsAction,
  populateBuckets,
  confirmPopulateBuckets,
  populateBucketsAction,
  updateRetention,
  confirmUpdateRetention,
  updateRetentionAction,
  updateLockPolicy,
  confirmUpdateLockPolicy,
  updateLockPolicyAction,
  confirmSetLegalHoldFileEnabled,
  setLegalHoldFileEnabledAction,
  confirmSetRetentionPeriodFileEnabled,
  setRetentionPeriodFileEnabledAction,
  confirmSetLegalHoldFileRetention,
  setLegalHoldFileRetentionAction,
  confirmSetRetentionPeriodFileRetention,
  setRetentionPeriodFileRetentionAction,
};
```

repl.steps.js-查看和刪除存儲桶中的文件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  ChecksumAlgorithm,
```

```
  DeleteObjectCommand,
  GetObjectLegalHoldCommand,
  GetObjectLockConfigurationCommand,
  GetObjectRetentionCommand,
  ListObjectVersionsCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  DELETE_FILE: 2,
  DELETE_FILE_WITH_RETENTION: 3,
  OVERWRITE_FILE: 4,
  VIEW_RETENTION_SETTINGS: 5,
  VIEW_LEGAL_HOLD_SETTINGS: 6,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new scenarios.ScenarioInput(
    "replChoice",
    `Explore the S3 locking features by selecting one of the following choices`,
    {
      type: "select",
      choices: [
        { name: "List all files in buckets", value: choices.LIST_ALL_FILES },
        { name: "Attempt to delete a file.", value: choices.DELETE_FILE },
        {
          name: "Attempt to delete a file with retention period bypass.",
          value: choices.DELETE_FILE_WITH_RETENTION,
        },
        { name: "Attempt to overwrite a file.", value: choices.OVERWRITE_FILE },
        {
```

```
          name: "View the object and bucket retention settings for a file.",
          value: choices.VIEW_RETENTION_SETTINGS,
        },
        {
          name: "View the legal hold settings for a file.",
          value: choices.VIEW_LEGAL_HOLD_SETTINGS,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key, VersionId } = version;
      files.push({ bucket, key: Key, version: VersionId });
    }
  }

  return files;
};

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const replAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.noLockBucketName,
        state.lockEnabledBucketName,
        state.retentionBucketName,
```

```
  ]);

  const fileInput = new scenarios.ScenarioInput(
    "selectedFile",
    "Select a file:",
    {
      type: "select",
      choices: files.map((file, index) => ({
        name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${
          file.version
        })`,
        value: index,
      })),
    },
  );

  const { replChoice } = state;

  switch (replChoice) {
    case choices.LIST_ALL_FILES: {
      const files = await getAllFiles(client, [
        state.noLockBucketName,
        state.lockEnabledBucketName,
        state.retentionBucketName,
      ]);
      state.replOutput = files
        .map(
          (file) =>
            `${file.bucket}: ${file.key} (version: ${file.version})`,
        )
        .join("\n");
      break;
    }
    case choices.DELETE_FILE: {
      /** @type {number} */
      const fileToDelete = await fileInput.handle(state);
      const selectedFile = files[fileToDelete];
      try {
        await client.send(
          new DeleteObjectCommand({
            Bucket: selectedFile.bucket,
            Key: selectedFile.key,
            VersionId: selectedFile.version,
          }),
```

```
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.DELETE_FILE_WITH_RETENTION: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
                BypassGovernanceRetention: true,
              }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.OVERWRITE_FILE: {
          /** @type {number} */
          const fileToOverwrite = await fileInput.handle(state);
          const selectedFile = files[fileToOverwrite];
          try {
            await client.send(
              new PutObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                Body: "New content",
                ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
              }),
            );
```

```
            state.replOutput = `Overwrote ${selectedFile.key} in
 ${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to overwrite object ${selectedFile.key} in
 bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.VIEW_RETENTION_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const retention = await client.send(
              new GetObjectRetentionCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            const bucketConfig = await client.send(
              new GetObjectLockConfigurationCommand({
                Bucket: selectedFile.bucket,
              }),
            );
            state.replOutput = `Object retention for ${selectedFile.key}
 in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
 ${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
 ${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
          } catch (err) {
            state.replOutput = `Unable to fetch object lock retention:
 '${err.message}'`;
          }
          break;
        }
        case choices.VIEW_LEGAL_HOLD_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const legalHold = await client.send(
```

```
              new GetObjectLegalHoldCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Object legal hold for ${selectedFile.key} in
 ${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
          } catch (err) {
            state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
          }
          break;
        }
        default:
          throw new Error(`Invalid replChoice: ${replChoice}`);
      }
    },
    {
      whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new scenarios.ScenarioOutput(
          "REPL output",
          (state) => state.replOutput,
          { preformatted: true },
        ),
      },
    },
  );

export { replInput, replAction, choices };
```

clean.steps.js-銷毀所有創建的資源。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
  GetObjectLegalHoldCommand,
  GetObjectRetentionCommand,
```

```
  PutObjectLegalHoldCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { noLockBucketName, lockEnabledBucketName, retentionBucketName } =
      state;

    const buckets = [
      noLockBucketName,
      lockEnabledBucketName,
      retentionBucketName,
    ];

    for (const bucket of buckets) {
      /** @type {import("@aws-sdk/client-s3").ListObjectVersionsCommandOutput} */
      let objectsResponse;

      try {
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
```

```
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Object's bucket has already been deleted.");
          continue;
        } else {
          throw e;
        }
      }

      for (const version of objectsResponse.Versions || []) {
        const { Key, VersionId } = version;

        try {
          const legalHold = await client.send(
            new GetObjectLegalHoldCommand({
              Bucket: bucket,
              Key,
              VersionId,
            }),
          );

          if (legalHold.LegalHold?.Status === "ON") {
            await client.send(
              new PutObjectLegalHoldCommand({
                Bucket: bucket,
                Key,
                VersionId,
                LegalHold: {
                  Status: "OFF",
                },
              }),
            );
          }
        } catch (err) {
          console.log(
            `Unable to fetch legal hold for ${Key} in ${bucket}:
'${err.message}'`,
          );
        }

        try {
          const retention = await client.send(
            new GetObjectRetentionCommand({
              Bucket: bucket,
```

```
              Key,
              VersionId,
            }),
          );

          if (retention.Retention?.Mode === "GOVERNANCE") {
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
                BypassGovernanceRetention: true,
              }),
            );
          }
        } catch (err) {
          console.log(
            `Unable to fetch object lock retention for ${Key} in ${bucket}:
 '${err.message}'`,
          );
        }

        await client.send(
          new DeleteObjectCommand({
            Bucket: bucket,
            Key,
            VersionId,
          }),
        );
      }

      await client.send(new DeleteBucketCommand({ Bucket: bucket }));
      console.log(`Delete for ${bucket} complete.`);
    }
  });

export { confirmCleanup, cleanupAction };
```

- 如需 API 詳細資訊，請參閱《AWS SDK for JavaScript API 參考》中的下列主題。
  - GetObjectLegalHold
  - GetObjectLockConfiguration

- GetObjectRetention
- PutObjectLegalHold
- PutObjectLockConfiguration
- PutObjectRetention

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件 AWS 管理 Amazon S3 儲存貯體的存取控制清單 (ACL)

下列程式碼範例示範如何管理 Amazon S3 儲存貯體的存取控制清單 (ACL)。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to manage Amazon Simple Storage Service
/// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
/// access.
/// </summary>
public class ManageACLs
{
    public static async Task Main()
    {
        string bucketName = "doc-example-bucket1";
        string newBucketName = "doc-example-bucket2";
        string keyName = "sample-object.txt";
```

```
            string emailAddress = "someone@example.com";

            // If the AWS Region where your bucket is located is different from
            // the Region defined for the default user, pass the Amazon S3
bucket's
            // name to the client constructor. It should look like this:
            // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
            IAmazonS3 client = new AmazonS3Client();

            await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the
ACL
        /// information and then adds a new ACL to one of the objects in the
        /// Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// methods to create a bucket, get an ACL, and add a different ACL to
        /// one of the objects.</param>
        /// <param name="bucketName">A string representing the original Amazon S3
        /// bucket name.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new bucket that will be created.</param>
        /// <param name="keyName">A string representing the key name of an Amazon
S3
        /// object for which we will change the ACL.</param>
        /// <param name="emailAddress">A string representing the email address
        /// belonging to the person to whom access to the Amazon S3 bucket will
be
        /// granted.</param>
        public static async Task TestBucketObjectACLsAsync(
            IAmazonS3 client,
            string bucketName,
            string newBucketName,
            string keyName,
            string emailAddress)
        {
            try
            {
                // Create a new Amazon S3 bucket and specify canned ACL.
```

```
                var success = await CreateBucketWithCannedACLAsync(client,
 newBucketName);

                // Get the ACL on a bucket.
                await GetBucketACLAsync(client, bucketName);

                // Add (replace) the ACL on an object in a bucket.
                await AddACLToExistingObjectAsync(client, bucketName, keyName,
 emailAddress);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Exception: {amazonS3Exception.Message}");
            }
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL attached.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new Amazon S3 bucket.</param>
        /// <returns>Returns a boolean value indicating success or failure.</
returns>
        public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
 client, string newBucketName)
        {
            var request = new PutBucketRequest()
            {
                BucketName = newBucketName,
                BucketRegion = S3Region.EUWest1,

                // Add a canned ACL.
                CannedACL = S3CannedACL.LogDeliveryWrite,
            };

            var response = await client.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }


        /// <summary>
        /// Retrieves the ACL associated with the Amazon S3 bucket name in the
```

```
        /// bucketName parameter.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket for which we want to
 get the
        /// ACL list.</param>
        /// <returns>Returns an S3AccessControlList returned from the call to
        /// GetACLAsync.</returns>
        public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
 client, string bucketName)
        {
            GetACLResponse response = await client.GetACLAsync(new GetACLRequest
            {
                BucketName = bucketName,
            });

            return response.AccessControlList;
        }



        /// <summary>
        /// Adds a new ACL to an existing object in the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
 S3
        /// bucket containing the object to which we want to apply a new ACL.</
param>
        /// <param name="keyName">A string representing the name of the object
        /// to which we want to apply the new ACL.</param>
        /// <param name="emailAddress">The email address of the person to whom
        /// we will be applying to whom access will be granted.</param>
        public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
 string bucketName, string keyName, string emailAddress)
        {
            // Retrieve the ACL for an object.
            GetACLResponse aclResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
```

```
            });

            S3AccessControlList acl = aclResponse.AccessControlList;

            // Retrieve the owner.
            Owner owner = acl.Owner;

            // Clear existing grants.
            acl.Grants.Clear();

            // Add a grant to reset the owner's full permission
            // (the previous clear statement removed all permissions).
            var fullControlGrant = new S3Grant
            {
                Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
            };
            acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

            // Specify email to identify grantee for granting permissions.
            var grantUsingEmail = new S3Grant
            {
                Grantee = new S3Grantee { EmailAddress = emailAddress },
                Permission = S3Permission.WRITE_ACP,
            };

            // Specify log delivery group as grantee.
            var grantLogDeliveryGroup = new S3Grant
            {
                Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/
s3/LogDelivery" },
                Permission = S3Permission.WRITE,
            };

            // Create a new ACL.
            var newAcl = new S3AccessControlList
            {
                Grants = new List<S3Grant> { grantUsingEmail,
 grantLogDeliveryGroup },
                Owner = owner,
            };

            // Set the new ACL. We're throwing away the response here.
            _ = await client.PutACLAsync(new PutACLRequest
            {
```

```
                BucketName = bucketName,
                Key = keyName,
                AccessControlList = newAcl,
            });
        }

    }
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。
  - GetBucketAcl
  - GetObjectAcl
  - PutBucketAcl
  - PutObjectAcl

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件使用 Lambda 函數批次管理版本控制的 Amazon S3 物件 AWS

下列程式碼範例示範如何使用 Lambda 函數批次管理 S3 物件版本。

Python

適用於 Python (Boto3) 的 SDK

示範如何透過建立呼叫 AWS Lambda 函數來執行處理的任務，以批次處理 Amazon 簡單儲存服務 (Amazon S3) 版本控制的物件。此範例建立了一個啟用版本控制的儲存貯體，上傳一段 Lewis Carroll You Are Old, Father William 詩句，並使用 Amazon S3 Batch Job 以不同的方式扭曲詩句。

了解如何：

- 建立版本化物件上執行的 Lambda 函數。
- 建立更新物件的資訊清單。
- 建立叫用 Lambda 函數的批次任務以更新物件。
- 刪除 Lambda 函數。
- 清空並刪除版本化儲存貯體。

此範例最佳檢視於 GitHub。有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon S3

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件 AWS 剖析 Amazon S3 URI

下列程式碼範例示範如何剖析 Amazon S3 URI，以擷取如儲存貯體名稱和物件金鑰的重要元件。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

透過使用 S3Uri 類別剖析 Amazon S3 URI。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

    /**
     *
     * @param s3Client    - An S3Client through which you acquire an S3Uri
 instance.
```

```
     * @param s3ObjectUrl - A complex URL (String) that is used to demonstrate
 S3Uri
     *                     capabilities.
     */
    public static void parseS3UriExample(S3Client s3Client, String s3ObjectUrl) {
        logger.info(s3ObjectUrl);
        // Console output:
        // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

        // Create an S3Utilities object using the configuration of the s3Client.
        S3Utilities s3Utilities = s3Client.utilities();

        // From a String URL create a URI object to pass to the parseUri()
 method.
        URI uri = URI.create(s3ObjectUrl);
        S3Uri s3Uri = s3Utilities.parseUri(uri);

        // If the URI contains no value for the Region, bucket or key, the SDK
 returns
        // an empty Optional.
        // The SDK returns decoded URI values.

        Region region = s3Uri.region().orElse(null);
        log("region", region);
        // Console output: 'region: us-west-1'.

        String bucket = s3Uri.bucket().orElse(null);
        log("bucket", bucket);
        // Console output: 'bucket: myBucket'.

        String key = s3Uri.key().orElse(null);
        log("key", key);
        // Console output: 'key: resources/doc.txt'.

        Boolean isPathStyle = s3Uri.isPathStyle();
        log("isPathStyle", isPathStyle);
        // Console output: 'isPathStyle: true'.

        // If the URI contains no query parameters, the SDK returns an empty map.
        Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
        log("rawQueryParameters", queryParams);
        // Console output: 'rawQueryParameters: {versionId=[abc123],
 partNumber=[77,
```

```
        // 88]}'.

        // Retrieve the first or all values for a query parameter as shown in the
        // following code.
        String versionId =
 s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
        log("firstMatchingRawQueryParameter-versionId", versionId);
        // Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

        String partNumber =
 s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
        log("firstMatchingRawQueryParameter-partNumber", partNumber);
        // Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

        List<String> partNumbers =
 s3Uri.firstMatchingRawQueryParameters("partNumber");
        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /*
         * Object keys and query parameters with reserved or unsafe characters,
 must be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?
query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".",
 the dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
        } else {
            logger.info("{}: {}", s3UriElement, element);
```

```
        }
    }
```

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件執行 Amazon S3 物件的多部分複本 AWS

下列程式碼範例示範如何執行 Amazon S3 物件的分段複製。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to perform a multi-part copy from one Amazon
/// Simple Storage Service (Amazon S3) bucket to another.
/// </summary>
public class MPUapiCopyObj
{
    private const string SourceBucket = "doc-example-bucket1";
    private const string TargetBucket = "doc-example-bucket2";
    private const string SourceObjectKey = "example.mov";
    private const string TargetObjectKey = "copied_video_file.mov";

    /// <summary>
    /// This method starts the multi-part upload.
    /// </summary>
```

```
        public static async Task Main()
        {
            var s3Client = new AmazonS3Client();
            Console.WriteLine("Copying object...");
            await MPUCopyObjectAsync(s3Client);
        }

        /// <summary>
        /// This method uses the passed client object to perform a multipart
        /// copy operation.
        /// </summary>
        /// <param name="client">An Amazon S3 client object that will be used
        /// to perform the copy.</param>
        public static async Task MPUCopyObjectAsync(AmazonS3Client client)
        {
            // Create a list to store the copy part responses.
            var copyResponses = new List<CopyPartResponse>();

            // Setup information required to initiate the multipart upload.
            var initiateRequest = new InitiateMultipartUploadRequest
            {
                BucketName = TargetBucket,
                Key = TargetObjectKey,
            };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await client.InitiateMultipartUploadAsync(initiateRequest);

            // Save the upload ID.
            string uploadId = initResponse.UploadId;

            try
            {
                // Get the size of the object.
                var metadataRequest = new GetObjectMetadataRequest
                {
                    BucketName = SourceBucket,
                    Key = SourceObjectKey,
                };

                GetObjectMetadataResponse metadataResponse =
                    await client.GetObjectMetadataAsync(metadataRequest);
```

```
                var objectSize = metadataResponse.ContentLength; // Length in
bytes.

                // Copy the parts.
                var partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
                for (int i = 1; bytePosition < objectSize; i++)
                {
                    var copyRequest = new CopyPartRequest
                    {
                        DestinationBucket = TargetBucket,
                        DestinationKey = TargetObjectKey,
                        SourceBucket = SourceBucket,
                        SourceKey = SourceObjectKey,
                        UploadId = uploadId,
                        FirstByte = bytePosition,
                        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
                        PartNumber = i,
                    };

                    copyResponses.Add(await client.CopyPartAsync(copyRequest));

                    bytePosition += partSize;
                }

                // Set up to complete the copy.
                var completeRequest = new CompleteMultipartUploadRequest
                {
                    BucketName = TargetBucket,
                    Key = TargetObjectKey,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(copyResponses);

                // Complete the copy.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error encountered on server.
Message:'{e.Message}' when writing an object");
```

```
            }
            catch (Exception e)
            {
                Console.WriteLine($"Unknown encountered on server.
Message:'{e.Message}' when writing an object");
            }
        }
    }
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - GetObjectMetadata
  - UploadPartCopy

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件執行 Amazon S3 物件的多部分上傳 AWS

下列程式碼範例示範如何執行分段上傳至 Amazon S3 物件。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

這些程式碼範例使用下列匯入。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
```

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

除了 AWS CRT 型 S3 用戶端之外，內容大小超過閾值時使用 S3 Transfer Manager 明確執行分段上傳。預設閾值大小為 8 MB。

```
    public void multipartUploadWithTransferManager(String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
                .putObjectRequest(b -> b
                        .bucket(bucketName)
                        .key(key))
                .source(Paths.get(filePath))
                .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

使用 S3 客戶端 API 來執行多部分上傳。

```
   public void multipartUploadWithS3Client(String filePath) {

       // Initiate the multipart upload.
       CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
               .bucket(bucketName)
               .key(key));
       String uploadId = createMultipartUploadResponse.uploadId();

       // Upload the parts of the file.
       int partNumber = 1;
       List<CompletedPart> completedParts = new ArrayList<>();
       ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

       try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
           long fileSize = file.length();
           long position = 0;
           while (position < fileSize) {
               file.seek(position);
               long read = file.getChannel().read(bb);

               bb.flip(); // Swap position and limit before reading from the
buffer.
               UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                       .bucket(bucketName)
                       .key(key)
                       .uploadId(uploadId)
                       .partNumber(partNumber)
                       .build();

               UploadPartResponse partResponse = s3Client.uploadPart(
                       uploadPartRequest,
                       RequestBody.fromByteBuffer(bb));

               CompletedPart part = CompletedPart.builder()
                       .partNumber(partNumber)
                       .eTag(partResponse.eTag())
                       .build();
               completedParts.add(part);

               bb.clear();
               position += read;
```

```
                partNumber++;
            }
        } catch (IOException e) {
            logger.error(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

使用啟用多部分支援的 S3 AsyncClient API 來執行多部分上傳。

```
    public void multipartUploadWithS3AsyncClient(String filePath) {
        // Enable multipart support.
        S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
                .multipartEnabled(true)
                .build();

        CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b
-> b
                        .bucket(bucketName)
                        .key(key),
                Paths.get(filePath));

        response.join();
        logger.info("File uploaded in multiple 8 MiB parts using
S3AsyncClient.");
    }
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - UploadPart

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS 開發套件追蹤 Amazon S3 物件上傳或下載

下列程式碼範例顯示如何追蹤 Amazon S3 物件上傳或下載。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

跟踪文件上傳的進度。

```
   public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                               String key, URI filePathURI) {
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
               .putObjectRequest(b -> b.bucket(bucketName).key(key))
               .addTransferListener(LoggingTransferListener.create())  // Add
listener.
               .source(Paths.get(filePathURI))
               .build();

       FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

       fileUpload.completionFuture().join();
       /*
           The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
           You can also implement the interface to provide your own logic.

           Configure log4J2 with settings such as the following.
               <Configuration status="WARN">
                   <Appenders>
                       <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
```

```
                        <PatternLayout pattern="%m%n"/>
                    </Console>
                </Appenders>

                <Loggers>
                    <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                        <AppenderRef ref="AlignedConsoleAppender"/>
                    </logger>
                </Loggers>
            </Configuration>

           Log4J2 logs the progress. The following is example output for a 21.3
MB file upload.
            Transfer initiated...
            |                    | 0.0%
            |====                | 21.1%
            |============        | 60.5%
            |====================| 100.0%
            Transfer complete!
       */
   }
```

跟踪文件下載的進度。

```
   public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                             String key, String downloadedFileWithPath) {
       DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
               .getObjectRequest(b -> b.bucket(bucketName).key(key))
               .addTransferListener(LoggingTransferListener.create())  // Add
listener.
               .destination(Paths.get(downloadedFileWithPath))
               .build();

       FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

       CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
       /*
```

```
            The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
            You can also implement the interface to provide your own logic.

            Configure log4J2 with settings such as the following.
                <Configuration status="WARN">
                    <Appenders>
                        <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                            <PatternLayout pattern="%m%n"/>
                        </Console>
                    </Appenders>

                    <Loggers>
                        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3
MB file download.
                Transfer initiated...
                |=======             | 39.4%
                |===============     | 78.8%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。
  - GetObject
  - PutObject

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 進行單元和集成測試的示例方法

下列程式碼範例會示範如何在使用 AWS SDK 撰寫單元和整合測試時，以取得最佳實務技術的範例。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

Cargo.toml 用於測試範例。

```
[package]
name = "testing-examples"
version = "0.1.0"
authors = [
  "John Disanti <jdisanti@amazon.com>",
  "Doug Schwartz <dougsch@amazon.com>",
]
edition = "2021"

# snippet-start:[testing.rust.Cargo.toml]
[dependencies]
async-trait = "0.1.51"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-credential-types = { version = "1.0.1", features = [ "hardcoded-
credentials", ] }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime = { version = "1.0.1", features = ["test-util"] }
aws-smithy-runtime-api = { version = "1.0.1", features = ["test-util"] }
aws-types = { version = "1.0.1" }
clap = { version = "~4.4", features = ["derive"] }
http = "0.2.9"
mockall = "0.11.4"
serde_json = "1"
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
```

```
# snippet-end:[testing.rust.Cargo.toml]

[[bin]]
name = "main"
path = "src/main.rs"
```

使用自動模擬和服務包裝函式的單元測試範例。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

// snippet-start:[testing.rust.wrapper]
// snippet-start:[testing.rust.wrapper-uses]
use aws_sdk_s3 as s3;
#[allow(unused_imports)]
use mockall::automock;

use s3::operation::list_objects_v2::{ListObjectsV2Error, ListObjectsV2Output};
// snippet-end:[testing.rust.wrapper-uses]

// snippet-start:[testing.rust.wrapper-which-impl]
#[cfg(test)]
pub use MockS3Impl as S3;
#[cfg(not(test))]
pub use S3Impl as S3;
// snippet-end:[testing.rust.wrapper-which-impl]

// snippet-start:[testing.rust.wrapper-impl]
#[allow(dead_code)]
pub struct S3Impl {
    inner: s3::Client,
}

#[cfg_attr(test, automock)]
impl S3Impl {
    #[allow(dead_code)]
    pub fn new(inner: s3::Client) -> Self {
        Self { inner }
    }

    #[allow(dead_code)]
    pub async fn list_objects(
```

```
        &self,
        bucket: &str,
        prefix: &str,
        continuation_token: Option<String>,
    ) -> Result<ListObjectsV2Output, s3::error::SdkError<ListObjectsV2Error>> {
        self.inner
            .list_objects_v2()
            .bucket(bucket)
            .prefix(prefix)
            .set_continuation_token(continuation_token)
            .send()
            .await
    }
}
// snippet-end:[testing.rust.wrapper-impl]

// snippet-start:[testing.rust.wrapper-func]
#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3_list: S3,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3_list
            .list_objects(bucket, prefix, next_token.take())
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
```

```
}
// snippet-end:[testing.rust.wrapper-func]
// snippet-end:[testing.rust.wrapper]

// snippet-start:[testing.rust.wrapper-test-mod]
#[cfg(test)]
mod test {
    // snippet-start:[testing.rust.wrapper-tests]
    use super::*;
    use mockall::predicate::eq;

    // snippet-start:[testing.rust.wrapper-test-single]
    #[tokio::test]
    async fn test_single_page() {
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
    }
    // snippet-end:[testing.rust.wrapper-test-single]

    // snippet-start:[testing.rust.wrapper-test-multiple]
    #[tokio::test]
    async fn test_multiple_pages() {
        // Create the Mock instance with two pages of objects now
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
```

```
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .set_next_continuation_token(Some("next".to_string()))
                    .build())
            });
        mock.expect_list_objects()
            .with(
                eq("test-bucket"),
                eq("test-prefix"),
                eq(Some("next".to_string())),
            )
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(3).build(),
                        s3::types::Object::builder().size(9).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        assert_eq!(19, size);
    }
    // snippet-end:[testing.rust.wrapper-test-multiple]
    // snippet-end:[testing.rust.wrapper-tests]
}
// snippet-end:[testing.rust.wrapper-test-mod]
```

集成測試示例使用 StaticReplayClient.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
```

```
// snippet-start:[testing.rust.replay-uses]
use aws_sdk_s3 as s3;
// snippet-end:[testing.rust.replay-uses]

#[allow(dead_code)]
// snippet-start:[testing.rust.replay]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3: s3::Client,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3
            .list_objects_v2()
            .prefix(prefix)
            .bucket(bucket)
            .set_continuation_token(next_token.take())
            .send()
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}
// snippet-end:[testing.rust.replay]

#[allow(dead_code)]
// snippet-start:[testing.rust.replay-tests]
// snippet-start:[testing.rust.replay-make-credentials]
fn make_s3_test_credentials() -> s3::config::Credentials {
```

```
    s3::config::Credentials::new(
        "ATESTCLIENT",
        "astestsecretkey",
        Some("atestsessiontoken".to_string()),
        None,
        "",
    )
}
// snippet-end:[testing.rust.replay-make-credentials]

// snippet-start:[testing.rust.replay-test-module]
#[cfg(test)]
mod test {
    // snippet-start:[testing.rust.replay-test-single]
    use super::*;
    use aws_config::BehaviorVersion;
    use aws_sdk_s3 as s3;
    use aws_smithy_runtime::client::http::test_util::{ReplayEvent,
 StaticReplayClient};
    use aws_smithy_types::body::SdkBody;

    #[tokio::test]
    async fn test_single_page() {
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_1.xml")))
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
```

```
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
        replay_client.assert_requests_match(&[]);
    }
    // snippet-end:[testing.rust.replay-test-single]

    // snippet-start:[testing.rust.replay-test-multiple]
    #[tokio::test]
    async fn test_multiple_pages() {
        // snippet-start:[testing.rust.replay-create-replay]
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_1.xml")))
                    .unwrap(),
            );
        let page_2 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix&continuation-token=next")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_2.xml")))
                    .unwrap(),
            );
```

```
        let replay_client = StaticReplayClient::new(vec![page_1, page_2]);
        // snippet-end:[testing.rust.replay-create-replay]
        // snippet-start:[testing.rust.replay-create-client]
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );
        // snippet-end:[testing.rust.replay-create-client]

        // Run the code we want to test with it
        // snippet-start:[testing.rust.replay-test-and-verify]
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();

        assert_eq!(19, size);

        replay_client.assert_requests_match(&[]);
        // snippet-end:[testing.rust.replay-test-and-verify]
    }
    // snippet-end:[testing.rust.replay-test-multiple]
}
// snippet-end:[testing.rust.replay-tests]
// snippet-end:[testing.rust.replay-test-module]
```

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

以遞迴的方式將本機目錄上傳至 Amazon Simple Storage Service (Amazon S3) 儲存貯體

下列程式碼範例示範如何以遞迴的方式將本機目錄上傳至 Amazon Simple Storage Service (Amazon S3) 儲存貯體。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用 S3 上TransferManager傳本地目錄。檢視完整檔案並測試。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public Integer uploadDirectory(S3TransferManager transferManager,
            URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
                .bucket(bucketName)
                .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考UploadDirectory中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS 開發套件在 Amazon S3 上傳或下載大型檔案

下列程式碼範例示範如何將大型檔案上傳至 Amazon S3，以及從中下載大型檔案。

如需詳細資訊，請參閱使用分段上傳以上傳物件。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用 Amazon S3 呼叫可在 S3 儲存貯體之間傳輸檔案的函數 TransferUtility。

```
global using System.Text;
global using Amazon.S3;
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;



// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
```

```
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
 $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
\TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");

var success = await TransferMethods.UploadSingleFileAsync(transferUtil,
 bucketName, fileToUpload, localPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
 {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();
```

```
PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil,
 bucketName, keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
 {bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDownload"];

Console.WriteLine($"Downloading {keyName} from {bucketName}.");

success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
 keyName, localPath);
if (success)
{
    Console.WriteLine("$Successfully downloaded the file, {keyName} from
 {bucketName}.");
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");
Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();
```

```
success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil,
 bucketName, s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to
 {downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();

// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}

// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);

    DisplayTitle("Amazon S3 Transfer Utility Basics");
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer
 Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to
 an\n\t  S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
 directory.");
    Console.WriteLine($"\n{sepBar}");
}

// Pauses the scenario.
static void PressEnter()
{
```

```
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\n");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth -
 textToCenter.Length) / 2));
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
    if (fileList.Length > 0)
    {
        foreach (var fileName in fileList)
        {
            Console.WriteLine(fileName);
        }
    }
}

// Displays a list of the files in the specified S3 bucket and prefix.
static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
 s3Path)
{
    ListObjectsV2Request request = new()
    {
        BucketName = bucketName,
        Prefix = s3Path,
        MaxKeys = 5,
    };

    var response = new ListObjectsV2Response();

    do
```

```
    {
        response = await client.ListObjectsV2Async(request);

        response.S3Objects
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));

        // If the response is truncated, set the request ContinuationToken
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated);
}
```

上傳單一檔案。

```
        /// <summary>
        /// Uploads a single file from the local computer to an S3 bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the file
        /// will be stored.</param>
        /// <param name="fileName">The name of the file to upload.</param>
        /// <param name="localPath">The local path where the file is stored.</
param>
        /// <returns>A boolean value indicating the success of the action.</
returns>
        public static async Task<bool> UploadSingleFileAsync(
            TransferUtility transferUtil,
            string bucketName,
            string fileName,
            string localPath)
        {
            if (File.Exists($"{localPath}\\{fileName}"))
            {
                try
                {
                    await transferUtil.UploadAsync(new
 TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
```

```
                        Key = fileName,
                        FilePath = $"{localPath}\\{fileName}",
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Could not upload {fileName} from
 {localPath} because:");
                    Console.WriteLine(s3Ex.Message);
                    return false;
                }
            }
            else
            {
                Console.WriteLine($"{fileName} does not exist in {localPath}");
                return false;
            }
        }
```

上傳整個本機目錄。

```
        /// <summary>
        /// Uploads all the files in a local directory to a directory in an S3
        /// bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the files
        /// will be stored.</param>
        /// <param name="keyPrefix">The key prefix is the S3 directory where
        /// the files will be stored.</param>
        /// <param name="localPath">The local directory that contains the files
        /// to be uploaded.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> UploadFullDirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string keyPrefix,
```

```
            string localPath)
        {
            if (Directory.Exists(localPath))
            {
                try
                {
                    await transferUtil.UploadDirectoryAsync(new
TransferUtilityUploadDirectoryRequest
                    {
                        BucketName = bucketName,
                        KeyPrefix = keyPrefix,
                        Directory = localPath,
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Can't upload the contents of {localPath}
because:");
                    Console.WriteLine(s3Ex?.Message);
                    return false;
                }
            }
            else
            {
                Console.WriteLine($"The directory {localPath} does not exist.");
                return false;
            }
        }
```

下載單一檔案。

```
        /// <summary>
        /// Download a single file from an S3 bucket to the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// file to download.</param>
```

```
        /// <param name="keyName">The name of the file to download.</param>
        /// <param name="localPath">The path on the local computer where the
        /// downloaded file will be saved.</param>
        /// <returns>A Boolean value indicating the results of the action.</
returns>
        public static async Task<bool> DownloadSingleFileAsync(
        TransferUtility transferUtil,
            string bucketName,
            string keyName,
            string localPath)
        {
            await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
            {
                BucketName = bucketName,
                Key = keyName,
                FilePath = $"{localPath}\\{keyName}",
            });

            return (File.Exists($"{localPath}\\{keyName}"));
        }
```

下載 S3 儲存貯體的內容。

```
        /// <summary>
        /// Downloads the contents of a directory in an S3 bucket to a
        /// directory on the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The bucket containing the files to
 download.</param>
        /// <param name="s3Path">The S3 directory where the files are located.</
param>
        /// <param name="localPath">The local path to which the files will be
        /// saved.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> DownloadS3DirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
```

```
            string s3Path,
            string localPath)
        {
            int fileCount = 0;

            // If the directory doesn't exist, it will be created.
            if (Directory.Exists(s3Path))
            {
                var files = Directory.GetFiles(localPath);
                fileCount = files.Length;
            }

            await transferUtil.DownloadDirectoryAsync(new
TransferUtilityDownloadDirectoryRequest
            {
                BucketName = bucketName,
                LocalDirectory = localPath,
                S3Directory = s3Path,
            });

            if (Directory.Exists(localPath))
            {
                var files = Directory.GetFiles(localPath);
                if (files.Length > fileCount)
                {
                    return true;
                }

                // No change in the number of files. Assume
                // the download failed.
                return false;
            }

            // The local directory doesn't exist. No files
            // were downloaded.
            return false;
        }
```

使用追蹤上傳進度 TransferUtility。

```
    using System;
```

```
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;

/// <summary>
/// This example shows how to track the progress of a multipart upload
/// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
/// upload to an Amazon S3 bucket.
/// </summary>
public class TrackMPUUsingHighLevelAPI
{
    public static async Task Main()
    {
        string bucketName = "doc-example-bucket";
        string keyName = "sample_pic.png";
        string path = "filepath/directory/";
        string filePath = $"{path}{keyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();

        await TrackMPUAsync(client, bucketName, filePath, keyName);
    }

    /// <summary>
    /// Starts an Amazon S3 multipart upload and assigns an event handler to
    /// track the progress of the upload.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// perform the multipart upload.</param>
    /// <param name="bucketName">The name of the bucket to which to upload
    /// the file.</param>
    /// <param name="filePath">The path, including the file name of the
    /// file to be uploaded to the Amazon S3 bucket.</param>
    /// <param name="keyName">The file name to be used in the
    /// destination Amazon S3 bucket.</param>
    public static async Task TrackMPUAsync(
        IAmazonS3 client,
        string bucketName,
        string filePath,
        string keyName)
```

```
        {
            try
            {
                var fileTransferUtility = new TransferUtility(client);

                // Use TransferUtilityUploadRequest to configure options.
                // In this example we subscribe to an event.
                var uploadRequest =
                    new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        FilePath = filePath,
                        Key = keyName,
                    };

                uploadRequest.UploadProgressEvent +=
                    new EventHandler<UploadProgressArgs>(
                        UploadRequest_UploadPartProgressEvent);

                await fileTransferUtility.UploadAsync(uploadRequest);
                Console.WriteLine("Upload completed");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:: {ex.Message}");
            }
        }

        /// <summary>
        /// Event handler to check the progress of the multipart upload.
        /// </summary>
        /// <param name="sender">The object that raised the event.</param>
        /// <param name="e">The object that contains multipart upload
        /// information.</param>
        public static void UploadRequest_UploadPartProgressEvent(object sender,
UploadProgressArgs e)
        {
            // Process event.
            Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
        }
    }
```

使用加密上傳物件。

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Uses the Amazon Simple Storage Service (Amazon S3) low level API to
/// perform a multipart upload to an Amazon S3 bucket.
/// </summary>
public class SSECLowLevelMPUcopyObject
{
    public static async Task Main()
    {
        string existingBucketName = "doc-example-bucket";
        string sourceKeyName = "sample_file.txt";
        string targetKeyName = "sample_file_copy.txt";
        string filePath = $"sample\\{targetKeyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USEast1.
        IAmazonS3 client = new AmazonS3Client();

        // Create the encryption key.
        var base64Key = CreateEncryptionKey();

        await CreateSampleObjUsingClientEncryptionKeyAsync(
            client,
            existingBucketName,
            sourceKeyName,
            filePath,
            base64Key);
    }

    /// <summary>
    /// Creates the encryption key to use with the multipart upload.
    /// </summary>
    /// <returns>A string containing the base64-encoded key for encrypting
```

```
        /// the multipart upload.</returns>
        public static string CreateEncryptionKey()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);
            return base64Key;
        }

        /// <summary>
        /// Creates and uploads an object using a multipart upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object used to
        /// initialize and perform the multipart upload.</param>
        /// <param name="existingBucketName">The name of the bucket to which
        /// the object will be uploaded.</param>
        /// <param name="sourceKeyName">The source object name.</param>
        /// <param name="filePath">The location of the source object.</param>
        /// <param name="base64Key">The encryption key to use with the upload.</
param>
        public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
            IAmazonS3 client,
            string existingBucketName,
            string sourceKeyName,
            string filePath,
            string base64Key)
        {
            List<UploadPartResponse> uploadResponses = new
 List<UploadPartResponse>();

            InitiateMultipartUploadRequest initiateRequest = new
 InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
               await client.InitiateMultipartUploadAsync(initiateRequest);
```

```
            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                    {
                        BucketName = existingBucketName,
                        Key = sourceKeyName,
                        UploadId = initResponse.UploadId,
                        PartNumber = i,
                        PartSize = partSize,
                        FilePosition = filePosition,
                        FilePath = filePath,
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key,
                    };

                    // Upload part and add response to our list.
                    uploadResponses.Add(await
client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine($"Exception occurred: {exception.Message}");
```

```
                // If there was an error, abort the multipart upload.
                AbortMultipartUploadRequest abortMPURequest = new
 AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                };

                await client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
    }
```

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用上傳管理員將資料分成多個部分，並同時上傳它們來上傳大型物件。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}
```

```
// UploadLargeObject uses an upload manager to upload data to an object in a
 bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(bucketName string, objectKey string,
 largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }

 return err
}
```

使用下載管理員分段取得資料並同時下載它們來下載大型物件。

```
// DownloadLargeObject uses a download manager to download an object from a
 bucket.
// The download manager gets the data in parts and writes them to a buffer until
 all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(bucketName string, objectKey
 string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
 {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(context.TODO(), buffer, &s3.GetObjectInput{
```

```
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用 S3 呼叫在 S3 儲存貯體之間傳輸檔案的函數TransferManager。

```
    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
            URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

上傳整個本機目錄。

```
   public Integer uploadDirectory(S3TransferManager transferManager,
           URI sourceDirectory, String bucketName) {
       DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
               .source(Paths.get(sourceDirectory))
               .bucket(bucketName)
               .build());

       CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
       completedDirectoryUpload.failedTransfers()
               .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
       return completedDirectoryUpload.failedTransfers().size();
   }
```

上傳單一檔案。

```
   public String uploadFile(S3TransferManager transferManager, String
bucketName,
                            String key, URI filePathURI) {
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
           .putObjectRequest(b -> b.bucket(bucketName).key(key))
           .source(Paths.get(filePathURI))
           .build();

       FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

       CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
       return uploadResult.response().eTag();
   }
```

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

上傳大型檔案。

```
import {
  CreateMultipartUploadCommand,
  UploadPartCommand,
  CompleteMultipartUploadCommand,
  AbortMultipartUploadCommand,
  S3Client,
} from "@aws-sdk/client-s3";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

export const main = async () => {
  const s3Client = new S3Client({});
  const bucketName = "test-bucket";
  const key = "multipart.txt";
  const str = createString();
  const buffer = Buffer.from(str, "utf8");

  let uploadId;

  try {
    const multipartUpload = await s3Client.send(
      new CreateMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
```

```
    uploadId = multipartUpload.UploadId;

    const uploadPromises = [];
    // Multipart uploads require a minimum size of 5 MB per part.
    const partSize = Math.ceil(buffer.length / 5);

    // Upload each part.
    for (let i = 0; i < 5; i++) {
      const start = i * partSize;
      const end = start + partSize;
      uploadPromises.push(
        s3Client
          .send(
            new UploadPartCommand({
              Bucket: bucketName,
              Key: key,
              UploadId: uploadId,
              Body: buffer.subarray(start, end),
              PartNumber: i + 1,
            }),
          )
          .then((d) => {
            console.log("Part", i + 1, "uploaded");
            return d;
          }),
      );
    }

    const uploadResults = await Promise.all(uploadPromises);

    return await s3Client.send(
      new CompleteMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
        MultipartUpload: {
          Parts: uploadResults.map(({ ETag }, i) => ({
            ETag,
            PartNumber: i + 1,
          })),
        },
      }),
    );
```

```
    // Verify the output by downloading the file from the Amazon Simple Storage
 Service (Amazon S3) console.
    // Because the output is a 25 MB string, text editors might struggle to open
 the file.
  } catch (err) {
    console.error(err);

    if (uploadId) {
      const abortCommand = new AbortMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
      });

      await s3Client.send(abortCommand);
    }
  }
};
```

下載大型檔案。

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream } from "fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
```

```
  const [start, end] = range.split("-");
  return {
    start: parseInt(start),
    end: parseInt(end),
    length: parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

// When downloading a large file, you might want to break it down into
// smaller pieces. Amazon S3 accepts a Range header to specify the start
// and end of the byte range to be downloaded.
const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    console.log(`Downloading bytes ${nextRange.start} to ${nextRange.end}`);

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
};

export const main = async () => {
  await downloadInChunks({
    bucket: "my-cool-bucket",
    key: "my-cool-object.txt",
  });
};
```

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

建立使用多個可用的傳輸管理設定函數傳輸檔案。使用回呼類別將回呼程序寫入檔案傳輸。

```
import sys
import threading

import boto3
from boto3.s3.transfer import TransferConfig


MB = 1024 * 1024
s3 = boto3.resource("s3")


class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
```

```
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
```

```
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.
```

```
    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
 sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
```

```
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
 sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

展示傳輸管理函數並回報結果。

```
import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")


class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
```

```
        self._size_multiplier = 0
        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
files? "
            )
            if not os.path.isdir(self.demo_folder):
                print(f"{self.demo_folder} isn't a folder!")
                self.demo_folder = None

        while not self.demo_bucket:
            self.demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
                "demonstration files? "
            )
            try:
                self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
            except ParamValidationError as err:
                print(err)
                self.demo_bucket = None
            except ClientError as err:
                print(err)
                print(
                    f"Either {self.demo_bucket} doesn't exist or you don't "
                    f"have access to it."
                )
                self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
download_args=None
    ):
        """Run a demonstration.
```

```
        Ask the user if they want to run this specific demonstration.
        If they say yes, create a file on the local path, upload it
        using the specified upload function, then download it using the
        specified download function.
        """
        if download_args is None:
            download_args = {}
        if upload_args is None:
            upload_args = {}
        question = question.format(self.file_size_mb)
        answer = input(f"{question} (y/n)")
        if answer.lower() == "y":
            local_file_path, object_key, download_file_path =
self._create_demo_file()

            file_transfer.TransferConfig = self._config_wrapper(
                TransferConfig, CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Uploading", local_file_path, object_key, **upload_args
            )
            start_time = time.perf_counter()
            thread_info = upload_func(
                local_file_path,
                self.demo_bucket,
                object_key,
                self.file_size_mb,
                **upload_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

            file_transfer.TransferConfig = self._config_wrapper(
                TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Downloading", object_key, download_file_path, **download_args
            )
            start_time = time.perf_counter()
            thread_info = download_func(
                self.demo_bucket,
                object_key,
                download_file_path,
```

```
                self.file_size_mb,
                **download_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

    def last_name_set(self):
        """Get the name set used for the last demo."""
        return self._chore_list[-1]

    def cleanup(self):
        """
        Remove files from the demo folder, and uploaded objects from the
        Amazon S3 bucket.
        """
        print("-" * self._terminal_width)
        for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
            print(f"Removing {local_file_path}")
            try:
                os.remove(local_file_path)
            except FileNotFoundError as err:
                print(err)

            print(f"Removing {downloaded_file_path}")
            try:
                os.remove(downloaded_file_path)
            except FileNotFoundError as err:
                print(err)

            if self.demo_bucket:
                print(f"Removing {self.demo_bucket}:{s3_object_key}")
                try:

self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
                except ClientError as err:
                    print(err)

    def _setup_platform_specific(self):
        """Set up platform-specific command used to create a large file."""
        if platform.system() == "Windows":
            self._create_file_cmd = "fsutil file createnew {} {}"
            self._size_multiplier = MB
        elif platform.system() == "Linux" or platform.system() == "Darwin":
```

```
            self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={MB}
count={{}}"
            self._size_multiplier = 1
        else:
            raise EnvironmentError(
                f"Demo of platform {platform.system()} isn't supported."
            )

    def _create_demo_file(self):
        """
        Create a file in the demo folder specified by the user. Store the local
        path, object name, and download path for later cleanup.

        Only the local file is created by this method. The Amazon S3 object and
        download file are created later during the demonstration.

        Returns:
        A tuple that contains the local file path, object name, and download
        file path.
        """
        file_name_template = "TestFile{}-{}.demo"
        local_suffix = "local"
        object_suffix = "s3object"
        download_suffix = "downloaded"
        file_tag = len(self._chore_list) + 1

        local_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag, local_suffix)
        )

        s3_object_key = file_name_template.format(file_tag, object_suffix)

        downloaded_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag,
download_suffix)
        )

        filled_cmd = self._create_file_cmd.format(
            local_file_path, self.file_size_mb * self._size_multiplier
        )

        print(
            f"Creating file of size {self.file_size_mb} MB "
            f"in {self.demo_folder} by running:"
```

```
        )
        print(f"{'':4}{filled_cmd}")
        os.system(filled_cmd)

        chore = (local_file_path, s3_object_key, downloaded_file_path)
        self._chore_list.append(chore)
        return chore

    def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
        """Report configuration and extra arguments used for a file transfer."""
        print("-" * self._terminal_width)
        print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
        if kwargs:
            print("With extra args:")
            for arg, value in kwargs.items():
                print(f'{"":4}{arg:<20}: {value}')

    @staticmethod
    def ask_user(question):
        """
        Ask the user a yes or no question.

        Returns:
        True when the user answers 'y' or 'Y'; otherwise, False.
        """
        answer = input(f"{question} (y/n) ")
        return answer.lower() == "y"

    @staticmethod
    def _config_wrapper(func, config_attrs):
        def wrapper(*args, **kwargs):
            config = func(*args, **kwargs)
            print("With configuration:")
            for attr in config_attrs:
                print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
            return config

        return wrapper

    @staticmethod
    def _report_transfer_result(thread_info, elapsed):
        """Report the result of a transfer, including per-thread data."""
        print(f"\nUsed {len(thread_info)} threads.")
        for ident, byte_count in thread_info.items():
```

```
            print(f"{'':4}Thread {ident} copied {byte_count} bytes.")
        print(f"Your transfer took {elapsed:.2f} seconds.")


def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold,
    )

    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksize_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
            "metadata": {
                "upload_type": "chunky",
                "favorite_color": "aqua",
                "size": "medium",
            }
```

```
        },
    )

    # Upload using server-side encryption with customer-provided
    # encryption keys.
    # Generate a 256-bit key from a passphrase.
    sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
    demo_manager.demo(
        "Do you want to upload and download a {} MB file using "
        "server-side encryption?",
        file_transfer.upload_with_sse,
        file_transfer.download_with_sse,
        upload_args={"sse_key": sse_key},
        download_args={"sse_key": sse_key},
    )

    # Download without specifying an encryption key to show that the
    # encryption key must be included to download an encrypted object.
    if demo_manager.ask_user(
        "Do you want to try to download the encrypted "
        "object without sending the required key?"
    ):
        try:
            _, object_key, download_file_path = demo_manager.last_name_set()
            file_transfer.download_with_default_configuration(
                demo_manager.demo_bucket,
                object_key,
                download_file_path,
                demo_manager.file_size_mb,
            )
        except ClientError as err:
            print(
                "Got expected error when trying to download an encrypted "
                "object without specifying encryption info:"
            )
            print(f"{'':4}{err}")

    # Remove all created and downloaded files, remove all objects from
    # S3 storage.
    if demo_manager.ask_user(
        "Demonstration complete. Do you want to remove local files " "and S3
objects?"
    ):
        demo_manager.cleanup()
```

```
if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
use std::fs::File;
use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::error::DisplayErrorContext;
use aws_sdk_s3::operation::{
    create_multipart_upload::CreateMultipartUploadOutput,
 get_object::GetObjectOutput,
};
use aws_sdk_s3::types::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::{config::Region, Client as S3Client};
use aws_smithy_types::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use s3_service::error::Error;
use std::process;
```

```
use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() {
    if let Err(err) = run_example().await {
        eprintln!("Error: {}", DisplayErrorContext(err));
        process::exit(1);
    }
}

async fn run_example() -> Result<(), Error> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("doc-example-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    s3_service::create_bucket(&client, &bucket_name, region.as_ref()).await?;

    let key = "sample.txt".to_string();
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await
        .unwrap();
    let upload_id = multipart_upload_res.upload_id().unwrap();

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)
            .take(256)
            .map(char::from)
            .collect();
        let return_string: String = "\n".to_string();
```

```
        file.write_all(rand_string.as_ref())
            .expect("Error writing to file.");
        file.write_all(return_string.as_ref())
            .expect("Error writing to file.");
    }

    let path = Path::new(&key);
    let file_size = tokio::fs::metadata(path)
        .await
        .expect("it exists I swear")
        .len();

    let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
    let mut size_of_last_chunk = file_size % CHUNK_SIZE;
    if size_of_last_chunk == 0 {
        size_of_last_chunk = CHUNK_SIZE;
        chunk_count -= 1;
    }

    if file_size == 0 {
        panic!("Bad file size.");
    }
    if chunk_count > MAX_CHUNKS {
        panic!("Too many chunks! Try increasing your chunk size.")
    }

    let mut upload_parts: Vec<CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();
        //Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
```

```
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;
        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }
    let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await
        .unwrap();

    let data: GetObjectOutput = s3_service::download_object(&client,
&bucket_name, &key).await?;
    let data_length: u64 = data
        .content_length()
        .unwrap_or_default()
        .try_into()
        .unwrap();
    if file.metadata().unwrap().len() == data_length {
        println!("Data lengths match.");
    } else {
        println!("The data was not the same size!");
    }

    s3_service::delete_objects(&client, &bucket_name)
```

```
        .await
        .expect("Error emptying bucket.");
    s3_service::delete_bucket(&client, &bucket_name)
        .await
        .expect("Error deleting bucket.");

    Ok(())
}
```

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS 開發套件將大小不明的串流上傳至 Amazon S3 物件

下列程式碼範例顯示如何將大小不明的串流上傳至 Amazon S3 物件。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用 AWS CRT 型 S3 用戶端。

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

```
    /**
     * @param s33CrtAsyncClient - To upload content from a stream of unknown
 size, use the AWS CRT-based S3 client. For more information, see
     *                          https://docs.aws.amazon.com/sdk-for-java/latest/
developer-guide/crt-based-s3-client.html.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
 Returns metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStream(S3AsyncClient s33CrtAsyncClient,
 String bucketName, String key) {

        BlockingInputStreamAsyncRequestBody body =
                AsyncRequestBody.forBlockingInputStream(null); // 'null'
 indicates a stream will be provided later.

        CompletableFuture<PutObjectResponse> responseFuture =
                s33CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
 body);

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());

        // Provide the stream of data to be uploaded.
        body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        return response;
    }
}
```

使用 Amazon S3 Transfer Manager。

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.util.UUID;

    /**
     * @param transferManager - To upload content from a stream of unknown size,
 use the S3TransferManager based on the AWS CRT-based S3 client.
     *                        For more information, see https://
docs.aws.amazon.com/sdk-for-java/latest/developer-guide/transfer-manager.html.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
 result of the completed upload.
     */
    public CompletedUpload uploadStream(S3TransferManager transferManager, String
 bucketName, String key) {

        BlockingInputStreamAsyncRequestBody body =
                AsyncRequestBody.forBlockingInputStream(null); // 'null'
 indicates a stream will be provided later.

        Upload upload = transferManager.upload(builder -> builder
                .requestBody(body)
                .putObjectRequest(req -> req.bucket(bucketName).key(key))
                .build());

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());

        // Provide the stream of data to be uploaded.
        body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

        return upload.completionFuture().join();
    }
```

```
}
```

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用總和檢查碼搭配使用開發套件的 Amazon S3 物件 AWS

下列程式碼範例示範如何使用總和檢查搭配 Amazon S3 物件。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

這些程式碼範例使用下列匯入的子集。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
```

```
import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

當您建置 PutObjectRequest時，為 putObject 方法指定總和檢查演算法。

```
    public void putObjectWithChecksum() {
        s3Client.putObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.CRC32),
                RequestBody.fromString("This is a test"));
    }
```

建置時，請驗證getObject方法的總和檢查碼。 GetObjectRequest

```
    public GetObjectResponse getObjectWithChecksum() {
        return s3Client.getObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumMode(ChecksumMode.ENABLED))
                .response();
    }
```

當您建置 PutObjectRequest時，為 putObject 方法預先計算總和檢查。

```
    public void putObjectWithPrecalculatedChecksum(String filePath) {
```

```
        String checksum = calculateChecksum(filePath, "SHA-256");

        s3Client.putObject((b -> b
                .bucket(bucketName)
                .key(key)
                .checksumSHA256(checksum)),
                RequestBody.fromFile(Paths.get(filePath)));
    }
```

除了 AWS CRT 型 S3 用戶端之外，內容大小超過閾值時使用 S3 Transfer Manager 明確執行分段上傳。預設閾值大小為 8 MB。

您可以指定要讓 SDK 使用的總和檢查演算法。根據預設，SDK 會使用 CRC32 演算法。

```
    public void multipartUploadWithChecksumTm(String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
                .putObjectRequest(b -> b
                        .bucket(bucketName)
                        .key(key)
                        .checksumAlgorithm(ChecksumAlgorithm.SHA1))
                .source(Paths.get(filePath))
                .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

使用 S3 客戶端 API 或（S3 AsyncClient API）執行多部分上傳。如果您指定額外的總和檢查，則必須指定在上傳初始化時要使用的演算法。您還必須為每一個分段請求指定演算法，並在每一個分段上傳後提供為其計算的總和檢查。

```
    public void multipartUploadWithChecksumS3Client(String filePath) {
        ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
```

```
                .checksumAlgorithm(algorithm)); // Checksum specified on
initiation.
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .checksumAlgorithm(algorithm) // Checksum specified on
each part.
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .checksumCRC32(partResponse.checksumCRC32()) // Provide
the calculated checksum.
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
```

```
            System.err.println(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - UploadPart

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件使用 Amazon S3 版本控制物件 AWS

以下程式碼範例顯示做法：

- 建立已使用版本控制的 S3 儲存貯體。
- 取得物件的所有版本。
- 將物件復原至先前的版本。
- 刪除並還原已使用版本控制的物件。
- 永久刪除物件的所有版本。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

建立可包裝 S3 動作的函數。

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
```

```
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
 expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket



def rollback_object(bucket, object_key, version_id):
    """
```

```
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
f"{object_key}."
        )
```

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest
 version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
```

```
        logger.warning("Got an active version for %s, nothing to do.",
 object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)



def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

上傳詩句的片段至已使用版本控制的物件，並執行一系列動作。

```
def usage_demo_single_object(obj_prefix="demo-versioning/"):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open("father_william.txt") as file:
        stanzas = file.read().split("\n\n")

    width = get_terminal_size((80, 20))[0]
    print("-" * width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print(
```

```
        "This demonstration uploads a single stanza of a poem to an Amazon "
        "S3 bucket and then applies various revisions to it."
    )
    print("-" * width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket("bucket-" + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
    print("\nApplying some revisions to the stanza...")
    obj_stanza_1 = bucket.Object(f"{obj_prefix}stanza-1")
    obj_stanza_1.put(Body=bytes(stanzas[0], "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].upper(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].lower(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0][::-1], "utf-8"))
    print(
        "The latest version of the stanza is now:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Versions are returned in order, most recent first.
    obj_stanza_1_versions =
 bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    print(
        "The version data of the stanza revisions:",
        *[
            f"    {version.version_id}, last modified {version.last_modified}"
            for version in obj_stanza_1_versions
        ],
        sep="\n",
    )

    # Rollback two versions.
    print("\nRolling back two versions...")
    rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
    print(
        "The latest version of the stanza:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )
```

```
    # Delete the stanza
    print("\nDeleting the stanza...")
    obj_stanza_1.delete()
    try:
        obj_stanza_1.get()
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            print("The stanza is now deleted (as expected).")
        else:
            raise

    # Revive the stanza
    print("\nRestoring the stanza...")
    revive_object(bucket, obj_stanza_1.key)
    print(
        "The stanza is restored! The latest version is again:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Permanently delete all versions of the object. This cannot be undone!
    print("\nPermanently deleting all versions of the stanza...")
    permanently_delete_object(bucket, obj_stanza_1.key)
    obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    if len(list(obj_stanza_1_versions)) == 0:
        print("The stanza has been permanently deleted and now has no versions.")
    else:
        print("Something went wrong. The stanza still exists!")

    print(f"\nRemoving {bucket.name}...")
    bucket.delete()
    print(f"{bucket.name} deleted.")
    print("Demo done!")
```

- 如需 API 的詳細資訊，請參閱《適用於 Python (Boto3) 的AWS SDK API 參考資料》中的下列主題。
  - CreateBucket

- DeleteObject
- ListObjectVersions
- PutBucketLifecycleConfiguration

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使 AWS 用開發套件的 Amazon S3 無伺服器範例

下列程式碼範例說明如何搭配 AWS 開發套件使用 Amazon S3。

範例

- 使用 Amazon S3 觸發條件調用 Lambda 函數

使用 Amazon S3 觸發條件調用 Lambda 函數

下列程式碼範例顯示如何實作 Lambda 函數來接收上傳物件至 S3 儲存貯體時觸發的事件。此函數會從事件參數擷取 S3 儲存貯體名稱和物件金鑰，並呼叫 Amazon S3 API 以擷取和記錄物件的內容類型。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 .NET 搭配 Lambda 來使用 S3 事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
```

```
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
        {
        }

        internal Function(AmazonS3Client s3Client)
        {
            _s3Client = s3Client ?? new AmazonS3Client();
        }

        public async Task<string> Handler(S3Event evt, ILambdaContext context)
        {
            try
            {
                if (evt.Records.Count <= 0)
                {
                    context.Logger.LogLine("Empty S3 Event received");
                    return string.Empty;
                }

                var bucket = evt.Records[0].S3.Bucket.Name;
                var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

                context.Logger.LogLine($"Request is for {bucket} and {key}");

                var objectResult = await _s3Client.GetObjectAsync(bucket, key);

                context.Logger.LogLine($"Returning {objectResult.Key}");

                return objectResult.Key;
            }
            catch (Exception e)
            {
```

```
                context.Logger.LogLine($"Error processing request -
 {e.Message}");

                return string.Empty;
            }
        }
    }
}
```

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Go 搭配 Lambda 來使用 S3 事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Printf("failed to load default config: %s", err)
  return err
 }
 s3Client := s3.NewFromConfig(sdkConfig)
```

```
 for _, record := range s3Event.Records {
  bucket := record.S3.Bucket.Name
  key := record.S3.Object.URLDecodedKey
  headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
   Bucket: &bucket,
   Key:    &key,
  })
  if err != nil {
   log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
   return err
  }
  log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
 *headOutput.ContentType)
 }

 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Java 搭配 Lambda 來使用 S3 事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
```

```
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotifi

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();

          S3Client s3Client = S3Client.builder().build();
          HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);

          logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

          return "Ok";
        } catch (Exception e) {
          throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使 Lambda JavaScript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

exports.handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g,
 ' '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make
 sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

使 Lambda TypeScript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
  // Get the object from the event and show its content type
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  try {
    const { ContentType } = await s3.send(new HeadObjectCommand(params));
    console.log('CONTENT TYPE:', ContentType);
    return ContentType;
  } catch (err) {
    console.log(err);
    const message = `Error getting object ${key} from bucket ${bucket}. Make sure
 they exist and your bucket is in the same region as this function.`;
    console.log(message);
    throw new Error(message);
  }
};
```

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 PHP 使用 Lambda 使用 S3 事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';


class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' .
 $bucket . '. Make sure they exist and your bucket is in the same region as this
 function.' . "\n";
                throw $e;
```

```
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Python 搭配 Lambda 來使用 S3 事件。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
 encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
```

```
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and
 your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用紅寶石與 Lambda 一個 S3 事件。

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
```

```
    raise e
  end
end
```

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在無伺服器範例儲存庫中設定和執行。

使用 Rust 搭配 Lambda 來使用 S3 事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};


/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
```

```
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
    tracing::info!(records = ?evt.payload.records.len(), "Received request from
 SQS");

    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket
 name to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
 exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await;

    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
 contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }

    Ok(())
}
```

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使 AWS 用開發套件的 Amazon S3 跨服務範例

下列範例應用程式使用 AWS 開發套件將 Amazon S3 與其他 AWS 服務應用程式結合使用。每個範例都包含一個連結 GitHub，您可以在其中找到如何設定和執行應用程式的指示。

範例

- 建置 Amazon Transcribe 應用程式
- 使用 AWS SDK 將文本轉換為語音並返回文本
- 建立相片資產管理應用程式，讓使用者以標籤管理相片
- 建立 Amazon Textract Explorer 應用程式
- 使用開發套件使用 Amazon 重新認知功能偵測影像中的個人防護裝置 AWS
- 使用 AWS SDK 檢測從圖像中提取的文本中的實體
- 使用 AWS SDK 偵測影像中的臉孔
- 使用開發套件使用 Amazon Rekognition 偵測影像中的物件 AWS
- 使用開發套件使用 Amazon Rekognition 偵測影片中的人物和物件 AWS
- 使用 SDK 儲存 EXIF 和其他影像資訊 AWS
- 使用 S3 物件 Lambda 為您的應用程式轉換資料

建置 Amazon Transcribe 應用程式

下列程式碼範例示範如何使用 Amazon Transcribe 在瀏覽器中轉錄和顯示語音錄音。

JavaScript

適用於 JavaScript (v3) 的開發套件

建立使用 Amazon Transcribe 的應用程式，在瀏覽器中轉錄和顯示語音錄音。應用程式使用兩個 Amazon Simple Storage Service (Amazon S3) 儲存貯體，一個負責支援應用程式程式碼，另一個負責存放文字記錄。應用程式會使用 Amazon Cognito 使用者集區來對您的使用者進行身分驗證。經過驗證的使用者具有 AWS Identity and Access Management (IAM) 許可以存取所需 AWS 服務。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例也可在 AWS SDK for JavaScript v3 開發人員指南中取得。

此範例中使用的服務

- Amazon Cognito Identity
- Amazon S3
- Amazon Transcribe

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 將文本轉換為語音並返回文本

以下程式碼範例顯示做法：

- 使用 Amazon Polly 將純文字 (UTF-8) 輸入檔案合成至音訊檔案中。
- 將音訊檔案上傳至 Amazon S3 儲存貯體。
- 使用 Amazon Transcribe 將音訊檔案轉換為文字。
- 顯示文字。

Rust

適用於 Rust 的 SDK

使用 Amazon Polly 將純文字 (UTF-8) 輸入檔案合成至音訊檔案中，將音訊檔案上傳至 Amazon S3 儲存貯體，使用 Amazon Transcribe 將該音訊檔案轉換為文字，然後顯示文字。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Polly
- Amazon S3
- Amazon Transcribe

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

建立相片資產管理應用程式，讓使用者以標籤管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

.NET

AWS SDK for .NET

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

適用於 C++ 的 SDK

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3

- Amazon SNS

Java

適用於 Java 2.x 的 SDK

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

適用於 JavaScript (v3) 的開發套件

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

適用於 Kotlin 的 SDK

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

適用於 PHP 的開發套件

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

適用於 Rust 的 SDK

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

如要深入探索此範例的來源，請參閱 AWS 社群上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

建立 Amazon Textract Explorer 應用程式

下列程式碼範例示範如何透過互動式應用程式探索 Amazon Textract 輸出。

JavaScript

適用於 JavaScript (v3) 的開發套件

示範如何使用建置 React 應用程式，該應用程式使用 Amazon Textract 擷取文件影像中的資料，並將其顯示在互動式網頁中。 AWS SDK for JavaScript 此範例會在 Web 瀏覽器中

執行，且登入資料需要經過驗證的 Amazon Cognito 身分。它使用 Amazon Simple Storage Service (Amazon S3 進行儲存，對於通知，它會輪詢訂閱 Amazon Simple Notification Service (Amazon SNS)) 主題的 Amazon Simple Queue Service (Amazon SQS) 佇列。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

適用於 Python (Boto3) 的 SDK

示範如何使用 AWS SDK for Python (Boto3) 搭配 Amazon Textract 來偵測文件影像中的文字、表單和表格元素。輸入影像和 Amazon Textract 輸出會顯示在 Tkinter 應用程式中，可讓您探索偵測到的元素。

- 將文件影像提交到 Amazon Textract，並探索偵測到元素的輸出。
- 將影像直接傳送至 Amazon Textract 或透過 Amazon Simple Storage Service (Amazon S3) 儲存貯體。
- 使用非同步 API 可以在任務完成時啟動將通知發布到 Amazon Simple Notification Service (Amazon SNS) 主題的任務。
- 輪詢 Amazon Simple Queue Service (Amazon SQS) 佇列以取得任務完成訊息並顯示結果。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件使用 Amazon 重新認知功能偵測影像中的個人防護裝置 AWS

以下程式碼範例說明如何建置可使用 Amazon Rekognition 在映像中偵測個人防護裝備 (PPE) 的應用程式。

Java

適用於 Java 2.x 的 SDK

說明如何建立使用個人防護裝備偵測影像的 AWS Lambda 功能。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

適用於 JavaScript (v3) 的開發套件

示範如何搭配使用 Amazon Rekognition AWS SDK for JavaScript 來建立應用程式，以偵測位於亞馬遜簡單儲存服務 (Amazon S3) 儲存貯體中的映像中的個人防護設備 (PPE)。該應用程式將結果儲存到 Amazon DynamoDB 資料表中，並使用 Amazon Simple Email Service (Amazon SES) 向管理員傳送包含結果的電子郵件通知。

了解如何：

- 使用 Amazon Cognito 建立未經身分驗證的使用者。
- 使用 Amazon Rekognition 分析映像中是否具有 PPE。
- 驗證 Amazon SES 的電子郵件地址。
- 以結果更新 DynamoDB 資料表。
- 使用 Amazon SES 傳送電子郵件通知。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 檢測從圖像中提取的文本中的實體

下列程式碼範例示範如何使用 Amazon Comprehend 偵測 Amazon Textract 從存放在 Amazon S3 中的影像中提取的文字中的實體。

Python

適用於 Python (Boto3) 的 SDK

示範如何使用 Jupyter 筆記本 AWS SDK for Python (Boto3) 中的偵測從影像擷取的文字中的實體。本範例使用 Amazon Textract 從儲存於 Amazon Simple Storage Service (Amazon S3) 和 Amazon Comprehend 中的影像提取文字，以偵測擷取文字中的實體。

此範例是 Jupyter 的筆記型電腦，必須在可以託管的筆記型電腦的環境中運行。有關如何使用 Amazon 運行示例的說明 SageMaker，請參閱 TextractAndComprehendNotebook.ip ynb 中的說明。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Comprehend
- Amazon S3
- Amazon Textract

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 偵測影像中的臉孔

以下程式碼範例顯示做法：

- 在 Amazon S3 儲存貯體儲存映像。
- 使用 Amazon Rekognition 偵測面部細節，例如年齡範圍、性別和情感 (例如微笑)。
- 顯示這些詳細資訊。

Rust

適用於 Rust 的 SDK

將映像儲存在 Amazon S3 儲存貯體中，並包含上傳字首，使用 Amazon Rekognition 偵測面部細節，例如年齡範圍、性別和情感 (微笑等)，並顯示這些詳細資訊。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Rekognition
- Amazon S3

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件使用 Amazon Rekognition 偵測影像中的物件 AWS

下列程式碼範例說明如何建置可使用 Amazon Rekognition 按類別偵測映像中物件的應用程式。

.NET

AWS SDK for .NET

說明如何使用 Amazon Rekognition .NET API 建立應用程式，該應用程式可使用 Amazon Rekognition 對 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的映像按類別識別物件。此應用程式可使用 Amazon Simple Email Service (Amazon SES) 向管理員傳送包含結果的電子郵件通知。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

適用於 Java 2.x 的 SDK

說明如何使用 Amazon Rekognition Java API 建立應用程式，該應用程式可使用 Amazon Rekognition 對 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的映像按類別識別物件。此應用程式可使用 Amazon Simple Email Service (Amazon SES) 向管理員傳送包含結果的電子郵件通知。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

適用於 JavaScript (v3) 的開發套件

示範如何搭配使用 Amazon Rekognition AWS SDK for JavaScript 來建立使用 Amazon Rekognition 的應用程式，在位於亞馬遜簡單儲存服務 (Amazon S3) 儲存貯體中的映像中依類別識別物件。此應用程式可使用 Amazon Simple Email Service (Amazon SES) 向管理員傳送包含結果的電子郵件通知。

了解如何：

- 使用 Amazon Cognito 建立未經身分驗證的使用者。
- 使用 Amazon Rekognition 分析映像中的物件。
- 驗證 Amazon SES 的電子郵件地址。
- 使用 Amazon SES 傳送電子郵件通知。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

適用於 Kotlin 的 SDK

展示如何使用 Amazon Rekognition Kotlin API 建立應用程式，該應用程式使用 Amazon Rekognition 對位於 Amazon Simple Storage Service (Amazon S3) 儲存貯體中的映像按類別識別物件。此應用程式可使用 Amazon Simple Email Service (Amazon SES) 向管理員傳送包含結果的電子郵件通知。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

適用於 Python (Boto3) 的 SDK

說明如何使用建立可讓您執行下列作業的 Web 應用程式： AWS SDK for Python (Boto3)

- 將相片上傳到 Amazon Simple Storage Service (Amazon S3) 儲存貯體。
- 使用 Amazon Rekognition 分析和標籤照片。
- 使用 Amazon Simple Email Service (Amazon SES) 傳送映像分析的電子郵件報告。

這個例子包含兩個主要組成部分：一個用 React 構建的網頁，以及一個用 Python 編寫的 REST 服務，它是用燒瓶 REST 構建的。 JavaScript

您可以使用 React 網頁執行以下操作：

- 顯示儲存於 S3 儲存貯體中的映像的清單。

- 將映像從您的電腦上傳至 S3 儲存貯體。
- 顯示識別映像中偵測到的專案的映像和標籤。
- 取得 S3 儲存貯體中所有映像的報告，並傳送報告的電子郵件。

該網頁呼叫 REST 服務。該服務將請求發送到 AWS 來執行下列動作：

- 取得並篩選 S3 儲存貯體中的映像的清單。
- 將相片上傳至 S3 儲存貯體。
- 使用 Amazon Rekognition 分析個別照片，並取得標識照片中偵測到的專案的標籤清單。
- 分析 S3 儲存貯體中的所有相片，然後使用 Amazon SES 傳送報告的電子郵件。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Rekognition
- Amazon S3
- Amazon SES

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件使用 Amazon Rekognition 偵測影片中的人物和物件 AWS

下列程式碼範例示範如何使用 Amazon Rekognition 偵測映像中的人物和物件。

Java

適用於 Java 2.x 的 SDK

示範如何使用 Amazon Rekognition Java API 來建立應用程式，以偵測位於 Amazon Simple Storage Service (Amazon S3) 儲存貯體的映像中的人臉和物件。此應用程式可使用 Amazon Simple Email Service (Amazon SES) 向管理員傳送包含結果的電子郵件通知。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Rekognition

- Amazon S3
- Amazon SES

JavaScript

適用於 JavaScript (v3) 的開發套件

示範如何搭配使用 Amazon Rekognition AWS SDK for JavaScript 來建立應用程式，以偵測位於亞馬遜簡單儲存服務 (Amazon S3) 儲存貯體中影片中的臉部和物件。此應用程式可使用 Amazon Simple Email Service (Amazon SES) 向管理員傳送包含結果的電子郵件通知。

了解如何：

- 使用 Amazon Cognito 建立未經身分驗證的使用者。
- 使用 Amazon Rekognition 分析映像中是否具有 PPE。
- 驗證 Amazon SES 的電子郵件地址。
- 使用 Amazon SES 傳送電子郵件通知。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Rekognition
- Amazon S3
- Amazon SES

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 SDK 儲存 EXIF 和其他影像資訊 AWS

以下程式碼範例顯示做法：

- 從 JPG、JPEG 或 PNG 檔案中取得 EXIF 資訊。
- 將映像檔案上傳至 Amazon S3 儲存貯體。
- 使用 Amazon Rekognition 識別檔案中的三個主要屬性 (標籤)。
- 將 EXIF 和標籤資訊新增至區域中的 Amazon DynamoDB 資料表。

Rust

適用於 Rust 的 SDK

從 JPG、JPEG 或 PNG 檔案獲取 EXIF 資訊，將映像檔案上傳至 Amazon S3 儲存貯體，使用 Amazon Rekognition 識別三個主要屬性 (Amazon Rekognition 中的標籤)，然後將 EXIF 和標籤資訊新增至區域中的 Amazon DynamoDB 資料表中。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- DynamoDB
- Amazon Rekognition
- Amazon S3

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 S3 物件 Lambda 為您的應用程式轉換資料

下列程式碼範例示範如何使用 S3 物件 Lambda 轉換應用程式的資料。

.NET

AWS SDK for .NET

示範如何將自訂程式碼新增至標準 S3 GET 請求，以修改從 S3 擷取的要求物件，讓物件符合要求用戶端或應用程式的需求。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Lambda
- Amazon S3

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

疑難排解

本節說明如何針對 Amazon S3 功能進行疑難排解，並說明如何取得您聯絡 AWS Support 時所需的請求 ID。

主題

針對 Amazon S3 中的拒絕存取 (403 禁止) 錯誤進行疑難排解

Important

我們於 2024 年 5 月 13 日開始部署變更，以免除非值區擁有者發起的未經授權要求的費用。完成此變更部署後，如果要求是從個別 AWS 帳戶或 AWS 組織以外的地方起始，儲存貯體擁有者將永遠不會針對傳回 `AccessDenied` (HTTP`403 Forbidden`) 錯誤的要求產生要求或頻寬費用。如需完整的 HTTP 清單3XX和不會計費的4XX狀態碼清單的詳細資訊，請參閱Amazon S3 錯誤回應的帳單。此帳單變更不需要更新您的應用程式，並適用於所有 S3 儲存貯體。當這項變更的部署完成時 AWS 區域，我們會更新我們的文件。

下列主題涵蓋 Amazon S3 中拒絕存取 (403 禁止) 錯誤的最常見原因。

Note

對於 `Access Denied` (HTTP`403 Forbidden`)，當請求在儲存貯體擁有者的個別 AWS 帳戶或儲存貯體擁有者的 AWS 組織外部啟動時，S3 不會向儲存貯體擁有者收取費用。

主題

> **Note**
>
> 如果您嘗試對許可問題進行故障診斷，請從儲存貯體政策和 IAM 政策一節開始，且務必遵循檢查許可的秘訣中的指引。

儲存貯體政策與 IAM 政策

儲存貯體層級操作

如果沒有儲存貯體政策，則值區會隱含地允許來自儲存貯體擁有帳戶中任何 AWS Identity and Access Management (IAM) 身分的要求。儲存貯體也會隱含地拒絕來自任何其他帳戶中任何其他 IAM 身分的請求，以及匿名 (未簽署) 請求。不過，如果沒有適當的 IAM 使用者政策，則會隱含地拒絕請求者 (除非他們是根使用者) 提出任何請求。如需評估邏輯的詳細資訊，請參閱《IAM 使用者指南》中的判斷允許還是拒絕帳戶內的請求。

物件層級操作

如果物件是儲存貯體擁有帳戶所擁有，則儲存貯體政策和 IAM 使用者政策的運作方式與物件層級操作的運作方式相同，因為它們都適用於物件層級操作。例如，如果沒有適當的儲存貯體政策，則儲存貯體會隱含地允許來自儲存貯體擁有帳戶中任何 IAM 身分的物件請求。儲存貯體也會隱含地拒絕來自任何其他帳戶中任何其他 IAM 身分的物件請求，以及匿名 (未簽署) 請求。不過，如果沒有適當的 IAM 使用者政策，則會隱含地拒絕請求者 (除非他們是根使用者) 提出任何物件請求。

如果物件是由外部帳戶擁有，則只能透過物件存取控制清單 (ACL) 授予物件的存取權。儲存貯體政策和 IAM 使用者政策仍可用來拒絕物件請求。

因此，若要確保您的儲存貯體政策或 IAM 使用者政策不會造成「拒絕存取」(403 禁止) 錯誤，請確定符合下列需求：

- 對於同一帳戶存取，在儲存貯體政策或 IAM 使用者政策中，對您嘗試授予其許可的請求者不得有明確的 Deny 陳述式。如果您只想要使用儲存貯體政策和 IAM 使用者政策授予許可，則其中一項政策中至少須有一個明確的 Allow 陳述式。
- 對於跨帳戶存取權，在儲存貯體政策或 IAM 使用者政策中，對您嘗試授予其許可的請求者不得有明確的 Deny 陳述式。如果您只想要使用儲存貯體政策和 IAM 使用者政策來授予跨帳戶許可，則請求者的儲存貯體政策和 IAM 使用者政策都必須包含明確的 Allow 陳述式。

> **Note**
>
> 儲存貯體政策中的 Allow 陳述式僅適用於同一儲存貯體擁有帳戶所擁有的物件。不過，儲存貯體政策中的 Deny 陳述式會套用至所有物件，不論物件擁有權為何。

檢視或編輯儲存貯體政策

> **Note**
>
> 若要檢視或編輯儲存貯體政策，您必須具備 s3:GetBucketPolicy 許可。

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 從儲存貯體清單中，選擇要檢視或建立其儲存貯體政策的儲存貯體名稱。
4. 選擇許可索引標籤標籤。
5. 在 Bucket policy (儲存貯體政策) 下方，選擇 Edit (編輯)。Edit bucket policy (編輯儲存貯體政策) 頁面隨即出現。

若要使用 AWS Command Line Interface (AWS CLI) 檢閱或編輯值區政策，請使用get-bucket-policy指令。

> **Note**
>
> 如果您因為儲存貯體政策不正確而遭到鎖定，請使用 root AWS Management Console 使用者認證登入。若要重新取得儲存貯體的存取權，請務必使用根使用者憑證刪除儲存貯體政策。

檢查許可的秘訣

若要檢查請求者是否具有適當的許可來執行 Amazon S3 操作，請嘗試以下動作：

- 識別請求者。如果它是未簽署的請求，則為沒有 IAM 使用者政策的匿名請求。如果它是使用預先簽署 URL 的請求，則使用者政策將與已簽署請求之 IAM 使用者或角色的使用者政策相同。
- 確認您使用的是正確的 IAM 使用者或角色。您可以檢查 AWS Management Console 的右上角或使用 aws sts get-caller-identity 命令來驗證 IAM 使用者或角色。
- 檢查與 IAM 使用者或角色相關的 IAM 政策。您可以使用下列其中一種方法：
  - 使用 IAM 政策模擬器測試 IAM 政策。
  - 檢閱不同的 IAM 政策類型。
- 如有需要，請編輯您的 IAM 使用者政策。
- 請檢閱下列明確拒絕或允許存取的政策範例：
  - 明確允許 IAM 使用者政策：IAM：以程式設計方式和在主控台中允許和拒絕對多個服務的存取
  - 明確允許儲存貯體政策：將許可授予多個帳戶，以上傳物件或設定物件 ACL 進行公開存取
  - 明確拒絕 IAM 使用者政策AWS:: AWS 根據要求拒絕存取 AWS 區域
  - 明確拒絕儲存貯體政策：寫入儲存貯體的所有物件都需要 SSE-KMS

Amazon S3 ACL 設定

檢查 ACL 設定時，請先檢閱物件擁有權設定，以檢查儲存貯體上是否已啟用 ACL。請注意，ACL 許可只能用來授予許可，而且無法用來拒絕請求。ACL 也無法用來將存取權授予儲存貯體政策或 IAM 使用者政策中明確拒絕的請求者。

「物件擁有權」會設定為套用儲存貯體擁有者強制執行。

如果已啟用儲存貯體擁有者強制執行設定，則 ACL 設定不太可能造成「拒絕存取 (403 禁止)」錯誤，因為此設定會停用所有套用至儲存貯體和物件的 ACL。儲存貯體擁有者強制執行是 Amazon S3 儲存貯體的預設 (和建議) 設定。

「物件擁有權」設定會設定為儲存貯體擁有者偏好或物件寫入者

ACL 許可仍適用於儲存貯體擁有者偏好設定或物件寫入者設定。有兩種 ACL：儲存貯體 ACL 和物件 ACL。如需這兩種 ACL 類型之間的差異，請參閱 ACL 許可與存取政策許可的對應。

根據遭拒請求的動作，檢查儲存貯體或物件的 ACL 許可：

- 如果 Amazon S3 拒絕了`LIST`、`PUT` 物件、`GetBucketAcl` 或 `PutBucketAcl` 請求，請檢閱儲存貯體的 ACL 許可。

  > **Note**
  >
  > 您無法使用儲存貯體 ACL 設定授予 `GET` 物件許可。

- 如果 Amazon S3 拒絕了 S3 物件上的 `GET` 請求，或 PutObjectAcl 請求，請檢閱物件的 ACL 許可。

  > **Important**
  >
  > 如果擁有物件的帳戶與擁有儲存貯體的帳戶不同，則儲存貯體政策不會控制物件的存取。

針對跨帳戶物件擁有權期間來自 **GET** 物件請求的「拒絕存取 (403 禁止)」錯誤進行疑難排解

檢閱儲存貯體的物件擁有權設定，以判斷物件擁有者。如果您有權存取物件 ACL，也可以檢查物件擁有者的帳戶。(若要檢視物件擁有者的帳戶，請檢閱 Amazon S3 主控台中的物件 ACL 設定。) 或者，您也可以提出 `GetObjectAcl` 請求以尋找物件擁有者的正式 ID，以驗證物件擁有者帳戶。根據預設，ACL 會將 `GET` 請求的明確允許許可授予物件擁有者的帳戶。

在確認了物件擁有者與儲存貯體擁有者不同之後，請根據您的使用案例和存取層級，選擇下列其中一種方法來協助解決「拒絕存取 (403 禁止)」錯誤：

- 停用 ACL (建議) - 此方法將套用至所有物件，並可由儲存貯體擁有者執行。此方法會自動給與儲存貯體擁有者擁有權，並讓其完全控制儲存貯體中的每個物件。實作此方法之前，請檢查停用 ACL 的先決條件。如需如何將儲存貯體設定為儲存貯體擁有者強制執行 (建議) 模式的相關資訊，請參閱在現有儲存貯體上設定物件擁有權。

> **Important**
>
> 若要避免發生「拒絕存取 (403 禁止)」錯誤，請務必先將 ACL 許可遷移至儲存貯體政策，然後再停用 ACL。如需詳細資訊，請參閱從 ACL 許可遷移的儲存貯體政策範例。

- 將物件擁有者變更為儲存貯體擁有者 - 此方法可套用至個別物件，但只有物件擁有者 (或具有適當許可的使用者) 才能變更物件的擁有權。可能需要支付額外的 PUT 費用。(如需詳細資訊，請參閱 Amazon S3 定價。) 此方法授予儲存貯體擁有者物件的完整擁有權，允許儲存貯體擁有者透過儲存貯體政策控制物件的存取。

  若要變更物件的擁有權，請執行下列其中一個動作：

  - 您 (儲存貯體擁有者) 可將物件複製回儲存貯體。
  - 您可以將儲存貯體的「物件擁有權」設定變更為儲存貯體擁有者偏好。如果停用版本控制，則會覆寫儲存貯體中的物件。如果啟用了版本控制，則相同物件的重複版本將會出現在儲存貯體中，而儲存貯體擁有者可以將生命週期規則設定為過期。如需如何變更「物件擁有權」設定的指示，請參閱設定現有儲存貯體的「物件擁有權」。

    > **Note**
    >
    > 當您將「物件擁有權」設定更新為儲存貯體擁有者偏好時，此設定只會套用至上載至儲存貯體的新物件。

  - 您可以使用 `bucket-owner-full-control` 標準物件 ACL 讓物件擁有者重新上傳物件。

> **Note**
>
> 對於跨帳戶上傳，您也可以在儲存貯體政策中要求 `bucket-owner-full-control` 標準物件 ACL。如需儲存貯體政策範例，請參閱授予跨帳戶許可，以在確保儲存貯體擁有者具有完全控制時上傳物件。

- 將物件寫入者保留為物件擁有者 - 此方法不會變更物件擁有者，但可讓您個別授予物件的存取權。若要授予物件的存取權，您必須具有該物件的 `PutObjectAcl` 許可。然後，若要修正「拒絕存取 (403 禁止)」錯誤，請將請求者新增為被授予者，以存取物件 ACL 中的物件。如需詳細資訊，請參閱 設定 ACL。

S3 封鎖公開存取設定

如果失敗的請求涉及公開存取或公開政策，請檢查帳戶、儲存貯體或 S3 存取點上的 S3 封鎖公開存取設定。從 2023 年 4 月起，新儲存貯體的所有封鎖公開存取設定預設為啟用。如需 Amazon S3 如何定義「公開」的詳細資訊，請參閱 「公有」的意義。

設定為 TRUE 時，「封鎖公開存取」設定會作為明確拒絕政策，覆寫 ACL、儲存貯體政策和 IAM 使用者政策所允許的許可。若要判斷您的「封鎖公開存取」設定是否拒絕您的請求，請檢閱下列案例：

- 如果指定的存取控制清單 (ACL) 為公有，則 `BlockPublicAcls` 設定會拒絕您的 `PutBucketAcl` 和 `PutObjectACL` 呼叫。
- 如果請求包含公有 ACL，則 `BlockPublicAcls` 設定會拒絕您的 `PutObject` 呼叫。
- 如果 `BlockPublicAcls` 設定套用至帳戶，且請求包含公有 ACL，則任何包含公有 ACL 的 `CreateBucket` 呼叫都會失敗。
- 如果請求的許可僅由公有 ACL 授予，則 `IgnorePublicAcls` 設定會拒絕該請求。
- 如果指定的儲存貯體政策允許公開存取，則 `BlockPublicPolicy` 設定會拒絕您的 `PutBucketPolicy` 呼叫。
- 如果將 `BlockPublicPolicy` 設定套用至存取點，則指定公有政策且透過存取點進行的所有 `PutAccessPointPolicy` 和 `PutBucketPolicy` 呼叫都會失敗。
- 如果存取點或值區具有公用 AWS 服務 原則，則此`RestrictPublicBuckets`設定會拒絕除主體以外的所有跨帳戶呼叫。此設定也會拒絕所有匿名 (或未簽署) 呼叫。

若要檢閱和更新您的「封鎖公開存取」設定組態，請參閱 為您的 S3 儲存貯體設定封鎖公開存取。

Amazon S3 加密設定

Amazon S3 支援儲存貯體上的伺服器端加密。伺服器端加密是指接收資料的應用程式或服務在目的地將資料加密。Amazon S3 會在物件層級將資料寫入資料中心的磁碟時加密，並在您存取 AWS 資料時為您解密。

依預設，Amazon S3 現在將伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 套用為 Amazon S3 中每個儲存貯體的基本加密層級。Amazon S3 也可讓您在上傳物件時指定伺服器端加密方法。

檢閱儲存貯體的伺服器端加密狀態和加密設定

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 從儲存貯體清單中，選擇要檢查其加密設定的儲存貯體。
4. 選擇屬性索引標籤。
5. 向下捲動至預設加密區段，然後檢視加密類型設定。

若要使用檢查您的加密設定 AWS CLI，請使用get-bucket-encryption指令。

檢查物件的加密狀態

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 從儲存貯體清單中，選擇包含該物件的儲存貯體名稱。
4. 從物件清單中，選擇您想要新增或變更其加密的物件名稱。

   物件的詳細資訊頁面隨即出現。
5. 向下捲動至伺服器端加密設定區段，以檢視物件的伺服器端加密設定。

若要使用檢查物件加密狀態 AWS CLI，請使用head-object指令。

加密和許可需求

Amazon S3 支援三種類型的伺服器端加密：

- 使用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密
- 使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 進行伺服器端加密
- 使用客戶提供金鑰 (SSE-C) 的伺服器端加密

根據您的加密設定，確定符合下列許可需求：

- SSE-S3 - 不需要額外的許可。
- SSE-KMS (搭配客戶受管金鑰) - 若要上傳物件，需要 AWS KMS key 上的 `kms:GenerateDataKey` 權限。若要下載物件並執行物件的分段上傳，需要 KMS 金鑰上的 `kms:Decrypt` 許可。
- SSE-KMS (使用 AWS 受管金鑰) — 請求者必須來自擁有 KMS 金鑰的相同帳戶。`aws/s3`請求者亦須具有正確的 Amazon S3 許可才能存取物件。

- SSE-C (搭配客戶提供的金鑰) - 不需要其他許可。您可以設定儲存貯體政策，針對儲存貯體中的物件使用客戶提供的加密金鑰來要求和限制伺服器端加密。

如果物件使用客戶受管金鑰加密，請確定 KMS 金鑰政策允許您執行 `kms:GenerateDataKey` 或 `kms:Decrypt` 動作。如需檢查 KMS 金鑰政策的指示，請參閱《AWS Key Management Service 開發人員指南》中的檢視金鑰政策。

S3 物件鎖定設定

如果您的儲存貯體已啟用 S3 物件鎖定，且物件受到保留期或法務保存的保護，則在您嘗試刪除物件時，Amazon S3 會傳回「拒絕存取 (403 禁止)」錯誤。

檢查儲存貯體是否已啟用物件鎖定

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 從儲存貯體清單中，選擇您要檢閱的儲存貯體名稱。
4. 選擇屬性索引標籤。
5. 向下捲動至物件鎖定區段。確認物件鎖定設定為已啟用或已停用。

若要判斷物件是否受到保留期或法務保存保護，請檢視物件的鎖定資訊。

如果物件受到保留期或法務保存保護，請檢查下列情況：

- 如果物件版本受到合規保留模式保護，則沒有任何方式可將其永久刪除。來自任何請求者 (包括根使用者) 的永久 `DELETE` 請求都會導致「拒絕存取 (403 禁止)」錯誤。此外，請注意，當您針對受合規保留模式保護的物件提交 `DELETE` 請求時，Amazon S3 會為該物件建立刪除標記。
- 如果物件版本受控管保留模式保護，且您具有 `s3:BypassGovernanceRetention` 許可，您可以略過保護並永久刪除版本。如需詳細資訊，請參閱繞過控管模式。
- 如果物件版本受法務保存保護，則永久 `DELETE` 請求可能會導致「拒絕存取 (403 禁止)」錯誤。若要永久刪除物件版本，您必須移除物件版本上的法務保存。若要移除法務保存，您必須具有 `s3:PutObjectLegalHold` 許可。如需移除法務保存的詳細資訊，請參閱 設定 S3 物件鎖定。

VPC 端點政策

如果您使用虛擬私有雲端 (VPC) 端點來存取 Amazon S3，請確定 VPC 端點政策不會封鎖您存取 Amazon S3 資源。依預設，VPC 端點政策允許所有對 Amazon S3 的請求。您也可以設定 VPC 端點政策來限制特定請求。如需如何檢查 VPC 端點政策的詳細資訊，請參閱《AWS PrivateLink 指南》中的使用端點政策控制對 VPC 端點的存取。

AWS Organizations 政策

如果您 AWS 帳戶 屬於某個組織， AWS Organizations 政策可能會阻止您存取 Amazon S3 資源。根據預設， AWS Organizations 政策不會封鎖任何向 Amazon S3 發出的請求。不過，請確定您的 AWS Organizations 政策尚未設定為封鎖 S3 儲存貯體的存取。如需如何檢查 AWS Organizations 政策的指示，請參閱《AWS Organizations 使用指南》中的「列出所有政策」。

存取點設定

如果您在透過 Amazon S3 存取點提出請求時收到「拒絕存取 (403 禁止)」錯誤，則可能需要檢查下列事項：

- 存取點的組態
- 用於存取點的 IAM 使用者政策
- 用來管理或設定跨帳戶存取點的儲存貯體政策

存取點組態與政策

- 建立存取點時，您可以選擇將網際網路或 VPC 指定為網路原點。如果網路原點設定為僅限 VPC，Amazon S3 將拒絕對非來自指定 VPC 的存取點所發出的任何請求。若要檢查存取點的網路原點，請參閱 建立受限於 Virtual Private Cloud 的存取點。
- 使用存取點，您也可以設定自訂的「封鎖公開存取」設定，其運作方式與儲存貯體或帳戶層級的「封鎖公開存取」設定類似。若要檢查您的自訂「封鎖公開存取」設定，請參閱 管理存取點的公開存取。
- 若要使用存取點對 Amazon S3 提出成功的請求，請確保請求者具有必要的 IAM 許可。如需詳細資訊，請參閱 配置使用存取點的 IAM 原則。
- 如果請求涉及跨帳戶存取點，請確定儲存貯體擁有者已更新儲存貯體政策，以授權來自存取點的請求。如需詳細資訊，請參閱 授予跨帳戶存取點的許可。

如果檢查本主題中的所有項目後，拒絕存取 (403 禁止) 錯誤仍然存在，請擷取您的 Amazon S3 請求 ID 並聯絡 AWS Support 以取得其他指引。

針對批次操作進行疑難排解

下列主題列出常見錯誤，協助您針對「批次操作」期間您可能遇到的問題進行疑難排解。

常見錯誤

- 有許可問題或「S3 物件鎖定」保留模式為啟用狀態時，不會交付任務報告
- 「S3 批次複寫」失敗並顯示錯誤：產生清單檔案時找不到符合篩選條件的金鑰
- 將新的複寫規則新增至現有複寫組態之後，發生「批次操作」失敗
- Batch 操作失敗的對象，錯誤 400 InvalidRequest：任務失敗，由於缺失 VersionId
- 在啟用任務標籤選項的情況下建立任務失敗
- 存取遭拒無法讀取清單檔案

有許可問題或「S3 物件鎖定」保留模式為啟用狀態時，不會交付任務報告

若目的地儲存貯體上缺少必要的許可，或「物件鎖定」保留模式 (控管模式或合規模式) 為啟用狀態，則會發生下列錯誤。

錯誤：失敗的原因。任務報告無法寫入您的報告儲存貯體。請檢查您的許可。

IAM 角色和信任政策必須設定為允許 S3 Batch Operations 存取作為報告傳遞目標的儲存貯體中的 PUT 物件。如果缺少這些必要許可，任務報告傳遞就會失敗。

啟用保留模式時，值區會受到 write-once-read-many (WORM) 保護。不支援在目的地儲存貯體上使用「物件鎖定」並啟用保留模式，否則任務完成報告傳遞嘗試會失敗。若要修正此問題，請為未啟用「物件鎖定」保留模式的任務完成報告選擇目的地儲存貯體。

「S3 批次複寫」失敗並顯示錯誤：產生清單檔案時找不到符合篩選條件的金鑰

錯誤：產生清單檔案時找不到符合篩選條件的金鑰。

下列其中一個原因會發生此錯誤：

- 當來源儲存貯體中的物件存放在 S3 Glacier 彈性擷取或 S3 Glacier Deep Archive 儲存類別中時。

若要在這些物件上使用「批次複寫」，請先在「批次操作」任務中使用「S3 啟動還原物件」操作，將它們還原到「S3 標準」儲存體類別。如需詳細資訊，請參閱 還原已封存的物件 和還原物件 (批次作業)。在還原了物件之後，您可以使用「批次複寫」任務來複寫它們。

- 當提供的篩選條件與來源值區中的任何有效物件不相符時。

  驗證並更正篩選條件。例如，在「Batch 複製」規則中，篩選器準則會尋找*##-s3-bucket* 中具有前置詞的所有物件。Tax/如果輸入的前綴名稱不正確，在開頭和結尾都有斜線，/Tax/而不是僅在結尾輸入，則找不到 S3 物件。若要解決錯誤，請在此情況下，在複製規則Tax/中從/Tax/到更正前置詞。

將新的複寫規則新增至現有複寫組態之後，發生「批次操作」失敗

「批次操作」會嘗試針對來源儲存貯體的複寫組態中的每個規則執行現有的物件複寫。如果任何現有的複寫規則發生問題，可能會發生失敗。

「批次操作」任務的完成報告會說明任務失敗原因。如需常見錯誤的清單，請參閱 Amazon S3 複寫失敗原因。

Batch 操作失敗的對象，錯誤 400 InvalidRequest：任務失敗，由於缺失 VersionId

如果「批次操作」任務正在對版本控制的儲存貯體中的物件執行動作，並在清單檔案遇到含有空白版本 ID 欄位的物件，則會發生下列範例錯誤。

錯誤：*BUCKET_NAME###/#####*失敗，400，工作因遺失而失敗 InvalidRequest VersionId

發生這個錯誤的原因是清單檔案中的版本 ID 欄位是空字串，而不是常值 `null` 字串。

該特定物件或多個物件「批次操作」將會失敗，但整個任務不會失敗。如果清單檔案格式設定為在操作期間使用版本 ID，則會發生這個問題。非版本控制的任務不會遇到此問題，因為它們只在每個物件的最新版本上操作，且會忽略清單檔案中的版本 ID。

若要修正此問題，請將空白版本 ID 轉換為 `null` 字串。如需詳細資訊，請參閱 the section called “將空白版本 ID 字串轉換為空字串”。

在啟用任務標籤選項的情況下建立任務失敗

若沒有 `s3:PutJobTagging` 許可，在啟用任務標籤選項的情況下建立「批次操作」任務會導致 `403 access denied` 錯誤。

若要在啟用工作標籤選項的情況下建立 Batch 作業任務，建立 Batch 作業工作的 AWS Identity and Access Management (IAM) 使用者除了`s3:PutJobTagging`權限外，還必須具有`s3:CreateJob`權限。

如需「批次操作」所需許可的詳細資訊，請參閱 the section called “授予許可”。

存取遭拒無法讀取清單檔案

當您嘗試建立「批次操作」任務時，如果「批次操作」無法讀取清單檔案，則可能會發生下列錯誤。

AWS CLI

失敗的原因禁止讀取清單： AccessDenied

Amazon S3 主控台

警告：無法取得清單檔案物件的 ETag。指定不同的物件以繼續。

若要解決此問題，請執行下列動作：

- 確認您用來建立 Batch 作業工作的 IAM 角色具有`s3:GetObject`權限。 AWS 帳戶 帳戶的 IAM 角色必須具有 `s3:GetObject` 許可，才能允許「批次操作」讀取資訊清單檔案。

  如需「批次操作」所需許可的詳細資訊，請參閱 the section called “授予許可”。
- 檢查清單檔案物件的中繼資料是否有任何存取與「S3 物件擁有權」不符。如需「S3 物件擁有權」的詳細資訊，請參閱 the section called “控制物件所有權”。
- 檢查是否使用 AWS Key Management Service (AWS KMS) 金鑰來加密資訊清單檔案。

  Batch 作業支援 AWS KMS已加密的 CSV 庫存報告。不過，Batch 作業不支援 AWS KMS已加密的 CSV 資訊清單檔案。如需更多詳細資訊，請參閱 設定 Amazon S3 清查 及 指定資訊清單。

針對 CORS 進行疑難排解

如果使用 CORS 組態存取儲存貯體時，遇到意外行為，請嘗試以下步驟進行疑難排解：

1. 確認已在儲存貯體上設定了 CORS 組態。

   若 CORS 組態已設定，主控台會在 Properties (屬性) 儲存貯體的 Permissions (許可) 區段中，顯示 Edit CORS Configuration (編輯 CORS 組態) 的連結。

2. 使用您偏好的工具擷取完整的要求及回應。Amazon S3 接收到的每一個要求都必須與您的要求中的資料相符的 CORS 規則，如下列所示：

   a. 確認要求有 Origin 的標頭。

      若沒有該標頭，Amazon S3 不會將要求視為跨來源要求，也不會在回應中傳送 CORS 回應標頭。

   b. 確認要求中的 Origin 標頭至少符合特定的 `AllowedOrigin` 中一個 `CORSRule` 元素。

      Origin 要求標頭中的配置、主機及連接埠數值，都必須符合 `AllowedOrigin` 中的 `CORSRule`元素。例如，若將 `CORSRule` 設為允許來源 `http://www.example.com`，則要求中的 `https://www.example.com` 及 `http://www.example.com:80` 來源，都不符合您組態中允許的來源。

   c. 確認要求中的方法 (或為預檢要求中，則為 `Access-Control-Request-Method` 中指定的方法) 為相同 `AllowedMethod` 中的 `CORSRule` 元素。

   d. 若是預檢要求，如果要求包含 `Access-Control-Request-Headers` 標頭，則請確認 `CORSRule` 對於 `AllowedHeader` 標頭中的每個值，都包含了 `Access-Control-Request-Headers` 項目。

針對 Amazon S3 生命週期問題進行疑難排解

以下資訊有助於針對 Amazon S3 生命週期規則的常見問題進行疑難排解。

主題

我在儲存貯體上執行了清單操作，並看到以為已逾期或由生命週期規則轉換的物件。

S3 生命週期物件轉換和物件逾期為非同步操作。因此，在物件符合過期或轉移資格的時間與其實際轉換或過期的時間之間，可能會有延遲。只要滿足生命週期規則，就會套用計費，即使動作尚未完成也是一樣。此行為的例外狀況為，如果您擁有轉換至 S3 Intelligent-Tiering 儲存類別的生命週期規則集。在此情況下，直到物件轉換至 S3 Intelligent-Tiering 儲存類別後，才會發生計費變更。如需計費中變更的詳細資訊，請參閱在儲存貯體上設定生命週期組態。

> **Note**
>
> Amazon S3 不會將小於 128 KB 的物件從「S3 標準」或「S3 標準 - IA」儲存體類別轉換為「S3 智慧型分層」、「S3 標準 - IA」或「S3 單區域 - IA」儲存體類別。

如何監視生命週期規則所採取的動作？

欲監視生命週期規則所採取的動作，您可以使用下列功能：

- S3 事件通知 — 您可以設定 S3 事件通知，以便在任何 S3 生命週期到期或轉換事件時收到通知。
- S3 伺服器存取日誌 — 您可以為 S3 儲存貯體啟用伺服器存取日誌，以擷取 S3 生命週期動作，例如物件轉換到另一個儲存類別或物件過期。如需詳細資訊，請參閱生命週期和記錄。

若要每天檢視由生命週期動作造成的儲存變更，建議您使用 S3 儲存鏡頭儀表板，而不要使用 Amazon CloudWatch 指標。在 Storage Lens 儀表板中，您可以檢視下列監控物件計數或大小的指標：

- 目前版本位元組
- 目前版本物件計數
- 非目前版本位元組
- 非目前版本物件計數
- 刪除標記物件計數
- 刪除標記儲存體位元組
- 未完成分段上傳位元組
- 未完成分段上傳物件計數

即使在啟用版本控制的儲存貯體上設定生命週期規則之後，我的 S3 物件計數仍會增加。

在啟用版本控制的值區中，當物件到期時，物件不會從值區中完全刪除。相反地，建立刪除標記作為物件的最新版本。刪除標記仍會計為物件。因此，如果建立生命週期規則，僅使目前版本過期，則 S3 儲存貯體中的物件計數實際上會增加，而不是下降。

例如，假設 S3 儲存貯體已啟用 100 個物件的版本控制，而生命週期規則設定為使物件的目前版本在 7 天後過期。第七天之後，物件計數會增加到 200，因為除了 100 個原始物件 (現在為非目前版本) 之外，還建立了 100 個刪除標記。如需啟用版本控制的儲存貯體之「S3 生命週期」組態規則動作的詳細資訊，請參閱在儲存貯體上設定生命週期組態。

若要永久移除物件，請新增其他生命週期組態，以刪除先前版本的物件、過期的刪除標記，以及未完成的分段上傳。如需如何建立新生命週期規則的指示，請參閱在儲存貯體上設定生命週期組態。

> **Note**
>
> - Amazon S3 會將物件的轉換或過期日期四捨五入到隔天的午夜 UTC。
>
> 評估物件的生命週期動作時，Amazon S3 會使用 UTC 的物件建立時間。例如，假設一個具有生命週期規則的非版本化值區，該規則設定為在一天後到期物件。假設物件是在太平洋夏令時間 (PDT) 1 月 1 日 17:05 建立的，該時間對應於世界標準時間 1 月 2 日 00:05。該物件在 1 月 3 日 00:05 UTC 變成一天的舊時間，這使得當 S3 生命週期在 1 月 4 日 UTC 00:00 評估物件時符合到期的資格。
>
> 由於 Amazon S3 生命週期動作是非同步發生的，因此在生命週期規則中指定的日期和物件的實際實際轉換之間可能會有些許延遲。如需詳細資訊，請參閱轉換或到期延遲。
>
> 如需詳細資訊，請參閱生命週期規則：依據物件的存在時間。
> - 對於受「物件鎖定」保護的 S3 物件，不會永久刪除目前的版本。相反地，會將刪除標記新增至物件，使其成為非目前版本。然後會保留非目前版本，且永久不會過期。

如何使用生命週期規則清空 S3 儲存貯體？

S3 生命週期規則是清空 S3 儲存貯體與其數百萬個物件的有效工具。若要從 S3 儲存貯體刪除大量物件，請務必使用下列兩對生命週期規則：

- 使物件的目前版本到期和永久刪除物件的先前版本

- 刪除過期的刪除標記和刪除未完成的分段上傳

如需如何建立生命週期組態規則的步驟，請參閱在儲存貯體上設定生命週期組態。

> **Note**
>
> 對於受「物件鎖定」保護的 S3 物件，不會永久刪除目前的版本。相反地，會將刪除標記新增至物件，使其成為非目前版本。然後會保留非目前版本，且永久不會過期。

將物件轉換為成本較低的儲存體類別之後，我的 Amazon S3 計費增加了。

有數個原因，在將物件轉換為成本較低的儲存體類別之後，您的帳單可能會增加：

- 小型物件的 S3 Glacier 額外費用

  對於轉換為 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 的每個物件，總共有額外的 40 KB 與此儲存體更新相關聯。作為額外 40 KB 的一部分，8 KB 用來存放中繼資料和物件的名稱。此 8 KB 是根據「S3 標準」費率收費。剩餘的 32 KB 用於索引編製和相關中繼資料。此 32 KB 是根據 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 定價收費。

  因此，如果您要儲存許多小型物件，我們不建議使用生命週期轉換。相反地，若要降低任何額外費用，請將多個小型物件彙總為少量的大型物件，然後再將它們存放在 Amazon S3 中。如需成本考量的詳細資訊，請參閱轉換為 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 儲存體類別 (物件封存)。

- 最低儲存費用

  某些 S3 儲存體類別有最短儲存持續時間需求。在滿足最短持續時間之前，從這些類別中刪除、覆寫或轉換的物件，會被收取按比例分配的提前轉換或刪除費用。這些最短儲存持續時間需求如下：

  - S3 標準 - IA 和 S3 單區域 - IA - 30 天
  - S3 Glacier Flexible Retrieval 和 S3 Glacier Instant Retrieval - 90 天
  - S3 Glacier Deep Archive - 180 天

  如需這些需求的詳細資訊，請參閱使用 S3 生命週期轉移物件的限制一節。如需一般 S3 定價資訊，請參閱 Amazon S3 定價和 AWS 定價計算機。

- 生命週期轉換成本

每次透過生命週期規則將物件轉換為不同的儲存體類別時，Amazon S3 就會將該轉換計為一個轉換請求。這些轉換請求的成本不包括在這些儲存體類別的成本。如果打算轉換大量物件，建議在轉換為較低成本方案時將請求成本納入考慮。如需詳細資訊，請參閱 Simple Storage Service (Amazon S3) 定價。

我已更新儲存貯體政策，但我的 S3 物件仍被過期的生命週期規則刪除。

儲存貯體政策中的 Deny 陳述式不會防止生命週期規則中定義的物件過期。生命週期動作 (例如轉換或過期) 不會使用 S3 DeleteObject 操作。相反地，使用內部 S3 端點來執行「S3 生命週期」動作。(如需詳細資訊，請參閱生命週期和記錄。)

若要防止生命週期規則採取任何動作，您必須編輯、刪除或停用規則。

是否可以復原「S3 生命週期」規則使其過期的 S3 物件？

若要復原由「S3 生命週期」使其過期的物件，唯一方法就是透過版本控制，而在物件符合過期資格之前，版本控制必須已就位。您無法復原生命週期規則所執行的過期操作。如果物件是由既定的「S3 生命週期」規則永久刪除，則無法復原這些物件。若要在儲存貯體上啟用版本控制，請參閱 the section called “使用 S3 版本控制”。

如果您已將版本控制套用至儲存貯體，且物件的非目前版本仍保持完整不變，則可以還原過期物件的先前版本。如需「S3 生命週期」規則動作之行為和版本控制狀態的詳細資訊，請參閱描述生命週期動作的元素中的生命週期動作和儲存貯體版本控制狀態表格。

> **Note**
>
> 如果 S3 儲存貯體受 AWS Backup 或 S3 複寫保護，您也可以使用這些功能來復原過期的物件。

故障排除複寫

本節列出「Amazon S3 複寫」的疑難排解提示，以及「S3 批次複寫」錯誤的相關資訊。

主題

- S3 複寫的疑難排解提示
- 批次複寫錯誤

S3 複寫的疑難排解提示

在您設定複寫之後，如果物件複本未出現在目的地儲存貯體中，請使用這些故障診斷技巧以識別並修正問題。

- 大部分的物件會在 15 分鐘內複寫。Amazon S3 複寫物件所需的時間長短取決於幾個因素，包括來源和目的地區域對，以及物件的大小。若是大型物件，最長可能需要數小時的複寫時間。如需複寫時間的可見性，您可以使用 S3 複寫時間控制 (S3 RTC)。

  如果要複寫的物件很大，請等待一段時間，然後查看其是否出現在目的地中。您也可以檢查來源物件的複寫狀態。如果物件複寫狀態為 PENDING，表示 Amazon S3 尚未完成複寫。如果物件複寫狀態為 FAILED，請檢查來源儲存貯體中所設的複寫組態。此外，若要在複寫期間接收失敗的相關資訊，您可以設定「Amazon S3 事件通知」複寫以接收失敗事件。如需詳細資訊，請參閱使用 Amazon S3 事件通知接收複寫失敗事件。
- 您可以呼叫 HeadObject API 操作，以檢查物件的複寫狀態。HeadObject API 操作會傳回物件的 PENDING、COMPLETED 或 FAILED 複寫狀態。在 HeadObject API 呼叫的回應時，會在 x-amz-replication-status 元素中傳回複寫狀態。

  > **Note**
  >
  > 若要執行 HeadObject，您必須對您正要請求的儲存貯體具有讀取權。HEAD 請求具有與 GET 請求相同的選項，而不需執行 GET 操作。例如，若要使用 AWS Command Line Interface (AWS CLI) 執行 HeadObject 請求，您可以執行下列命令。以您自己的資訊取代 *user input placeholders*。
  >
  > ```
  > aws s3api head-object --bucket my-bucket --key index.html
  > ```

- 在 HeadObject 傳回具有 FAILED 複寫狀態的物件之後，您可以使用「S3 批次複寫」來複寫這些失敗的物件。或者，您也可以將失敗的物件重新上傳至來源儲存貯體，這樣會啟動新物件的複寫。
- 在來源儲存貯體中的複寫組態中，驗證下列項目：
  - 目的地儲存貯體的 Amazon Resource Name (ARN) 正確。
  - 金鑰名稱前綴正確。例如，如果您設定組態來複寫具有前綴 Tax 的物件，則只會複寫具有 Tax/document1 或 Tax/document2 等金鑰名稱的物件。不會複寫具有金鑰名稱 document3 的物件。
  - 複寫規則的狀態為 Enabled。
- 在複寫組態中確認未在任何儲存貯體上暫停版本控制。來源與目的地儲存貯體都必須啟用版本控制。

- 如果將複寫規則設定為將物件擁有權變更為目的地儲存貯體擁有者，則用於複寫的 AWS Identity and Access Management (IAM) 角色必須具有 `s3:ObjectOwnerOverrideToBucketOwner` 許可。此許可是在資源 (在此情況下，指的是目的地儲存貯體) 上授予。例如，下列 `Resource` 陳述式說明如何在目的地儲存貯體上授予此許可：

```
{
  "Effect":"Allow",
  "Action":[
    "s3:ObjectOwnerOverrideToBucketOwner"
  ],
  "Resource":"arn:aws:s3:::DestinationBucket/*"
}
```

- 如果目的地儲存貯體是由另一個帳戶所擁有，則目的地儲存貯體的擁有者也須透過目的地儲存貯體政策，將 `s3:ObjectOwnerOverrideToBucketOwner` 許可授予來源儲存貯體擁有者。若要使用下列儲存貯體政策範例，請以您自己的資訊取代 *`user input placeholders`*：

```
{
  "Version": "2012-10-17",
  "Id": "Policy1644945280205",
  "Statement": [
    {
      "Sid": "Stmt1644945277847",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789101:role/s3-replication-role"
      },
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateTags",
        "s3:ObjectOwnerOverrideToBucketOwner"
      ],
      "Resource": "arn:aws:s3:::DestinationBucket/*"
    }
  ]
}
```

> **Note**
>
> 如果目的地儲存貯體的物件擁有權設定包含儲存貯體擁有者強制執行，則您不需要在複寫規則中更新設定，以將物件擁有權變更為目的地儲存貯體擁有者。依預設，物件擁有權變更將會發生。如需變更複本擁有權的詳細資訊，請參閱變更複本擁有者。

- 如果您在跨帳戶案例中設定複寫組態，其中來源和目的地儲存貯體屬於不同AWS 帳戶，則目的地儲存貯體無法設定為要求者付款值區。如需詳細資訊，請參閱 使用儲存體傳輸和用量的申請者付款儲存貯體。
- 如果儲存貯體的來源物件使用 AWS Key Management Service (AWS KMS) 金鑰進行加密，則必須將複寫規則設定為包含 AWS KMS 加密物件。請務必在 Amazon S3 主控台的加密設定下選取複寫使用 AWS KMS 加密的物件。然後，選取用於加密目的地物件的 AWS KMS 金鑰。

> **Note**
>
> 如果目的地儲存貯體位於不同的帳戶中，請指定目的地帳戶所擁有的 AWS KMS 客戶受管金鑰。請勿使用預設的 Amazon S3 受管金鑰 (aws/s3)。使用預設金鑰會搭配來源帳戶所擁有的 Amazon S3 受管金鑰加密物件，藉以防止物件與另一個帳戶共用。因此，目的地帳戶將無法存取目的地儲存貯體中的物件。

若要使用屬於目的地帳戶的 AWS KMS 金鑰來加密目的地物件，目的地帳戶必須將 `kms:GenerateDataKey` 和 `kms:Encrypt` 許可授予 KMS 金鑰政策中的複寫角色。若要在您的 KMS 金鑰政策中使用下列範例，請以您自己的資訊取代 *`user input placeholders`*：

```
{
    "Sid": "AllowS3ReplicationSourceRoleToUseTheKey",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::123456789101:role/s3-replication-role"
    },
    "Action": ["kms:GenerateDataKey", "kms:Encrypt"],
    "Resource": "*"
}
```

如果您對 AWS KMS 金鑰政策中的 `Resource` 陳述式使用星號 (*)，則此政策會將使用 KMS 金鑰的許可僅授予複寫角色。此政策不允許複寫角色提升其許可。

依預設，KMS 金鑰政策會授予根使用者金鑰的完整許可。這些許可可以委派給相同帳戶中的其他使用者。除非來源 KMS 金鑰政策中有 Deny 陳述式，否則使用 IAM 政策將複寫角色許可授予來源 KMS 金鑰就夠了。

> **Note**
>
> 限制存取特定 CIDR 範圍、VPC 端點或 S3 存取點的 KMS 金鑰政策可能會導致複寫失敗。

如果來源或目的地 KMS 金鑰根據加密內容授予許可，請確認已針對儲存貯體開啟「Amazon S3 儲存貯體金鑰」。如果儲存貯體已開啟「S3 儲存貯體」金鑰，則加密內容必須是儲存貯體層級資源，如下所示：

```
"kms:EncryptionContext:arn:aws:arn": [
     "arn:aws:s3:::SOURCE_BUCKET_NAME"
     ]
"kms:EncryptionContext:arn:aws:arn": [
     "arn:aws:s3:::DESTINATION_BUCKET_NAME"
     ]
```

除了 KMS 金鑰政策授予的許可之外，來源帳戶還必須將下列最低許可新增至複寫角色的 IAM 政策：

```
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": [
        "SourceKmsKeyArn"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "kms:GenerateDataKey",
        "kms:Encrypt"
    ],
    "Resource": [
```

```
        "DestinationKmsKeyArn"
    ]
}
```

如需如何複寫使用 AWS KMS 加密之物件的詳細資訊，請參閱複寫加密物件。

- 如果目的地儲存貯體的擁有者是另一個 AWS 帳戶，請確認儲存貯體擁有者在目的地儲存貯體上有儲存貯體政策，以允許來源儲存貯體擁有者複寫物件。如需範例，請參閱 當不同帳戶擁有來源與目的地儲存貯體時設定複寫。
- 如果您的物件在您驗證了許可之後仍未複寫，請檢查下列位置是否有任何明確的 Deny 陳述式：
  - 來源或目的地儲存貯體政策中的 Deny 陳述式。如果儲存貯體政策拒絕存取下列任何動作的複寫角色，則複寫會失敗：

    來源儲存貯體：

    ```
            "s3:GetReplicationConfiguration",
            "s3:ListBucket",
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
    ```

    目的地儲存貯體：

    ```
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
    ```

  - 附加至 IAM 角色的 Deny 陳述式或許可界限可能會導致複寫失敗。
  - 附加至來源或目的地帳戶之 AWS Organizations 服務控制政策中的 Deny 陳述式可能會導致複寫失敗。
- 如果目的地儲存貯體中未出現物件複本，可能是下列問題阻礙了複寫作業：
  - 如果來源儲存貯體中的物件是由另一個複寫組態所建立的複本，則 Amazon S3 不會複寫該複本。例如，如果您將複寫組態從儲存貯體 A 設定到儲存貯體 B，再設定到儲存貯體 C，則 Amazon S3 不會將儲存貯體 B 中的物件複本複寫至儲存貯體 C。

  - 來源儲存貯體擁有者可授予其他 AWS 帳戶 上傳物件的許可。根據預設，來源儲存貯體擁有者不具其他帳戶所建立之物件的任何許可。複寫組態只會複寫來源儲存貯體擁有者具備存取許可的物件。來源儲存貯體擁有者可以授予其他 AWS 帳戶 許可來有條件地建立物件，並要求這些物件的明確存取許可。如需政策範例，請參閱 授予跨帳戶許可，以在確保儲存貯體擁有者具有完全控制時上傳物件。

- 假設您在複寫組態中新增一個規則，以複寫含特定標籤的物件子集。在此情況下，您必須於建立物件時指派特定標籤金鑰與值，以便 Amazon S3 複寫物件。如果您先建立物件，之後才將標籤新增至現有物件，Amazon S3 就不會複寫物件。
- 使用「Amazon S3 事件通知」，將物件未複寫至其目的地 AWS 區域 的情況通知您。Amazon S3 事件通知可透過 Amazon Simple Queue Service (Amazon SQS)、Amazon Simple Notification Service (Amazon SNS) 或 AWS Lambda 取得。如需詳細資訊，請參閱 使用 Amazon S3 事件通知接收複寫失敗事件。

  您也可以使用「Amazon S3 事件通知」，以檢視複寫失敗原因。若要檢閱失敗原因的清單，請參閱 Amazon S3 複寫失敗原因。

批次複寫錯誤

若要針對未複寫到目的地儲存貯體的物件進行疑難排解，請針對用來建立「批次複寫」任務的儲存貯體、複寫角色和 IAM 角色檢查不同類型的許可。此外，請務必檢查公開存取設定和儲存貯體擁有權設定。

使用「批次複寫」時，您可能會遇到下列其中一個錯誤：

- 批次操作狀態為失敗，原因如下：無法將任務報告寫入報告儲存貯體。

  如果用於「批次操作」任務的 IAM 角色無法將完成報告放入您建立任務時所指定的位置中，就會發生此錯誤。若要解決此錯誤，請檢查 IAM 角色是否對要儲存「批次操作」完成報告的儲存貯體具有 `PutObject` 許可。最佳實務是將報告遞交到與來源儲存貯體不同的儲存貯體。
- 批次操作已完成但失敗，失敗總數不是 0。

  如果正在執行的「批次複寫」任務發生物件許可不足問題，則會發生這個錯誤。如果您針對「批次複寫」任務使用複寫規則，請確定用於複寫的 IAM 角色具有適當的許可，可從來源或目的地儲存貯體中存取物件。您也可以檢查批次複寫完成報告，以檢閱特定的 Amazon S3 複寫失敗原因。
- 批次任務已成功執行，但目的地儲存貯體中預期的物件數目不相同。

當「批次複寫」任務中提供的清單檔案中所列的物件與您在建立任務時選取的篩選條件之間若有不符時，就會發生這個錯誤。當來源儲存貯體中的物件不符合任何複寫規則，且未包含在產生的清單檔案中時，您可能也會收到此訊息。

針對伺服器存取記錄進行疑難排解

下列主題可以協助您針對使用 Amazon S3 設定記錄時可能遇到的問題進行疑難排解。

主題

設定記錄時的常見錯誤訊息

透過 AWS Command Line Interface (AWS CLI) 和 AWS SDK 啟用記錄時，可能會出現下列常見的錯誤訊息：

錯誤：不允許跨 S3 位置記錄

如果目的地儲存貯體 (也稱為目標儲存貯體) 與來源儲存貯體位於不同區域，則會發生不允許跨 S3 位置記錄錯誤。若要解決此錯誤，請確定設定為接收存取日誌的目的地儲存貯體與來源儲存貯體位於相同的 AWS 區域 和 AWS 帳戶。

錯誤：要記錄的儲存貯體與目標儲存貯體，兩者的擁有者必須相同

當您啟用伺服器存取記錄時，如果指定的目的地儲存貯體屬於不同的帳戶，就會發生此錯誤。若要解決此錯誤，請確定目的地儲存貯體與來源儲存貯體位於相同的 AWS 帳戶。

> **Note**
>
> 建議您選擇與來源儲存貯體不同的目的地儲存貯體。當來源儲存貯體與目的地儲存貯體相同時，會為寫入儲存貯體的日誌建立額外的日誌，這會增加您的儲存費用。這些關於日誌的額外日誌也可能使您難以找到您要尋找的特定日誌。如需更簡單的日誌管理，建議您將存取日誌儲存在不同的儲存貯體中。如需詳細資訊，請參閱 the section called “如何啟用日誌交付？”。

錯誤：用於記錄的目標儲存貯體不存在

在設定組態之前，目的地儲存貯體必須存在。此錯誤表示目的地儲存貯體不存在或找不到。請確定儲存貯體名稱拼寫正確，然後再試一次。

錯誤：儲存貯體擁有者強制執行的儲存貯體不允許目標授予

此錯誤表示目的地儲存貯體針對「S3 物件擁有權」使用儲存貯體擁有權強制執行設定。儲存貯體擁有者強制執行設定不支援目的地 (目標) 授權。如需詳細資訊，請參閱 日誌交付許可。

針對交付失敗進行疑難排解

若要避免伺服器存取記錄問題，請確定您遵循下列最佳實務：

- S3 日誌交付群組具有目的地儲存貯體的寫入存取權 - S3 日誌交付群組可將伺服器存取日誌交付到目的地儲存貯體。儲存貯體政策或儲存貯體存取控制清單 (ACL) 可以用來將寫入存取權授予目的地儲存貯體。不過，建議您使用儲存貯體政策，而不是 ACL。如需如何將寫入存取權授予目的地儲存貯體的詳細資訊，請參閱 日誌交付許可。

  > **Note**
  >
  > 如果目的地儲存貯體針對「物件擁有權」使用儲存貯體擁有權強制執行設定，請注意下列事項：
  >
  > - ACL 已停用，不再影響許可。這表示您無法更新儲存貯體 ACL 以授予 S3 日誌交付群組的存取權。相反地，若要授予記錄服務主體的存取權，您必須更新目的地儲存貯體的儲存貯體政策。
  > - 您無法將目的地授權納入您的 `PutBucketLogging` 組態。

- 目的地儲存貯體的儲存貯體政策允許存取日誌 - 檢查目的地儲存貯體的儲存貯體政策。搜尋儲存貯體政策找出包含 `"Effect": "Deny"` 的任何陳述式。然後，確認 Deny 陳述式並未阻止存取日誌寫入儲存貯體。
- 目的地儲存貯體上未啟用 S3 物件鎖定 - 檢查目的地儲存貯體是否啟用了「物件鎖定」。「物件鎖定」會封鎖伺服器存取日誌交付。您必須選擇未啟用「物件鎖定」的目的地儲存貯體。
- 如果目的地儲存貯體上已啟用預設加密，則會選取 Amazon S3 受管金鑰 (SSE-S3) - 只在您使用伺服器端加密搭配 Amazon S3 受管金鑰 (SSE-S3) 時，才能在目的地儲存貯體上使用預設儲存貯體加密。伺服器存取記錄目的地儲存貯體不支援使用 AWS Key Management Service (AWS KMS) 金鑰的預設伺服器端加密。如需啟用預設加密的詳細資訊，請參閱 設定預設加密。
- 目的地儲存貯體未啟用請求者付款 - 不支援使用「請求者付款」儲存貯體作為伺服器存取記錄的目的地儲存貯體。若要允許交付伺服器存取日誌，請停用目的地儲存貯體上的「請求者付款」選項。

- 檢閱您的 AWS Organizations 服務控制政策 - 使用 AWS Organizations 時，請檢查服務控制政策以確保允許 Amazon S3 存取。服務控制政策指定受影響帳戶的許可數上限。搜尋服務控制政策找出包含 `"Effect": "Deny"` 的任何陳述式，並確認 Deny 陳述式不會阻止任何存取日誌寫入儲存貯體。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的服務控制政策 (SCP)。
- 預留一些時間讓最近記錄組態變更生效 - 第一次啟用伺服器存取記錄，或變更日誌的目的地儲存貯體，需要時間才能完全生效。所有請求可能需要一個小時以上的時間才能適當地記錄和交付。

  若要檢查日誌交付失敗，請在 Amazon 中啟用請求指標 CloudWatch。如果日誌未在幾個小時內交付，請尋找 `4xxErrors` 指標，其可以指出日記交付失敗。如需啟用請求指標的詳細資訊，請參閱 the section called “建立所有物件的指標組態”。

針對版本控制疑難排解

下列主題可協助您針對一些常見的 Amazon S3 版本控制問題進行疑難排解。

主題

- 我想要復原已啟用版本控制的儲存貯體中意外刪除的物件
- 我想永久刪除版本控制的物件
- 啟用儲存貯體版本控制後，我遭遇效能降低

我想要復原已啟用版本控制的儲存貯體中意外刪除的物件

一般而言，從 S3 儲存貯體刪除物件版本時，沒有方法可供 Amazon S3 復原它們。不過，如果您已在 S3 儲存貯體上啟用 S3 版本控制，則未指定版本 ID 的 DELETE 請求將無法永久刪除物件。相反地，刪除標記會新增為預留位置。此刪除標記會成為物件的目前版本。

若要驗證已刪除的物件是永久刪除還是暫時刪除 (其位置有刪除標記)，請執行下列動作：

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在左側導覽窗格中，選擇 Buckets (儲存貯體)。
3. 在 Buckets (儲存貯體) 清單中，選擇包含該物件的儲存貯體名稱。
4. 在物件清單中，開啟搜尋列右側的顯示版本切換，然後在搜尋列中搜尋已刪除的物件。只在儲存貯體上先前已啟用版本控制時，才能使用此切換。

   您也可以使用 S3 清查來搜尋已刪除的物件。

5. 如果您在切換顯示版本或建立清查報告之後找不到物件，也找不到物件的刪除標記，則刪除是永久的，而且無法復原該物件。

您也可以使用 AWS Command Line Interface (AWS CLI) 中的 HeadObject API 操作來驗證已刪除物件的狀態。若要這樣做，請使用下列 head-object 命令，並以您自己的資訊取代 *user input placeholders*。

```
aws s3api head-object --bucket example-s3-bucket --key index.html
```

如果在目前版本為刪除標記的已版本控制物件上執行 head-object 命令，您將會收到 404 找不到錯誤訊息。例如：

調用 HeadObject 操作時發生錯誤（404）：未找到

如果您在已版本控制的物件上執行 head-object 命令，並提供物件的版本 ID，Amazon S3 會擷取物件的中繼資料，確認該物件仍然存在且不會永久刪除。

```
aws s3api head-object --bucket example-s3-bucket --key index.html --
version-id versionID
```

```
{
"AcceptRanges": "bytes",
"ContentType": "text/html",
"LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
"ContentLength": 77,
"VersionId": "Zg5HyL7m.eZU9iM7AVlJkrqAiE.0UG4q",
"ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
"Metadata": {}
}
```

如果找到物件且最新版本是刪除標記，則該物件的先前版本仍然存在。由於刪除標記是物件的目前版本，您可以將刪除標記刪除，來復原物件。

在您永久移除刪除標記之後，物件的第二個最新版本會變成物件的目前版本，使您的物件再次可用。如需如何復原物件的視覺描述，請參閱移除刪除標記。

若要移除特定版本的物件，您必須是儲存貯體擁有者。若要對刪除標記永久刪除，您必須在 DeleteObject 要求中包含其版本 ID。若要將刪除標記刪除，請使用下列命令，並以您自己的資訊取代 *user input placeholders*。

```
aws s3api delete-object --bucket example-s3-bucket --key index.html --
version-id versionID
```

如需 delete-object 命令的詳細資訊，請參閱《AWS CLI 命令參考》中的 delete-object。如需永久將刪除標記刪除的詳細資訊，請參閱 管理刪除標記。

我想永久刪除版本控制的物件

在已啟用版本控制的儲存貯體中，沒有版本 ID 的 DELETE 請求無法永久刪除物件。相反地，這類請求會插入刪除標記。

若要永久刪除已版本控制的物件，您可以從下列方法中選擇：

- 建立 S3 生命週期規則以永久刪除非目前版本。若要永久刪除非現行版本，請選取永久刪除物件的非現行版本，然後在物件變成非現行版本後的天數中輸入天數。您可以選擇指定要保留的較新版本數目，方法是在要保留的較新版本數目下方輸入值。如需建立此規則的詳細資訊，請參閱設定 S3 生命週期組態。
- 透過在 DELETE 請求中包含版本 ID 來刪除指定的版本。如需詳細資訊，請參閱如何永久刪除已版本控制的物件。
- 建立生命週期規則以結束目前版本。若要結束物件的目前版本，請選取讓物件的目前版本過期，然後在物件建立後的天數中輸入天數。如需建立此生命週期規則的詳細資訊，請參閱設定 S3 生命週期組態。
- 若要永久刪除所有已版本控制的物件並刪除標記，請建立兩個生命週期規則：一個讓目前版本過期，並永久刪除非目前版本的物件，另一個則用於刪除過期的物件刪除標記。

在已啟用版本控制的儲存貯體中，未指定版本 ID 的 DELETE 請求只能移除版本 ID 為 NULL 的物件。如果物件是在啟用版本控制時上載，則未指定版本 ID 的 DELETE 請求會建立該物件的刪除標記。

> **Note**
>
> 對於啟用 S3 物件鎖定的儲存貯體，具有受保護DELETE物件版本 ID 的物件請求會導致 403 拒絕存取錯誤。沒有版本 ID 的 DELETE 物件請求會將刪除標記新增為具有 200 OK 回應之物件的最新版本。無法永久刪除受「物件鎖定」保護的物件，直到移除其保留期和法務保存。如需詳細資訊，請參閱 the section called “S3 物件鎖定的運作方式”。

啟用儲存貯體版本控制後，我遭遇效能降低

如果刪除標記或已版本控制的物件太多，以及如果未遵循最佳實務，則已啟用版本控制的儲存貯體可能會降低效能。

刪除標記過多

在您於儲存貯體上啟用版本控制之後，對物件所做的 DELETE 請求 (沒有版本 ID) 會使用唯一的版本 ID 來建立刪除標記。具有讓物件的目前版本過期規則的生命週期組態會將具有唯一版本 ID 的刪除標記新增至每個物件。過多的刪除標記可能會降低儲存貯體中的效能。

儲存貯體上的版本控制暫停時，Amazon S3 會在新建立的物件上，將版本 ID 標記為 NULL。在暫停版本控制的儲存貯體中，過期動作會指示 Amazon S3 建立刪除標記，並以 NULL 作為版本 ID。在暫停版本控制的儲存貯體中，會針對任何刪除請求建立 NULL 刪除標記。當刪除所有物件版本，僅保留單一刪除標記時，這些 NULL 刪除標記也稱為過期的物件刪除標記。如果累積了太多 NULL 刪除標記，則儲存貯體中的效能會降低。

已版本控制的物件太多

如果已啟用版本控制的儲存貯體包含具有數百萬個版本的物件，則可能會增加 503 服務無法使用錯誤。當注意到針對已啟用 S3 版本控制之儲存貯體的 PUT 或 DELETE 物件請求，接收 HTTP 503 無法使用服務回應的數目明顯增加時，您可能在儲存貯體中具有一或多個含數百萬個版本的物件。當您的物件具有數百萬個版本時，Amazon S3 便會自動對儲存貯體的請求限流。限流請求可保護儲存貯體免於過多的請求流量，這可能會潛在地阻礙對同一個儲存貯體發出其他請求。

若要判斷哪些物件具有數百萬個版本，請使用「S3 清查」。「S3 清查」會產生一份報告，提供儲存貯體中物件的一般檔案清單。如需詳細資訊，請參閱 Amazon S3 清查。

若要驗證儲存貯體中是否有大量已版本控制的物件，請使用 S3 Storage Lens 指標來檢視目前版本物件計數、非目前版本物件計數和刪除標記物件計數。如需 Storage Lens 指標的詳細資訊，請參閱 Amazon S3 Storage Lens 指標詞彙表。

如果應用程式一再覆寫同一個物件，導致物件可能產生數百萬個版本，Amazon S3 團隊建議客戶展開調查，以判斷應用程式是否正常運作。例如，一週內每分鐘覆寫相同物件的應用程式可以建立超過一萬個版本。我們建議每個物件儲存不超過十萬個版本。如果您的使用案例需要一或多個物件的數百萬個版本，請聯絡 AWS Support 小組以協助您決定更好的解決方案。

最佳實務

為了防止版本控制相關的效能降低問題，建議您採用下列最佳實務：

- 啟用生命週期規則，使物件的先前版本過期。例如，您可以建立生命週期規則，在物件變成非目前版本的 30 天後使非目前版本過期。如果您不想要刪除所有的非目前版本，也可以保留多個非目前版本。如需詳細資訊，請參閱設定 S3 生命週期組態。
- 啟用生命週期規則，以刪除儲存貯體中沒有關聯資料物件的過期物件刪除標記。如需詳細資訊，請參閱移除過期的物件刪除標記。

如需其他 Amazon S3 效能最佳化最佳實務，請參閱最佳實務設計模式。

獲取 Amazon S3 請求 ID AWS Support

每當您 AWS Support 因為在 Amazon S3 中遇到錯誤或非預期行為而聯絡時，都必須提供與失敗動作相關聯的請求 ID。 AWS Support 使用這些要求 ID 來協助解決您遇到的問題。

請求 ID 成對出現、在 Amazon S3 處理的每個回應中傳回 (甚至是錯誤回應)，而且可透過詳細日誌取得。取得請求 ID 的常用方法有許多，包括 S3 存取日誌和 AWS CloudTrail 事件或資料事件。

復原這些記錄檔之後，請複製並保留這兩個值，因為聯絡時會需要這兩個值 AWS Support。如需有關連絡人的資訊 AWS Support，請參閱連絡人 AWS或AWS Support 說明文件。

使用 HTTP 取得請求 ID

您可以在 HTTP 要求傳抵目標應用程式之前，先記錄 HTTP 要求的位元數，從而取得自己的要求 ID `x-amz-request-id` 與 `x-amz-id-2`。您有多種第三方工具可供您用來復原 HTTP 要求的詳細日誌。請選擇您信任的工具來執行，當您送出另一個 Amazon S3 HTTP 請求時，接聽 Amazon S3 流量行經的連接埠。

HTTP 請求的請求 ID 對類似下列範例所示：

```
x-amz-request-id: 79104EXAMPLEB723
x-amz-id-2: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km
```

> **Note**
>
> 在大部分的封包擷取中，多會加密及隱藏 HTTPS 要求。

使用 Web 瀏覽器取得請求 ID

大多數 Web 瀏覽器都會提供開發人員工具，您可用來檢視請求標頭。

傳回錯誤之 Web 瀏覽器要求的要求 ID 對類似下列範例所示。

```
<Error><Code>AccessDenied</Code><Message>Access Denied</Message>
<RequestId>79104EXAMPLEB723</RequestId><HostId>IOWQ4fDEXAMPLEQM
+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km</HostId></Error>
```

若要從成功的請求取得請求 ID 對，請使用瀏覽器的開發人員工具檢視 HTTP 回應標頭。如需適用於特定瀏覽器的開發人員工具的資訊，請參閱 AWS re:Post 中的 Amazon S3 疑難排解 – 如何復原 S3 請求 ID。

使用 AWS SDK 取得要求識別碼

下列各節包含如何使用 AWS 開發套件設定日誌的資訊。雖然您可以啟用詳細記錄功能，記錄每個要求與回應，但不建議在生產系統中啟用記錄，因為大量的要求或回應可能會造成應用程式的速度明顯變慢。

對於 AWS SDK 請求，一對請求 ID 看起來像下列範例。

```
Status Code: 403, AWS Service: Amazon S3, AWS Request ID: 79104EXAMPLEB723
AWS Error Code: AccessDenied  AWS Error Message: Access Denied
S3 Extended Request ID: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK
+Jd1vEXAMPLEa3Km
```

使用適用於 Go 的 SDK 來取得要求識別碼

您可以使用適用 SDK for Go 來設定記錄。如需詳細資訊，請參閱 Go V2 開發人員指南中的 SDK 中的回應中繼資料。

使用適用於 PHP 的開發套件以取得請求 ID

您可使用 PHP 設定記錄功能。如需詳細資訊，請參閱適用於《AWS SDK for PHP 開發人員指南》中的如何得知網路上傳送哪些資料？。

使用適用於 Java 的開發套件取得請求 ID

您可以為特定請求或回應啟用記錄功能，只擷取及傳回相關的標頭。若要執行此作業，請匯入 `com.amazonaws.services.s3.S3ResponseMetadata` 類別。如此一來，您就能將請求存放在變數中，然後再執行實際的要求。若要取得所記錄的請求或回應，請呼叫 `getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID()`。

Example

```
PutObjectRequest req = new PutObjectRequest(bucketName, key, createSampleFile());
s3.putObject(req);
S3ResponseMetadata md = s3.getCachedResponseMetadata(req);
System.out.println("Host ID: " + md.getHostId() + " RequestID: " + md.getRequestId());
```

或者，您可以對每一個 Java 要求與回應使用詳細記錄功能。如需詳細資訊，請參閱《AWS SDK for Java 開發人員指南》中的詳細連線記錄。

使用取得 AWS SDK for .NET 要求識別碼

您可以使用內建 AWS SDK for .NET 的記錄工具來設定`System.Diagnostics`記錄。如需詳細資訊，請參閱使用 AWS SDK for .NET AWS 開發人員部落格文章的記錄。

> **Note**
>
> 傳回的記錄預設只會包含錯誤資訊。若要取得請求 ID，必須在設定檔中新增 `AWSLogMetrics` (以及選用的 `AWSResponseLogging`)。

使用適用於 Python (Boto3) 的開發套件取得請求 ID

使用 AWS SDK for Python (Boto3)，您可以記錄特定回應。您可以使用此功能僅擷取相關標頭。下列程式碼顯示如何將回應的部分記錄到檔案中：

```
import logging
import boto3
logging.basicConfig(filename='logfile.txt', level=logging.INFO)
logger = logging.getLogger(__name__)
s3 = boto3.resource('s3')
response = s3.Bucket(bucket_name).Object(object_key).put()
logger.info("HTTPStatusCode: %s", response['ResponseMetadata']['HTTPStatusCode'])
logger.info("RequestId: %s", response['ResponseMetadata']['RequestId'])
logger.info("HostId: %s", response['ResponseMetadata']['HostId'])
logger.info("Date: %s", response['ResponseMetadata']['HTTPHeaders']['date'])
```

您也可以擷取例外狀況，並在引發例外狀況時記錄相關資訊。如需詳細資訊，請參閱適用於 Python (Boto) 的AWS 開發套件 API 參考中的從錯誤回應中辨別實用的資訊。

此外，您可以使用下列程式碼，配置 Boto3 輸出詳細資訊偵錯記錄檔：

```
import boto3
boto3.set_stream_logger('', logging.DEBUG)
```

如需詳細資訊，請參閱適用於 Python (Boto) 的AWS 開發套件 API 參考中的 set_stream_logger。

使用適用於 Ruby 的開發套件取得請求 ID

您可以使用適用於 Ruby 的開發套件版本 1、2 或 3，以取得請求 ID。

- 使用適用於 Ruby 的開發套件 - 版本 1 – 您可以使用下一行程式碼，全域啟用 HTTP 連線記錄。

  ```
  s3 = AWS::S3.new(:logger => Logger.new($stdout), :http_wire_trace => true)
  ```

- 使用適用於 Ruby 的開發套件 - 版本 2 或版本 3 – 您可以使用下一行程式碼，全域啟用 HTTP 連線記錄。

  ```
  s3 = Aws::S3::Client.new(:logger => Logger.new($stdout), :http_wire_trace => true)
  ```

如需從 AWS 用戶端取得電線資訊的提示，請參閱偵錯秘訣：從用戶端取得配線追蹤資訊。

使用取得 AWS CLI 要求識別碼

若要在使用 AWS Command Line Interface (AWS CLI) 時取得要求 ID，請新增`--debug`至您的命令。

使用視窗取得 PowerShell 要求識別碼

如需有關使用 Windows 復原記錄檔的資訊 PowerShell，請參閱 AWS Tools for Windows PowerShell.NET 開發中的回應記錄部落格文章。

使用 AWS CloudTrail 資料事件取得要求 ID

使用資 CloudTrail 料事件設定的 Amazon S3 儲存貯體以記錄 S3 物件層級 API 操作，可提供有關 Amazon S3 中使用者、角色或 AWS 服務採取的動作的詳細資訊。您可以透過向 Athena 查詢 CloudTrail 事件來識別 S3 請求 ID。

使用 S3 伺服器存取記錄日誌來取得請求 ID

為 S3 伺服器存取記錄設定的 Amazon S3 儲存貯體，針對向儲存貯體提出的請求，提供詳細的記錄。您可以使用 Athena 查詢伺服器存取日誌來識別 S3 請求 ID。

文件歷史記錄

- 目前的 API 版本：2006-03-01

下表說明《Amazon Simple Storage Service API 參考》和《Amazon S3 使用者指南》每一發行版本中的重要變更。如需有關此文件更新的通知，您可以訂閱 RSS 摘要。

變更	描述	日期
Amazon S3 庫存支持 s3：InventoryAccessibleOptionalFields 條件密鑰	Amazon S3 庫存支援 s3：InventoryAccessibleOptionalFields 條件金鑰，以控制使用者是否可以在報告中包含選用的中繼資料欄位。如需詳細資訊，請參閱控制 S3 庫存報告組態建立。	2024年2月20日
Outposts 上 S3 的 IPv6 支援	您現在可以在 Outposts 雙堆疊端點上透過 S3 使用 IPv6 存取 Outposts 儲存貯體上的 S3。Outposts S3 的 IPv6 支援可讓您管理前哨儲存貯體上的 S3，以及透過 IPv6 網路的控制平面資源。	2024年1月16日
全新的高效能單一區域 Amazon S3 儲存類別 — S3 Express One Zone	Amazon S3 Express One Zone 是一種高效能的單一區域 Amazon S3 儲存類別，專門為對延遲最敏感的應用程式提供一致的個位數毫秒資料存取。如需詳細資訊，請參閱 S3 快速單一區域。	2023 年 11 月 28 日
適用於 Amazon S3 的掛載點新增對 S3 Express One Zone 的支援	現在您可以使用掛載點來掛載 S3 Express One Zone 目錄儲存貯體。	2023 年 11 月 28 日

Lambda 調用結構描述版本	Amazon S3 Batch Operations 導入新的 Lambda 調用結構描述版本，可搭配對目錄儲存貯體執行動作的 Batch Operations 任務使用。如需詳細資訊，請參閱使用 Lambda 和 Amazon S3 Batch Operations 搭配目錄儲存貯體。	2023 年 11 月 28 日
目錄儲存貯體的匯入動作	Amazon S3 導入了匯入動作。匯入是用來建立 Amazon S3 Batch Operations 任務的簡化方法，可將物件從一般用途儲存貯體複製到目錄儲存貯體。如需詳細資訊，請參閱將物件匯入目錄儲存貯體。	2023 年 11 月 28 日
使用 S3 Access Grants 管理 S3 存取	Amazon S3 存取授權可讓您大規模管理 AWS Identity and Access Management (IAM) 主體的資料許可，以及來自公司目錄的目錄身分，例如Azure AD。現在您可以強制執行最低權限 S3 許可，並根據業務需求輕鬆擴展這些許可。如需詳細資訊，請參閱使用 S3 Access Grants 管理存取。	2023 年 11 月 26 日
適用於 Amazon S3 的掛載點新增快取功能	您現在可以使用掛載點為重複存取的資料設定快取。	2023 年 11 月 22 日

增強的 Amazon S3 Batch 操作資訊清單產生	現在您可以指示 Amazon S3 Batch Operations 根據您在建立任務時指定的物件篩選條件自動產生清單檔案。此選項適用於您在 Amazon S3 主控台中建立的批次複寫任務，或使用 AWS 開發套件或 Amazon S3 REST API 建立的任何任務類型。 AWS CLI如需詳細資訊，請參閱建立 Amazon S3 Batch Operations 任務。	2023 年 11 月 22 日
現有的 Amazon S3 儲存貯體現在可以新增物件鎖定組態	現在您可以在現有 S3 儲存貯體上啟用物件鎖定。您可以為新的或現有的儲存貯體設定法務保存和保留期。如需詳細資訊，請參閱使用物件鎖定。	2023 年 11 月 20 日
S3 Storage Lens 字首的請求指標	S3 Storage Lens 在 Amazon S3 儲存貯體內導入了字首的請求指標。如需詳細資訊，請參閱指標類別。	2023 年 11 月 17 日
Amazon S3 Storage Lens 群組	S3 Storage Lens 導入了 Storage Lens 群組，這是根據物件中繼資料為物件的定義的自訂篩選條件。如需詳細資訊，請參閱使用 Amazon S3 Storage Lens 群組。	2023 年 11 月 15 日
新的 IAM 政策	S3 on Outposts 導入了服務連結角色 `AWSServiceRoleForS3OnOutposts` ，有助於管理網路資源。如需詳細資訊，請參閱針對 S3 on Outposts 使用服務連結角色。	2023 年 10 月 3 日

Amazon S3 提供用於刪除標記的 Last-Modified 時間	Amazon S3 會在 S3 Head 和 Get API 操作的回應標頭中，提供刪除標記的 Last-Modified 時間。如需詳細資訊，請參閱使用刪除標記。	2023 年 9 月 27 日
Amazon S3 更新到 AWS 受管政策	Amazon S3 新增了 s3:Describe* 許可至 AmazonS3ReadOnlyAccess 。如需詳細資訊，請參閱 Amazon S3 的 AWS 受管政策。	2023 年 8 月 11 日
改善透過 S3 批次操作發出標準還原請求的開始時間	透過 S3 批次操作發出之還原請求的標準擷取，現在可在幾分鐘內開始。如需詳細資訊，請參閱封存擷取選項。	2023 年 8 月 9 日
新增掛載點，這是高輸送量的用戶端，可將 Amazon S3 儲存貯體掛載為本機檔案系統。	有了掛載點，您的應用程式就可以透過檔案操作存取儲存在 Amazon S3 中的物件，利用檔案介面讓您的應用程式存取 Amazon S3 的彈性儲存和輸送量。	2023 年 8 月 9 日
使用 AWS Key Management Service 金鑰 (DSSE-KMS) 進行雙層伺服器端加密	使用 () 金鑰 AWS Key Management Service (DSSE-KMS AWS KMS) 的雙層伺服器端加密會在物件上傳到 Amazon S3 時，將兩層加密套用至物件。如需詳細資訊，請參閱搭配 AWS KMS 金鑰使用雙層伺服器端加密。	2023 年 6 月 13 日

Amazon S3 啟用 S3 封鎖公有存取權限，並停用所有新儲存貯體的 S3 存取控制清單 (ACL)。	Amazon S3 現在會自動啟用 S3 區塊公開存取，並停用所有 AWS 區域中所有新 S3 儲存貯體的 S3 存取控制清單 (ACL)。如需詳細資訊，請參閱封鎖對 Amazon S3 儲存體的公開存取權和控制物件的所有權並停用儲存貯體的 ACL。	2023 年 4 月 27 日
S3 複寫操作失敗指標	Amazon S3 新增了新的 Amazon CloudWatch 指標來監控 S3 複寫失敗情況。如需詳細資訊，請參閱監控複寫指標的進度。	2023 年 4 月 5 日
私有 DNS	AWS PrivateLink Amazon S3 現在支持私有 DNS。如需更多詳細資訊，請參閱私有 DNS。	2023 年 3 月 14 日
Amazon S3 主控台支援跨帳戶存取點	Amazon S3 現在支援以 Amazon S3 主控台建立跨帳戶存取點。如需詳細資訊，請參閱建立存取點。	2023 年 3 月 14 日
Amazon S3 on Outposts 支援 Outposts 上的 S3 複寫	透過本機 S3 複寫，您可將物件自動複寫到單一 Outposts 目的地儲存貯體或多個目的地儲存貯體。目標值區可以位於與來源值區不 AWS Outposts 同的 Outposts 中，也可以位於相同的 Outposts 中。如需詳細資訊，請參閱複寫 S3 on Outposts 的物件。	2023 年 3 月 14 日

Amazon S3 Object Lambda 存取點別名	在您建立 Object Lambda 存取點時，Amazon S3 會自動為您的 Object Lambda 存取點產生唯一的別名。您可以針對存取點資料平面操作在請求中使用此別名，而不是使用 Amazon S3 儲存貯體名稱或 Object Lambda 存取點 Amazon Resource Name (ARN)。如需詳細資訊，請參閱如何為您的物件 Lambda 存取點使用儲存貯體式別名。	2023 年 3 月 14 日
Amazon S3 多區域存取點跨帳戶支援	Amazon S3 現在支援以 Amazon S3 主控台建立跨帳戶多區域存取點。如需詳細資訊，請參閱建立多區域存取點。	2023 年 3 月 14 日
跨帳戶存取點	Amazon S3 支援建立跨帳戶存取點。您可以使用 AWS Command Line Interface (AWS CLI) 或 REST API `CreateAccessPoint` 操作來建立跨帳戶存取點。如需詳細資訊，請參閱建立存取點。	2022 年 11 月 30 日
Amazon S3 支援 Amazon S3 多區域存取點的容錯移轉控制	Amazon S3 引進了多區域存取點的容錯移轉控制。這些控制可讓您在幾分鐘內將透過 Amazon S3 多區域存取點路由的 S3 資料存取請求流量轉移到替代 AWS 區域 ，以測試並建置高度可用的應用程式。如需詳細資訊，請參閱 Amazon S3 多區域存取點容錯移轉控制。	2022 年 11 月 28 日

Amazon S3 Storage Lens 透過 34 個新指標提高整個組織的可見度	S3 Storage Lens 引進了 34 個額外指標，以發掘更深入的成本最佳化機會、識別資料保護最佳實務，並改善應用程式工作流程的效能。如需詳細資訊，請參閱 S3 Storage Lens 指標。	2022 年 11 月 17 日
對於 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive，Amazon S3 支援更高的還原請求速率	AWS 帳戶 對於 S3 冰川靈活擷取和 S3 Glacier Deep Archive 儲存類別，Amazon S3 以每秒高達 1,000 筆交易的速率支援還原請求。	2022 年 11 月 15 日
Amazon S3 on Outposts 支援其他 S3 生命週期動作和篩選條件	S3 on Outposts 支援其他 S3 生命週期規則來最佳化容量管理。您可以在物件老化或取代為較新的版本時使這些物件過期。您可以使用字首、物件索引標籤、或物件大小進行篩選，針對整個儲存貯體或儲存貯體中的物件子集建立生命週期規則。如需詳細資訊，請參閱建立和管理生命週期組態。	2022 年 11 月 2 日
SSE-C 物件支援 S3 複寫	您可以複寫使用伺服器端加密搭配客戶提供金鑰所建立的物件。如需複寫加密物件的詳細資訊，請參閱複寫使用伺服器端加密 (SSE-C、SSE-S3、SSE-KMS) 建立的物件。	2022 年 10 月 24 日

Amazon S3 on Outposts 支援存取點別名	使用 S3 on Outposts，您必須使用存取點來存取 Outposts 儲存貯體中的任何物件。每次建立儲存貯體的存取點時，S3 on Outposts 都會自動產生存取點別名。您可以針對任何資料平面操作使用此存取點別名，而不是存取點 Amazon Resource Name (ARN)。如需詳細資訊，請參閱針對您的 S3 on Outposts 存取點使用儲存貯體樣式別名。	2022 年 10 月 21 日
S3 Object Lambda 支援 `HeadObject` 、`ListObjects` 和 `ListObjectsV2` 操作	您可以使用自訂程式碼，修改標準 S3 GET、LIST 或 HEAD 請求傳回的資料，以篩選資料列、動態調整影像大小、修訂機密資料，以及更多動作。如需詳細資訊，請參閱使用 S3 Object Lambda 轉換物件。	2022 年 10 月 4 日
Amazon S3 on Outposts 支援 S3 版本控制	啟用時，S3 版本控制會在相同的儲存貯體中儲存物件的多個不同複本。您可以使用 S3 版本控制，保留、擷取和還原在 Outposts 儲存貯體中所存放每個物件的各個版本。S3 版本控制可協助您從意外的使用者動作和應用程式失敗中復原。如需詳細資訊，請參閱針對您的 S3 on Outposts 儲存貯體管理 S3 版本控制。	2022 年 9 月 21 日

AWS Backup 對於 Amazon S3	AWS Backup 這是一項全受管的原則式服務，可用來定義中央備份政策以保護 Amazon S3 資料。如需詳細資訊，請參AWS Backup 閱使用 Amazon S3。	2022 年 2 月 18 日
使用 S3 批次複寫來複寫現有物件	使用 S3 批次複寫，您可以複寫在複寫組態就位之前就存在的物件。複寫現有物件是透過使用批次操作任務來完成的。S3 批次複寫不同於即時複寫，後者會跨 Amazon S3 儲存貯體持續自動複製新物件。如需詳細資訊，請參閱使用 S3 批次複寫來複寫現有物件。	2022 年 2 月 8 日
S3 Glacier Flexible Retrieval 的重新命名	Glacier 儲存類別已重新命名為 S3 Glacier Flexible Retrieval。此變更不會影響 API。	2021 年 11 月 30 日
停用 ACL 的新 S3 物件擁有權設定	您可對物件所有權套用儲存貯體擁有者強制執行的設定，以停用儲存貯體及其中物件的 ACL，並取得儲存貯體中每個物件的擁有權。儲存貯體擁有者強制執行的設定可簡化存放在 Amazon S3 中之資料的存取管理。如需詳細資訊，請參閱控制物件的擁有權並停用儲存貯體的 ACL。	2021 年 11 月 30 日
新 S3 Intelligent-Tiering 儲存類別	S3 Intelligent-Tiering Archive Instant Access 是 S3 Intelligent-Tiering 下的額外儲存類別。如需詳細資訊，請參閱 S3 智慧型分層服務的運作方式。	2021 年 11 月 30 日

新 S3 Glacier Instant Retrieval 儲存類別	您現在可以將物件放置在 S3 Glacier Instant Retrieval 儲存類別中。如需此儲存類別的詳細資訊，請參閱使用 Amazon S3 儲存類別。	2021 年 11 月 30 日
AWS Backup 適用於 Amazon S3 預覽	AWS Backup 這是一項全受管的原則式服務，可用來定義中央備份政策以保護 Amazon S3 資料。如需詳細資訊，請參AWS Backup 閱使用 Amazon S3。	2021 年 11 月 30 日
AWS Identity and Access Management Access Analyzer 對於 Amazon S3	IAM Access Analyzer 會比對 IAM 政策文法和最佳實務來執行政策檢查，以驗證您的政策。若要進一步了解如何使用 IAM Access Analyzer 驗證政策，請參閱《IAM 使用者指南》中的 IAM Access Analyzer 政策驗證。	2021 年 11 月 30 日
新事件類型	Amazon S3 事件通知新增的事件類型，請參閱 Amazon S3 事件通知。	2021 年 11 月 29 日
EventBridge 在桶上啟用 Amazon	您可以 EventBridge 在 Amazon S3 儲存貯體上啟用以將事件傳送到 Amazon EventBridge，請參閱使用 EventBridge.	2021 年 11 月 29 日
新 S3 生命週期篩選條件	您可以根據物件大小建立生命週期規則，或指定要保留的非目前物件版本數目。如需詳細資訊，請參閱 S3 生命週期組態的範例。	2021 年 11 月 23 日

將 Amazon S3 儲存鏡頭指標發佈到 Amazon CloudWatch	您可以將 S3 儲存鏡頭使用情況和活動指標發佈 CloudWatch 到 Amazon，以便在 CloudWatch 儀表板中建立操作狀態的統一檢視。您也可以使用警示和觸發動作、量度數學運算和異常偵測等 CloudWatch 功能來監控 S3 Storage Lens 指標並採取行動。此外， CloudWatchAPI 可讓應用程式 (包括第三方供應商) 存取您的 S3 儲存鏡頭指標。如需詳細資訊，請參閱中的監控 S3 儲存鏡頭指標 CloudWatch。	2021 年 11 月 22 日
多區域存取點	您可以使用多區域存取點來建立全域端點，其中應用程式可用來滿足來自位於多個 AWS 區域的 Amazon S3 儲存貯體的請求。您可以使用此多區域存取點，以將資料路由至延遲最低的儲存貯體。如需多區域存取點及其使用方法的詳細資訊，請參閱 Amazon S3 中的多區域存取點。	2021 年 9 月 2 日

Amazon S3 on Outposts 為應用程式新增了直接本機存取	在 AWS Outposts 虛擬私有雲 (VPC) 之外執行您的應用程式，並在 Outposts 資料上存取 S3。您也可以直接透過內部部署網路來存取 S3 on Outposts 物件。如需使用客戶擁有的 IP (CoIP) 地址設定 S3 on Outposts 端點，以及透過建立本機閘道來存取物件的詳細資訊，請參閱使用僅限 VPC 存取點來存取 Amazon S3 on Outposts。	2021 年 7 月 29 日
Amazon S3 存取點別名	在您建立存取點時，Amazon S3 會自動產生可使用的別名，而不是儲存貯體名稱來進行資料存取。您可以在任何存取點資料平面操作中使用此存取點別名，而不是 Amazon 資源名稱 (ARN)。如需詳細資訊，請參閱針對您的存取點使用儲存貯體型別名。	2021 年 7 月 26 日
Amazon S3 庫存和 S3 批次操作支援 S3 儲存貯體金鑰狀態	Amazon S3 庫存和批次操作支援使用 S3 儲存貯體金鑰識別和複製現有物件。S3 儲存貯體金鑰可加速降低現有物件的伺服器端加密成本。如需詳細資訊，請參閱 Amazon S3 庫存和批次操作複製物件。	2021 年 6 月 3 日

Amazon S3 Storage Lens 指標帳戶快照	S3 Storage Lens 帳戶快照會在 S3 主控台首頁 (儲存貯體) 頁面上，透過彙總預設儀表板的指標，顯示您的總儲存空間、物件數量和平均物件大小。如需詳細資訊，請參閱 S3 Storage Lens 指標帳戶快照。	2021 年 5 月 5 日
增加 Amazon S3 on Outposts 端點支援	S3 on Outposts 現在可支援每個 Outposts 最多 100 個端點。如需詳細資訊，請參閱 S3 on Outposts 網路限制。	2021 年 4 月 29 日
Amazon S3 在 Outposts 事件通知在 Amazon CloudWatch 活動	您可以使用 CloudWatch 事件建立規則來擷取 Outposts API 事件上的任何 S3，並透過所有支援的 CloudWatch 目標接收通知。如需詳細資訊，請參閱使 CloudWatch 用事件在 Outposts 事件通知上接收 S3。	2021 年 4 月 19 日
S3 Object Lambda	藉助 S3 Object Lambda，您可將自己的程式碼新增至 Amazon S3 GET 請求，以便在資料傳回應用程式時對其做出修改和處理。您可以使用自訂程式碼，修改標準 S3 GET 請求傳回的資料，以篩選資料列、動態調整影像大小、修訂機密資料以及更多動作。如需詳細資訊，請參閱轉換物件。	2021 年 3 月 18 日

AWS PrivateLink	AWS PrivateLink 對於 Amazon S3，您可以使用虛擬私有雲 (VPC) 中的界面端點直接連接到 S3，而不是透過網際網路連線。可以從內部部署或不同 AWS 區域的應用程式中，直接存取界面端點。如需詳細資訊，請參閱 Amazon S3 的 AWS PrivateLink。	2021 年 2 月 2 日
在 Outposts 容量上管理 Amazon S3 AWS CloudTrail	Outposts 上的 S3 管理事件可透過 CloudTrail 日誌取得。如需詳細資訊，請參 Outposts 使用 CloudTrail.	2020 年 12 月 21 日
高度的一致性	Amazon S3 為 S3 儲存貯體中的物件提供了強大的 read-after-write 一致性PUT和DELETE請求 AWS 區域。此外，Amazon S3 Select、Amazon S3 存取控制清單、Amazon S3 物件標籤和物件中繼資料 (例如，HEAD 物件) 上的讀取操作均高度一致。如需詳細資訊，請參閱 Amazon S3 資料一致性模式。	2020 年 12 月 1 日
Amazon S3 複本修改同步	Amazon S3 複本修改同步可讓物件中繼資料 (例如標籤、ACL 和物件鎖定設定) 在來源物件和複本之間保持同步。啟用此功能時，Amazon S3 會複寫對來源物件或複本複本所做的中繼資料變更。如需詳細資訊，請參閱使用複本修改同步複寫中繼資料變更。	2020 年 12 月 1 日

Amazon S3 儲存貯體金鑰	Amazon S3 儲存貯體金鑰使用 AWS Key Management Service (SSE-KMS) 降低 Amazon S3 伺服器端加密的成本。這個新的伺服器端加密儲存貯體層級金鑰可透過減少從 Amazon S3 到 AWS KMS 的請求流量，降低高達 99% 的 AWS KMS 請求成本。如需詳細資訊，請參閱使用 S3 儲存貯體金鑰降低 SSE-KMS 成本。	2020 年 12 月 1 日
Amazon S3 Storage Lens	S3 Storage Lens 會彙總您的指標，並在 Amazon S3 主控台 Buckets (儲存貯體) 頁面上的 Account snapshot (帳戶快照) 區段中顯示資訊。S3 Storage Lens 也會提供互動式儀表板，您可以用來以視覺化方式呈現洞見與趨勢、標記異常值，以及接收最佳化儲存成本和套用資料保護最佳實務的建議。您的儀表板具有向下切入選項可產生及視覺化組織、帳戶、AWS 區域、儲存類別、儲存貯體、字首或 Storage Lens 群組層級的洞察。您也可以將 CSV 或 Parquet 格式的每日指標匯出傳送到 S3 儲存貯體。如需詳細資訊，請參閱使用 S3 Storage Lens 評估儲存活動和用量。	2020 年 11 月 18 日

使用追蹤 S3 請求 AWS X-Ray	Amazon S3 與 X-Ray 整合以傳播追蹤內容，並為您提供一個具有上游和下游節點的請求鏈。如需詳細資訊，請參閱使用 X-Ray 追蹤要求。	2020 年 11 月 16 日
S3 複寫指標	S3 複寫指標提供了複寫組態中複寫規則的詳細指標。如需詳細資訊，請參閱複寫指標和 Amazon S3 事件通知。	2020 年 11 月 9 日
S3 智慧型分層 Archive Access 和 Deep Archive Access	S3 Intelligent-Tiering Archive Access 和 Deep Archive Access 是 S3 Intelligent-Tiering 的額外儲存方案。如需詳細資訊，請參閱自動最佳化經常存取物件與不常存取物件的儲存體方案。	2020 年 11 月 9 日
刪除標記複寫	借助刪除標記複寫，您可以讓複寫規則確保將刪除標記複寫到目標儲存貯體中。如需詳細資訊，請參閱使用刪除標記複寫。	2020 年 11 月 9 日
S3 物件擁有權	物件擁有權是一項 S3 儲存貯體設定，可讓您針對上傳至儲存貯體的新物件來控制其擁有權。如需詳細資訊，請參閱使用 S3 物件擁有權。	2020 年 10 月 2 日

Amazon S3 on Outposts	使用 Amazon S3 on Outposts，您可以在 AWS Outposts 資源上建立 S3 儲存貯體，並針對需要本機資料存取、本機資料處理和資料存放的應用程式輕鬆存放和擷取物件現場部署。您可以透過 AWS Management Console、 AWS CLI、 AWS 開發套件或 REST API 在 Outposts 上使用 S3。如需詳細資訊，請參閱使用 Amazon S3 on Outposts。	2020 年 9 月 30 日
儲存貯體擁有者條件	您可以使用 Amazon S3 儲存貯體擁有者條件，確保您在 S3 操作中使用的儲存貯體屬於您預期的 AWS 帳戶 儲存貯體。如需詳細資訊，請參閱儲存貯體擁有者條件。	2020 年 9 月 11 日
S3 批次操作支援物件鎖定保留	您現在可以搭配 S3 物件鎖定使用批次作業，一次將保留設定套用至許多 Amazon S3 物件。如需詳細資訊，請參閱使用 S3 批次作業來設定 S3 物件鎖定保留日期。	2020 年 5 月 4 日
S3 批次操作支援物件鎖定法務保存	您現在可以搭配 S3 物件鎖定使用批次作業，一次為許多 Amazon S3 物件新增合法保留。如需詳細資訊，請參閱使用 S3 批次作業來設定 S3 物件鎖定法務保存。	2020 年 5 月 4 日

S3 批次操作的任務標籤	您可以將標籤新增至 S3 批次操作任務，以控制和標記這些任務。如需詳細資訊，請參閱 S3 批次操作任務的標籤。	2020 年 3 月 16 日
Amazon S3 存取點	Amazon S3 存取點針對 S3 中的共用資料集，簡化管理大規模的資料存取。存取點為連接到儲存貯體的指定網路端點，您可以使用這些端點來執行 S3 物件操作。如需詳細資訊，請參閱使用 Amazon S3 存取點來管理資料存取。	2019 年 12 月 2 日
Access Analyzer for Amazon S3	Amazon S3 存取分析器會向您發出設定為允許存取網際網路或其他 AWS 帳戶任何人 (包括組織外部帳戶) 的 S3 儲存貯體提醒您。如需詳細資訊，請參閱使用 Access Analyzer for Amazon S3。	2019 年 12 月 2 日
S3 複寫時間控制 (S3 RTC)	S3 複寫時間控制 (S3 RTC) 會在數秒內複寫您上傳至 Amazon S3 的多數物件，以及在 15 分鐘內複寫 99.99% 的這些物件。如需詳細資訊，請參閱使用 S3 複寫時間控制 (S3 RTC) 複寫物件。	2019 年 11 月 20 日
相同區域複寫	您可以使用相同區域複寫 (SRR)，在相同 AWS 區域中跨 Amazon S3 儲存貯體複製物件。如需跨區域複寫 (CRR) 和相同區域複寫的相關資訊，請參閱複寫。	2019 年 9 月 18 日

跨區域複寫支援 S3 物件鎖定	跨區域複寫現在支援物件鎖定。如需詳細資訊，請參閱 Amazon S3 會複寫什麼？。	2019 年 5 月 28 日
S3 批次操作	您可以使用 S3 批次操作，對 Amazon S3 物件執行大規模的批次操作。S3 批次操作可以對您指定的物件清單執行單一作業。單一任務可在包含數 EB 資料的數以億計物件上執行指定的操作。如需詳細資訊，請參閱執行 S3 批次作業。	2019 年 4 月 30 日
亞太區域 (香港) 區域	Amazon S3 現已在亞太區域 (香港) 區域提供。如需 Amazon S3 區域與端點的詳細資訊，請參閱《AWS 一般參考》中的區域與端點。	2019 年 4 月 24 日
已新增新的欄位至伺服器存取日誌	Amazon S3 將下列新欄位新增至伺服器存取日誌：Transport Layer Security (TLS) 版本。如需詳細資訊，請參閱伺服器存取日誌格式。	2019 年 3 月 28 日
新的存檔儲存體方案	Amazon S3 現已提供新的封存儲存體方案：S3 Glacier Deep Archive (DEEP_ARCHIVE)，用於存放不常存取的物件。如需詳細資訊，請參閱儲存類別。	2019 年 3 月 27 日

已新增新的欄位至伺服器存取日誌	Amazon S3 將下列新欄位新增至伺服器存取日誌：主機 ID、簽章版本、密碼套件、身分驗證類型和主機標頭。如需詳細資訊，請參閱伺服器存取日誌格式。	2019 年 3 月 5 日
支援 Parquet 格式的 Amazon S3 庫存檔案	針對庫存輸出檔案，Amazon S3 現在支援 Apache Parquet (Parquet) 格式以及 Apache 最佳化資料列單欄式 (ORC) 和逗點分隔值 (CSV) 檔案格式。如需詳細資訊，請參閱庫存。	2018 年 12 月 4 日
S3 物件鎖定	Amazon S3 現已推出物件鎖定功能，為 Amazon S3 物件提供單寫多讀 (WORM) 保護。如需詳細資訊，請參閱鎖定物件。	2018 年 11 月 26 日
還原速度升級	從 S3 Glacier Flexible Retrieval 儲存類別還原時，在還原進行當中，您可以使用 Amazon S3 還原速度升級來加快速度。如需詳細資訊，請參閱還原封存物件。	2018 年 11 月 26 日
還原事件通知	從 S3 Glacier Flexible Retrieval 儲存類別還原物件時，Amazon S3 事件通知現在支援起始和完成事件。如需詳細資訊，請參閱事件通知。	2018 年 11 月 26 日

直接 PUT 到 S3 Glacier Flexible Retrieval 儲存類別	Amazon S3 PUT 操作現在支援於建立物件時，指定 S3 Glacier Flexible Retrieval 作為儲存類別。從前，必須將物件從其他 Amazon S3 儲存類別轉換為 S3 Glacier Flexible Retrieval 儲存類別。此外，現在使用 S3 跨區域複寫 (CRR) 時，您也可以指定 S3 Glacier Flexible Retrieval 作為複寫物件的儲存類別。如需 S3 Glacier Flexible Retrieval 儲存類別的詳細資訊，請參閱儲存類別。如需為複寫物件指定儲存類別的詳細資訊，請參閱複寫組態概觀。如需直接 PUT 到 S3 Glacier Flexible Retrieval REST API 變更的詳細資訊，請參閱文件歷史記錄：直接 PUT 到 S3 Glacier Flexible Retrieval。	2018 年 11 月 26 日
新的儲存體方案	Amazon S3 現在提供新的儲存類別，名為 S3 智慧型分層服務 (`INTELLIGENT_TIERING`)，專為存取模式變化多端或未知的長期資料所設計。如需詳細資訊，請參閱儲存類別。	2018 年 11 月 26 日
Amazon S3 封鎖公開存取	Amazon S3 現在能夠針對每個儲存貯體或全帳戶，封鎖對儲存貯體和物件的公開存取。如需詳細資訊，請參閱使用 Amazon S3 封鎖公開存取。	2018 年 11 月 15 日

強化跨區域複寫 (CRR) 規則的篩選條件	在 CRR 規則組態中，您可以指定物件篩選條件，選擇要套用規則的一部分物件。從前只能篩選一個物件金鑰前綴。在本版中，您可以指定一個物件金鑰前綴、一或多個物件標籤或兩者皆選的篩選條件。如需詳細資訊，請參閱 CRR 設定：複寫組態概觀。	2018 年 9 月 19 日
新的 Amazon S3 Select 功能	Amazon S3 Select 現在支援 Apache 實木複合地板輸入、巢狀 JSON 物件的查詢，以及兩個新的 Amazon CloudWatch 監控指標 (`SelectScannedBytes` 和`SelectReturnedBytes`)。	2018 年 9 月 5 日
現在可以透過 RSS 獲得更新	您現在可以訂閱更新 RSS 訊息，以接收《Amazon S3 使用者指南》的更新通知。	2018 年 6 月 19 日

舊版更新

下表會說明 2018 年 6 月 19 日之前，《Amazon S3 使用者指南》每個版本的重要變更。

變更	描述	日期
程式碼範例更新	程式碼範例已更新： • C# — 將所有範例更新為使用任務型非同步模式。如需詳細資訊，請參閱 AWS SDK for .NET 開發人員指南中的適用於 .NET 的 Amazon Web Services 非同步 API。程式碼範例現在可與 AWS SDK for .NET第 3 版相容。 •	2018 年 4 月 30 日

變更	描述	日期
	Java — 將所有範例更新為使用用戶端產生器模型。如需用戶端產生器模型的詳細資訊，請參閱建立服務用戶端。 • PHP – 將所有範例更新為使用 AWS SDK for PHP 3.0。若要取得有關 AWS SDK for PHP 3.0 的更多資訊，請參閱AWS SDK for PHP。 • 紅寶石-已更新範例程式碼，以便範例可與 AWS SDK for Ruby 版本 3 搭配使用。	
Amazon S3 現在向 Amazon CloudWatch 日誌儲存指標報告 S3 冰川彈性擷取和ONEZONE_IA 儲存類別	除了報告使用實際位元組外，這些儲存指標包含適用儲存體類別 (ONEZONE_IA 、STANDARD_IA 和 S3 Glacier Flexible Retrieval) 的每物件額外負荷位元組。 • 對於 ONEZONE_IA 和 STANDARD_IA 儲存體類別物件，Amazon S3 會將小於 128 KB 的物件報告為 128 KB 的大小。如需詳細資訊，請參閱 使用 Amazon S3 儲存體方案。 • 對於 S3 Glacier Flexible Retrieval 儲存類別物件，儲存指標會報告下列額外負荷： • 每個物件 32 KB 額外負荷，以 S3 Glacier Flexible Retrieval 儲存類別定價計費 • 每個 8 KB 物件額外負荷，以 STANDARD 儲存體類別收費 如需詳細資訊，請參閱 使用 Amazon S3 生命週期轉換物件。 如需更多有關儲存指標的詳細資訊，請參閱「使用 Amazon 監控指標 CloudWatch」。	2018 年 4 月 30 日

變更	描述	日期
新的儲存體方案	Amazon S3 現已提供新的儲存體類別 STANDARD_IA (IA 表示不常存取) 來存放物件。此儲存體方案已針對長時間及較少存取的資料進行了最佳化。如需詳細資訊，請參閱 使用 Amazon S3 儲存體方案。	2018 年 4 月 4 日
Amazon S3 Select	Amazon S3 現在支援根據 SQL 運算式擷取物件內容。如需詳細資訊，請參閱 使用 Amazon S3 Select 篩選和擷取資料。	2018 年 4 月 4 日
亞太 (大阪當地) 區域	Amazon S3 現已在亞太區域 (大阪 – 當地) 區域提供。如需 Amazon S3 區域與端點的詳細資訊，請參閱《AWS 一般參考》中的區域與端點。 **Important** 亞太區域 (大阪 – 當地) 區域只能搭配亞太區域 (東京) 區域一起使用。若要請求存取亞太區域 (大阪 – 當地) 地區，請聯絡您的銷售代表。	2018 年 2 月 12 日
Amazon S3 庫存建立時間戳記	Amazon S3 庫存現在包含 Amazon S3 庫存報告建立起始時間日期的時間戳記。您可以使用時間戳記，從產生庫存報告的起始時間起，查明 Amazon S3 儲存體的變更。	2018 年 1 月 16 日
歐洲 (巴黎) 區域	Amazon S3 現已在歐洲 (巴黎) 區域提供。如需 Amazon S3 區域與端點的詳細資訊，請參閱《AWS 一般參考》中的區域與端點。	2017 年 12 月 18 日
中國 (寧夏) 區域	Amazon S3 現已在中國 (寧夏) 區域提供。如需 Amazon S3 區域與端點的詳細資訊，請參閱《AWS 一般參考》中的區域與端點。	2017 年 11 月 29 日

變更	描述	日期
支援 ORC 格式的 Amazon S3 庫存檔案	Amazon S3 現在除了逗號分隔值 (CSV) 檔案格式，也支援 Apache 最佳化行列式 (ORC) 格式的庫存輸出檔案。您現在也可以使用 Amazon Athena、Amazon Redshift Spectrum，以及 Presto、Apache Hive 和 Apache Spark 等其他工具，利用標準 SQL 來查詢 Amazon S3 庫存。如需詳細資訊，請參閱 Amazon S3 清查。	2017 年 11 月 17 日
S3 儲存貯體的預設加密	Amazon S3 預設加密提供為 S3 儲存貯體設定預設加密行為的方式。您可以在儲存貯體上設定預設加密，讓所有物件在存放於儲存貯體中時維持加密狀態。物件會使用伺服器端加密與 Amazon S3 受管金鑰 (SSE-S3) 或 AWS 受管金鑰 (SSE-KMS) 加密。如需詳細資訊，請參閱 對 Amazon S3 儲存貯體設定預設伺服器端加密行為。	2017 年 11 月 6 日
Amazon S3 庫存中的加密狀態	Amazon S3 現在支援在 Amazon S3 庫存中包含加密狀態，讓您可以針對合規稽核或其他用途檢視物件的靜態加密方式。您也可以設定為使用伺服器端加密 (SSE) 或 SSE-KMS 對 Amazon S3 庫存進行加密，讓所有庫存檔案都能據以加密。如需詳細資訊，請參閱 Amazon S3 清查。	2017 年 11 月 6 日
跨區域複寫 (CRR) 強化	跨區域複寫現在支援以下： • 在跨帳戶案例中，您可以新增 CRR 組態，將複本擁有權變更為擁有目的地儲存貯體的 AWS 帳戶 。如需詳細資訊，請參閱 變更複本擁有者。 • 根據預設，Amazon S3 不會複寫使用存放在 CRR 組態中的金鑰使用伺服器端加密建立的來源儲存貯體 AWS KMS 中的物件，您現在可以指示 Amazon S3 複寫這些物件。如需詳細資訊，請參閱 複寫加密的物件 (SSE-C、SSE-S3、SSE-KMS、DSSE-KMS)。	2017 年 11 月 6 日
Europe (London) Region	Amazon S3 現已在歐洲 (倫敦) 區域提供。如需 Amazon S3 區域與端點的詳細資訊，請參閱《AWS 一般參考》中的區域與端點。	2016 年 12 月 13 日

變更	描述	日期
Canada (Central) Region	Amazon S3 現已在加拿大 (中部) 區域提供。如需 Amazon S3 區域與端點的詳細資訊，請參閱《AWS 一般參考》中的區域與端點。	2016 年 12 月 8 日
物件標記	Amazon S3 現在支援物件標記。物件標記可讓您分類儲存。您也可以使用物件金鑰名稱的字首來分類儲存，物件標記會為其新增其他維度。 標記提供的新增優點。其中包含： • 物件標籤可實現精細的存取控制許可 (例如，您可利用特定標籤授予 IAM 使用者唯讀物件的許可)。 • 指定生命週期組態的精細控制。您可以指定標籤，選擇要套用生命週期規則的一部分物件。 • 如已設定跨區域複寫 (CRR)，Amazon S3 即可複寫標籤。您必須將必要許可授予已建立供 Amazon S3 擔任的 IAM 角色，以代替您複寫物件。 • 您也可以自訂 CloudWatch 指標和 CloudTrail 事件，依特定標籤篩選器顯示資訊。 如需詳細資訊，請參閱 使用標籤分類儲存空間。	2016 年 11 月 29 日
Amazon S3 生命週期現在支援以標籤為基礎的篩選條件	在生命週期組態中，Amazon S3 現在支援以標籤為基礎的篩選。您現在可以指定生命週期規則，在其中指定金鑰前綴、一或多個物件標籤，或這兩項的組合，來選取要套用生命週期規則的物件子集。如需詳細資訊，請參閱 管理儲存生命週期。	2016 年 11 月 29 日

變更	描述	日期
CloudWatch 值區的要求量度	Amazon S3 現在支援在儲存貯體上發出的請求的 CloudWatch 指標。當您啟用這些儲存貯體指標時，指標回報的時間間隔為 1 分鐘。您也可以設定儲存貯體中回報這些要求指標的物件。如需詳細資訊，請參閱 使用 Amazon 監控指標 CloudWatch。	2016 年 11 月 29 日
Amazon S3 庫存	Amazon S3 現在支援儲存庫存。Amazon S3 庫存每日或每週為 S3 儲存貯體或共用字首 (亦即名稱以共同字串開頭的物件)，提供物件及其相對應中繼資料的一般檔案輸出。 如需詳細資訊，請參閱 Amazon S3 清查。	2016 年 11 月 29 日
Amazon S3 分析 – 儲存類別分析	新的 Amazon S3 分析 – 儲存體類別分析功能會觀察資料存取模式，協助您判斷何時將不常存取的 STANDARD 儲存體轉移至 STANDARD_IA (IA 表示不常存取) 儲存體類別。在儲存體方案分析觀察一段時間之已篩選資料集的不常存取模式後，您可以使用分析結果協助改善生命週期組態。這項功能也包含可匯出至 S3 儲存貯體的指定儲存貯體、字首或標籤層級儲存體用量的詳細每日分析。	2016 年 11 月 29 日
新增從 S3 Glacier 還原封存物件時的「加速」與「大量」資料擷取	當還原封存在 S3 Glacier 的物件時，Amazon S3 現在除了標準擷取外，還支援「加速」與「大量」資料擷取。如需詳細資訊，請參閱 還原已封存的物件。	2016 年 11 月 21 日
CloudTrail 物件記錄	CloudTrail 支援記錄 Amazon S3 物件層級 API 操作GetObject ，例如PutObject 、和DeleteObject 。您可以設定事件選擇器記錄物件層級的 API 操作。如需詳細資訊，請參閱 使用記錄 Amazon S3 API 呼叫 AWS CloudTrail。	2016 年 11 月 21 日
US East (Ohio) Region	Amazon S3 現已在美國東部 (俄亥俄) 區域提供。如需 Amazon S3 區域與端點的詳細資訊，請參閱《AWS 一般參考》中的區域與端點。	2016 年 10 月 17 日

變更	描述	日期
Amazon S3 Transfer Acceleration 的 IPv6 支援	對於 Amazon S3 Transfer Acceleration，Amazon S3 現在支援網際網路通訊協定第 6 版 (IPv6)。您可以使用 Transfer Acceleration 端點的新雙堆疊，透過 IPv6 連接到 Amazon S3。如需詳細資訊，請參閱 Amazon S3 Transfer Acceleration 入門。	2016 年 10 月 6 日
IPv6 支援	Amazon S3 現在支援網際網路通訊協定第 6 版 (IPv6)。您可以使用雙堆疊端點透過 IPv6 存取 Amazon S3。如需詳細資訊，請參閱 透過 IPv6 向 Amazon S3 提出請求。	2016 年 8 月 11 日
Asia Pacific (Mumbai) Region	Amazon S3 現已在亞太區域 (孟買) 區域提供。如需 Amazon S3 區域與端點的詳細資訊，請參閱《AWS 一般參考》中的區域與端點。	2016 年 6 月 27 日
Amazon S3 Transfer Acceleration	Amazon S3 Transfer Acceleration 可讓用戶端與 S3 儲存貯體間的長距離檔案傳輸變得迅速、簡單又安全。傳輸加速充分利用 Amazon CloudFront 全球分佈的節點。 如需詳細資訊，請參閱 使用 Amazon S3 Transfer Acceleration 設定快速安全的檔案傳輸。	2016 年 4 月 19 日
生命週期支援移除過期物件刪除標記	生命週期組態 `Expiration` 動作現在可讓您指示 Amazon S3，移除版本化儲存貯體中過期的物件刪除標記。如需詳細資訊，請參閱 描述生命週期動作的元素。	2016 年 3 月 16 日

變更	描述	日期
儲存貯體生命週期組態現在支援停止未完成的分段上傳動作。	儲存貯體生命週期組態現在支援 `AbortIncompleteMultipartUpload` 動作，可用以指示 Amazon S3 停止在開始後一定天數內未完成的分段上傳。當分段上傳符合停止操作的條件時，Amazon S3 會刪除任何已上傳的部分並停止分段上傳。 如需概念資訊，請參閱《Amazon S3 使用者指南》中的下列主題： • 中止分段上傳 • 描述生命週期動作的元素 已更新下列 API 操作支援新的動作： • PUT 儲存貯體生命週期 – XML 組態現在可讓您在生命週期組態規則中指定 `AbortIncompleteMultipartUpload` 動作。 • 列出片段與啟動分段上傳 – 如果儲存貯體的生命週期規則指定 `x-amz-abort-date` 動作，這兩項 API 操作現在會傳回兩個額外的回應標頭 (`x-amz-abort-rule-id` 與 `AbortIncompleteMultipartUpload`)。這些回應標頭指出何時能夠停止已啟動的分段上傳操作，以及可套用的生命週期規則。	2016 年 3 月 16 日
Asia Pacific (Seoul) Region	Amazon S3 現已在亞太區域 (首爾) 區域提供。如需 Amazon S3 區域與端點的詳細資訊，請參閱《AWS 一般參考》中的區域與端點。	2016 年 1 月 6 日

變更	描述	日期
新的條件金鑰和分段上傳變更	IAM 政策現在支援 Amazon S3 `s3:x-amz-storage-class` 條件金鑰。如需詳細資訊，請參閱 使用條件鍵值區政策範例。 您再也不需要分段上傳啟動器來上傳分段並完成上傳。如需詳細資訊，請參閱 分段上傳 API 與許可。	2015 年 12 月 14 日
已重新命名美國標準區域	區域名稱已從「美國標準」變更為「美國東部 (維吉尼亞北部)」。這只是區域名稱變更，功能沒有任何改變。	2015 年 12 月 11 日
新的儲存體方案	Amazon S3 現已提供新的儲存類別 STANDARD_IA (IA 表示不常存取) 來存放物件。此儲存體方案已針對長時間及較少存取的資料進行了最佳化。如需詳細資訊，請參閱 使用 Amazon S3 儲存體方案。 生命週期組態功能更新，現已允許您將物件轉換到標準型 (IA) 儲存體方案。如需詳細資訊，請參閱 管理儲存生命週期。 跨區域複寫功能先前使用來源物件的儲存體方案，進行物件複寫。現在，當您設定跨區域複寫時，可以為目標儲存貯體中所建立的物件複本，指定儲存體方案。如需詳細資訊，請參閱 複製物件概觀。	2015 年 9 月 16 日
AWS CloudTrail 整合	新的 AWS CloudTrail 整合可讓您在 S3 儲存貯體中記錄 Amazon S3 API 活動。您可以用 CloudTrail 來追蹤 S3 儲存貯體的建立或刪除、存取控制修改或生命週期組態變更。如需詳細資訊，請參閱 使用記錄 Amazon S3 API 呼叫 AWS CloudTrail。	2015 年 9 月 1 日

變更	描述	日期
提升儲存貯體限制	Amazon S3 現在支援提高儲存貯體限制。根據預設，客戶最多可以在其中建立 100 個儲存貯體 AWS 帳戶。需要更多儲存貯體的客戶，可以提交提升服務限制來增加儲存貯體限制。如需如何增加儲存貯體限制的資訊，請參閱《AWS 一般參考》中的 AWS 服務 配額。如需詳細資訊，請參閱 使用 AWS 軟體開發套件 及 儲存貯體的法規與限制。	2015 年 8 月 4 日
一致性模式更新	Amazon S3 現在支援在美國東部 (維吉尼亞北部) 區域新增至 Amazon S3 的新物件的 read-after-write 一致性。在此更新之前，除美國東部 (維吉尼亞北部) 區域以外的所有區域都支援上傳到 Amazon S3 的新物件的read-after-write 一致性。透過此增強功能，Amazon S3 現在支援新增至 Amazon S3 的新物件在所有區域的 read-after-write 一致性。R ead-after-write 一致性可讓您在 Amazon S3 中建立物件後立即擷取物件。如需詳細資訊，請參閱 區域。	2015 年 8 月 4 日
事件通知	Amazon S3 事件通知已更新，新增當物件刪除時通知，並新增依字首及字尾比對來篩選物件名稱。如需詳細資訊，請參閱 Amazon S3 事件通知。	2015 年 7 月 28 日
Amazon CloudWatch 整合	全新的 Amazon CloudWatch 整合可讓您透過 Amazon S3 的 CloudWatch 指標監控和設定 Amazon S3 使用情況的警示。支援的指標包含標準儲存的總位元組、低冗餘儲存的總位元組，以及指定 S3 儲存貯體的總物件數。如需詳細資訊，請參閱 使用 Amazon 監控指標 CloudWatch。	2015 年 7 月 28 日
支援刪除及清空非空白的儲存貯體	Amazon S3 現在支援刪除及清空非空白的儲存貯體。如需詳細資訊，請參閱 清空儲存貯體。	2015 年 7 月 16 日
Amazon VPC 端點的儲存貯體政策	Amazon S3 新增支援 Virtual Private Cloud (VPC) 端點的儲存貯體政策。您可以使用 S3 儲存貯體政策，控制從特定的 VPC 端點或特定的 VPC 對儲存貯體的存取。VPC 端點的設定方式非常簡單，具高穩定性，能為 Amazon S3 提供安全的連線，無須使用閘道或 NAT 執行個體。如需詳細資訊，請參閱 使用儲存貯體政策控制來自 VPC 端點的存取。	2015 年 4 月 29 日

變更	描述	日期
事件通知	Amazon S3 事件通知已更新，以支援切換到以資源為基礎的 AWS Lambda 功能許可。如需詳細資訊，請參閱 Amazon S3 事件通知。	2015 年 4 月 9 日
跨區域複寫	Amazon S3 現在支援跨區域複寫。跨區域複寫是跨不同值區中物件的自動非同 AWS 區域步複製。如需詳細資訊，請參閱 複製物件概觀。	2015 年 3 月 24 日
事件通知	在儲存貯體通知組態中，Amazon S3 現在支援新的事件類型和目的地。在此版本發行之前，Amazon S3 僅支援 s3：ReducedRedundancyLostObject事件類型和 Amazon SNS 主題做為目的地。如需新事件類型的詳細資訊，請參閱「Amazon S3 事件通知」。	2014 年 11 月 13 日
使用客戶提供加密金鑰的伺服器端加密	使用 AWS Key Management Service (AWS KMS) 金鑰 (SSE-KMS) 進行伺服器端加密 Amazon S3 現在支援使用 AWS KMS. 此功能可讓您透過管理信封金鑰 AWS KMS，並透過 Amazon S3 呼叫 AWS KMS 來存取您設定的許可內的信封金鑰。 如需伺服器端加密的詳細資訊 AWS KMS，請參閱使用伺服器端加密保護資料 AWS Key Management Service。	2014 年 11 月 12 日
Europe (Frankfurt) Region	Amazon S3 現已在歐洲 (法蘭克福) 區域提供。	2014 年 10 月 23 日
使用客戶提供加密金鑰的伺服器端加密	Amazon S3 現在支援使用客戶提供加密金鑰 (SSE-C) 的伺服器端加密。伺服器端加密可讓您要求 Amazon S3 加密靜態資料。使用 SSE-C 時，Amazon S3 會使用您提供的自訂加密金鑰加密您的物件。因為 Amazon S3 為您執行加密，所以您可以使用自己的加密金鑰，不用費心撰寫或執行自己的加密程式碼。 如需 SSE-C 的詳細資訊，請參閱伺服器端加密 (使用客戶提供的加密金鑰)。	2014 年 6 月 12 日

變更	描述	日期
版本控制的生命週期支援	在這個版本之前，僅未使用版本控制的儲存貯體支援生命週期組態。現在未使用版本控制及已啟用版本控制的儲存貯體都可以設定生命週期。如需詳細資訊，請參閱 管理儲存生命週期。	2014 年 5 月 20 日
已修訂存取控制主題	已修訂的 Amazon S3 存取控制文件。如需詳細資訊，請參閱 適用於 Amazon S3 的 Identity and Access Management。	2014 年 4 月 15 日
已修訂伺服器存取記錄主題	已修訂伺服器存取記錄文件。如需詳細資訊，請參閱 使用伺服器存取記錄記錄要求。	2013 年 11 月 26 日
.NET SDK 範例已更新至 2.0 版	本指南的 .NET SDK 範例現已相容於 2.0 版。	2013 年 11 月 26 日
HTTP 上的 SOAP 支援已淘汰	HTTP 上的 SOAP 支援已淘汰，但仍可透過 HTTPS 取得。SOAP 不支援新的 Amazon S3 功能。我們建議您使用其他 API 或 AWS 開發套件。	2013 年 9 月 20 日
IAM 政策變數支援	IAM 政策語言現在支援變數。評估一項政策時，任何政策變數都會被來自已通過身分驗證的使用者工作階段的內容型資訊所提供的值所取代。您可使用政策變數定義一般用途政策，並不需明確列出政策的所有元件。如需政策變數的詳細資訊，請參閱《IAM 使用者指南》中的 IAM 政策變數概觀。 如需 Amazon S3 的政策變數範例，請參閱Amazon S3 的基於身分識別的政策範例。	2013 年 4 月 3 日
申請者付款的主控台支援	您現在可以使用 Amazon S3 主控台設定申請者付款的儲存貯體。如需詳細資訊，請參閱 使用儲存體傳輸和用量的申請者付款儲存貯體。	2012 年 12 月 31 日

變更	描述	日期
網站託管的根網域支援	Amazon S3 現在支援在根網域託管靜態網站。您網站的造訪者可以從他們的瀏覽器中存取您的網站，而無需在網址中指定 www (例如，他們可以使用 example.com，而不是 www.example.com)。許多客戶已在 Amazon S3 上託管靜態網站，這些網站可從 www 子網域存取 (例如，www.example.com)。在先前的版本中，若要支援根網域存取，您必須執行自己的 Web 伺服器，將來自瀏覽器的根網域請求，代理至您在 Amazon S3 上的網站。執行代理要求的 Web 伺服器可能會產生額外的成本、操作負擔，以及其他潛在失敗點。現在，www 與根網域位址都可以利用 Amazon S3 的高可用性與耐用性。如需詳細資訊，請參閱使用 Amazon S3 託管靜態網站。	2012 年 12 月 27 日
主控台修訂	Amazon S3 主控台已更新。涉及主控台的文件主題也已修訂完成。	2012 年 12 月 14 日
支援將資料封存至 S3 Glacier	Amazon S3 現在支援一種儲存選項，可讓您利用 S3 Glacier 的低成本儲存服務來封存資料。若要封存物件，您需要定義封存規則來識別物件，並定義時間範圍來決定 Amazon S3 何時將這些物件封存到 S3 Glacier。您可以使用 Amazon S3 主控台或使用 Amazon S3 API 或 AWS 開發套件以程式設計方式輕鬆地在儲存貯體上設定規則。 如需詳細資訊，請參閱 管理儲存生命週期。	2012 年 11 月 13 日
網頁重新導向支援	對於設定為網站的儲存貯體，Amazon S3 現在支援將物件請求重新導向至相同儲存貯體中的其他物件或外部 URL。如需詳細資訊，請參閱 (選用) 配置網頁重新導向。 如需託管網站的資訊，請參閱「使用 Amazon S3 託管靜態網站」。	2012 年 10 月 4 日

變更	描述	日期
支援跨來源資源共享 (CORS)	Amazon S3 現在支援跨來源資源分享 (CORS)。CORS 會定義一種方式，讓載入同一個網域的用戶端 Web 應用程式，可以與不同網域中的資源互動或加以存取。透過 Amazon S3 的 CORS 支援，您可以在 Amazon S3 上建立功能豐富的用戶端 Web 應用程式，也可以選擇允許跨來源存取您的 Amazon S3 資源。如需詳細資訊，請參閱 使用跨來源資源分享 (CORS)。	2012 年 8 月 31 日
支援成本分配標記	Amazon S3 現在支援成本分配標記，讓您標記 S3 儲存貯體，以便更容易根據專案或其他條件追蹤其成本。如需使用儲存貯體標記的詳細資訊，請參閱「使用成本分配 S3 儲存貯體標籤」。	2012 年 8 月 21 日
支援儲存貯體政策中的 MFA 保護 API 存取	Amazon S3 現在支援 MFA 保護的 API 存取，這項功能可在存取 Amazon S3 資源時強制執行 AWS 多重要素身份驗證，以提供額外的安全層級。此安全功能需要使用者提供有效的 MFA 代碼來證明 MFA 裝置的實體擁有。如需詳細資訊，請參閱 AWS 多重要素驗證。有任何請求存取 Amazon S3 資源時，您現在都可以要求 MFA 身分驗證。 為了強制執行 MFA 身分驗證，Amazon S3 現在於儲存貯體政策中支援 `aws:MultiFactorAuthAge` 金鑰。如需儲存貯體政策範例，請參閱「需要 MFA」。	2012 年 7 月 10 日
物件過期支援	您可以使用物件過期，在設定的期間之後，排程自動移除資料。將生命週期組態新增至儲存貯體以設定物件過期。	2011 年 12 月 27 日
支援的新區域	Amazon S3 現在支援南美洲 (聖保羅) 區域。如需詳細資訊，請參閱 存取及列出 Amazon S3 儲存貯體。	2011 年 12 月 14 日
刪除多項物件	Amazon S3 現在支援多物件刪除 API，可讓您在單一要求中刪除多個物件。使用此功能，您可以更快速從 Amazon S3 移除大量物件，比使用多個單獨的 DELETE 請求還快。如需詳細資訊，請參閱 刪除 Amazon S3 物件。	2011 年 12 月 7 日

變更	描述	日期
支援的新區域	Amazon S3 現在支援美國西部 (奧勒岡) 區域。如需詳細資訊，請參閱儲存貯體與區域。	2011 年 11 月 8 日
文件更新	文件錯誤修正	2011 年 11 月 8 日
文件更新	除文件錯誤修正之外，本版還包含下列改善： • 使用 AWS SDK for PHP 和的新伺服器端加密區段 AWS SDK for Ruby (請參閱使用 Amazon S3 受管金鑰 (SSE-S3) 指定伺服器端加密)。	2011 年 10 月 17 日
伺服器端加密支援	Amazon S3 現在支援伺服器端加密。可讓您要求 Amazon S3 加密靜態資料，即在 Amazon S3 將資料寫入資料中心磁碟時，加密物件資料。除了 REST API 更新之外， AWS SDK for Java 和 .NET 還提供要求伺服器端加密的必要功能。您也可以在使用 AWS Management Console上傳物件時，請求伺服器端加密。若要進一步了解資料加密，請參閱使用資料加密。	2011 年 10 月 4 日
文件更新	除文件錯誤修正之外，本版還包含下列改善： • 已將 Ruby 與 PHP 範例新增至「提出要求」一節。 • 新增的章節會說明如何產生及使用預先簽章的 URL。如需詳細資訊，請參閱 使用預先簽章的 URL 來共用物件 及使用預先簽章的 URL 來共用物件。 • 更新了一個現有的部分，引入了日食和視覺工作室的資 AWS 源管理器。如需詳細資訊，請參閱 使用開發 AWS 套件使用 Amazon S3 進行開發。	2011 年 9 月 22 日

變更	描述	日期
支援使用暫時性安全登入資料來傳送要求	除了使用您的 AWS 帳戶 和 IAM 使用者安全登入資料將驗證的請求傳送到 Amazon S3 之外，您現在還可以使用從 AWS Identity and Access Management (IAM) 取得的臨時安全登入資料傳送請求。您可以使用 AWS Security Token Service API 或 AWS SDK 包裝函式庫向 IAM 要求這些臨時登入資料。您可以要求這些暫時性安全登入資料供您自己使用，或將它們提供給聯合身分使用者及應用程式。此功能可讓您管理外部的使用者， AWS 並為他們提供臨時安全登入資料以存取您的 AWS 資源。 如需詳細資訊，請參閱 提出要求。 如需 IAM 的臨時安全登入資料支援的詳細資訊，請參閱《IAM 使用者指南》中的臨時安全登入資料。	2011 年 8 月 3 日
已擴充分段上傳 API，使其最多可複製 5 TB 的物件	在此版本之前，Amazon S3 API 支援複製物件的大小最多為 5 GB。為了能複製超過 5 GB 的物件，Amazon S3 現在加上新操作 `Upload Part (Copy)` 來擴充分段上傳 API。您可以使用此分段上傳操作來複製大小最多 5 TB 的物件。如需詳細資訊，請參閱 複製、移動和重新命名物件。 如需分段上傳 API 的概念資訊，請參閱「使用分段上傳來上傳和複製物件」。	2011 年 6 月 21 日
停用透過 HTTP 的 SOAP API 呼叫	為提升安全性，已停用透過 HTTP 的 SOAP API 呼叫。已驗證身分與匿名的 SOAP 請求都必須使用 SSL 傳送至 Amazon S3。	2011 年 6 月 6 日

變更	描述	日期
IAM 啟用跨帳戶委派	之前，若要存取 Amazon S3 資源，IAM 使用者需要父項 AWS 帳戶 和 Amazon S3 資源擁有者的許可。使用跨帳戶存取，IAM 使用者現在只需要具備擁有者帳戶的許可。也就是說，如果資源擁有者授予存取權 AWS 帳戶，則現在 AWS 帳戶 可以授予其 IAM 使用者對這些資源的存取權。 如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以將許可委派給 IAM 使用者。 如需指定儲存貯體政策委託人的詳細資訊，請參閱「值區政策的主體」。	2011 年 6 月 6 日
新連結	此項服務的端點資訊現位於《AWS 一般參考》。如需詳細資訊，請前往 AWS 一般參考中的「區域與端點」。	2011 年 3 月 1 日
支援在 Amazon S3 中託管靜態網站	Amazon S3 加強支援託管靜態網站。這包含支援索引文件及自訂錯誤文件。使用這些功能時，您儲存貯體或子資料夾的根要求 (例如，`http://mywebsite.com/subfolder`) 會傳回您的索引文件，而非您儲存貯體中物件的清單。若發生錯誤，Amazon S3 會傳回您的自訂錯誤訊息，而非 Amazon S3 錯誤訊息。如需詳細資訊，請參閱 使用 Amazon S3 託管靜態網站。	2011 年 6 月 6 日
此項服務的端點資訊現位於《AWS 一般參考》。如需詳細資訊，請前往 AWS 一般參考中的「區域與端點」。	2011 年 3 月 1 日	

變更	描述	日期
支援在 Amazon S3 中託管靜態網站	Amazon S3 加強支援託管靜態網站。這包含支援索引文件及自訂錯誤文件。使用這些功能時，您儲存貯體或子資料夾的根要求 (例如，`http://mywebsite.com/subfolder`) 會傳回您的索引文件，而非您儲存貯體中物件的清單。若發生錯誤，Amazon S3 會傳回您的自訂錯誤訊息，而非 Amazon S3 錯誤訊息。如需詳細資訊，請參閱 使用 Amazon S3 託管靜態網站。	2011 年 2 月 17 日
回應標頭 API 支援	GET 物件 REST API 現在可讓您變更每項要求的 REST GET 物件要求回應標頭。亦即，您可以改變回應中的物件中繼資料，不必改變物件本身。如需詳細資訊，請參閱 下載物件。	2011 年 1 月 14 日
大型物件支援	Amazon S3 已將可存放在 S3 儲存貯體的物件大小上限從 5 GB 增加至 5 TB。如果使用 REST API，單一 PUT 操作最多可以上傳最多 5 GB 的物件。若為大型物件，您必須使用分段上傳 REST API 將物件分段上傳。如需詳細資訊，請參閱 使用分段上傳來上傳和複製物件。	2010 年 12 月 9 日
分段上傳	分段上傳可以更快、更彈性地上傳至 Amazon S3。它讓您將單一物件上傳為一組物件。如需詳細資訊，請參閱 使用分段上傳來上傳和複製物件。	2010 年 11 月 10 日
儲存貯體政策中的正式 ID 支援	您現在可以在儲存貯體政策中指定正式 ID。如需詳細資訊，請參閱 值區政策的主體	2010 年 9 月 17 日
Amazon S3 與 IAM 搭配運作	此服務現在與 AWS Identity and Access Management (IAM) 整合。如需詳細資訊，請參閱《IAM 使用者指南》中的可搭配 IAM 運作的AWS 服務。	2010 年 9 月 2 日
通知	Amazon S3 通知功能可讓您設定儲存貯體，以便在 Amazon S3 偵測到儲存貯體上的重要事件時，讓 Amazon S3 將訊息發佈至 Amazon Simple Notification Service (Amazon SNS) 主題。如需詳細資訊，請參閱設定儲存貯體事件的通知。	2010 年 7 月 14 日

變更	描述	日期
儲存貯體政策	儲存貯體政策是一種存取管理系統，用以設定跨儲存貯體、物件與物件集的存取許可。此功能可補充並在許多情況下取代存取控制清單。如需詳細資訊，請參閱 Amazon S3 的存儲桶政策。	2010 年 7 月 6 日
所有區域皆可使用路徑型語法	對於美國傳統區域的任何儲存貯體，或者與請求端點位於相同區域的儲存貯體，Amazon S3 現在支援路徑型語法。如需詳細資訊，請參閱虛擬託管。	2010 年 6 月 9 日
歐洲 (愛爾蘭) 的新端點	Amazon S3 現在為歐洲 (愛爾蘭) 提供一個端點：`http://s3-eu-west-1.amazonaws.com` 。	2010 年 6 月 9 日
主控台	您現在可以透過 AWS Management Console使用 Amazon S3。請參閱《Amazon Simple Storage Service 使用者指南》，以了解主控台的所有 Amazon S3 功能。	2010 年 6 月 9 日
低冗餘	Amazon S3 現在能利用低冗餘將物件存放在 Amazon S3 中，讓您降低儲存成本。如需詳細資訊，請參閱低冗餘儲存。	2010 年 5 月 12 日
支援的新區域	Amazon S3 現在支援亞太區域 (新加坡) 區域。如需詳細資訊，請參閱儲存貯體與區域。	2010 年 4 月 28 日
物件版本控制	此版本介紹物件版本控制。所有物件現在都會有金鑰與版本。如果您對儲存貯體啟用版本控制，Amazon S3 會為所有新增至儲存貯體的物件提供唯一的版本 ID。這項功能可讓您復原不小心的覆寫與刪除。如需詳細資訊，請參閱版本控制與使用版本控制。	2010 年 2 月 8 日
支援的新區域	Amazon S3 現在支援美國西部 (加利佛尼亞北部) 區域。此區域的要求新端點是 `s3-us-west-1.amazonaws.com` 。如需詳細資訊，請參閱儲存貯體與區域。	2009 年 12 月 2 日

變更	描述	日期
AWS SDK for .NET	AWS 現在提供程式庫、範例程式碼、教學課程和其他資源，讓他們偏好使用 .NET 語言特定 API 作業而非 REST 或 SOAP 來建置應用程式的軟體開發人員。這些程式庫提供基本功能 (REST 或 SOAP API 中不包含)，例如要求身分驗證、要求重試與錯誤處理，以便更容易開始。如需語言特定程式庫與資源的詳細資訊，請參閱「使用開發 AWS 套件使用 Amazon S3 進行開發」。	2009 年 11 月 11 日

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。