사용자 가이드

AWS Identity and Access Management

AWS Identity and Access Management: 사용자 가이드

Table of Contents

IAM이란 무엇입니까?

Follow us on Twitter

AWS Identity and Access Management(IAM)은 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있는 웹 서비스입니다. IAM을 사용하면 사용자가 액세스할 수 있는 AWS 리소스를 제어하는 권한을 관리할 수 있습니다. IAM을 사용하여 리소스를 사용하도록 인증(로그인) 및 권한 부여(권한 있음)된 대상을 제어합니다. IAM은 AWS 계정에 대한 인증 및 권한 부여를 제어하는 데 필요한 인프라를 제공합니다.

ID

AWS 계정(을)를 생성할 때는 해당 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 단일 로그인 ID로 시작합니다. 이 ID는 AWS 계정루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업을 참조하십시오.

IAM을 사용하여 루트 사용자 외에 관리자, 분석가, 개발자 등 다른 ID를 설정하고 작업을 성공적으로 수행하는 데 필요한 리소스에 대한 액세스 권한을 부여할 수 있습니다.

액세스 관리

IAM에 설정된 사용자는 로그인 자격 증명을 사용하여 AWS에 인증합니다. AWS 계정에서 신뢰하는 보안 주체(IAM 사용자, 페더레이션 사용자, IAM 역할 또는 애플리케이션)와 로그인 보안 인증 정보를 일치시키는 방식으로 인증이 제공됩니다. 다음으로 보안 주체에게 리소스 액세스 권한을 부여하도록 요청이 이루어집니다. 사용자에게 리소스에 대한 권한이 부여된 경우 권한 부여 요청에 대한 응답으로 액세스 권한이 부여됩니다. 예를 들어, 콘솔에 처음 로그인하고 콘솔 홈 페이지에 있는 경우에는 특정 서비스에 액세스하고 있지 않습니다. 서비스를 선택하면 인증 요청이 해당 서비스로 전송되어 인증된 사용자 목록에 ID가 있는지, 부여된 액세스 수준을 제어하기 위해 어떤 정책이 적용되고 있는지, 적용될 수 있는 기타 정책이 있는지 확인합니다. AWS 계정 내의 보안 주체나 신뢰할 수 있는 AWS 계정을 통하여 인증을 요청할 수 있습니다.

인증된 후에는 보안 주체가 AWS 계정에서 리소스에 조치를 취하거나 작업을 수행할 수 있습니다. 예를 들어, 보안 주체는 새 Amazon Elastic Compute Cloud 인스턴스를 시작하거나, IAM 그룹 멤버십을 수정하거나, Amazon Simple Storage Service 버킷을 삭제할 수 있습니다.

> **Tip**
>
> AWS Training and Certification에서 IAM에 대한 10분 소개 동영상을 제공합니다.

AWS Identity and Access Management 소개

서비스 가용성

IAM은 다른 많은 AWS 서비스처럼 최종 일관성이 있습니다. IAM은 전 세계 Amazon 데이터 센터 내의 여러 서버로 데이터를 복제함으로써 고가용성을 구현합니다. 일부 데이터를 변경하겠다는 요청이 성공하면 변경이 실행되고 그 결과는 안전하게 저장됩니다. 그러나 변경 사항은 IAM 전체에 복제되어야 하고, 이 작업에는 어느 정도 시간이 소요됩니다. 그러한 변경 사항에는 사용자, 그룹, 역할 또는 정책을 만들거나 업데이트한 것이 포함됩니다. 그러한 IAM 변경 사항을 애플리케이션의 중요한 고가용성 코드 경로에 포함시키지 않는 것이 좋습니다. 대신 자주 실행하지 않는 별도의 초기화 루틴이나 설정 루틴에서 IAM을 변경하세요. 또한 프로덕션 워크플로우에서 변경 사항을 적용하기 전에 변경 사항이 전파되었는지 확인하세요. 자세한 내용은 변경 사항이 매번 즉시 표시되는 것은 아닙니다 단원을 참조하십시오.

서비스 비용 정보

AWS Identity and Access Management(IAM), AWS IAM Identity Center, AWS Security Token Service(AWS STS)는 추가 비용 없이 AWS 계정에 제공되는 기능입니다. IAM 사용자 또는 AWS STS 임시 보안 자격 증명을 사용하여 다른 AWS 서비스에 액세스하는 경우에만 요금이 부과됩니다.

IAM Access Analyzer 외부 액세스 분석은 추가 비용 없이 제공됩니다. 하지만 사용하지 않은 액세스 분석 및 고객 정책 확인에는 요금이 발생합니다. IAM Access Analyzer에 관련된 전체적인 요금 및 가격 목록은 IAM Access Analyzer 요금을 참조하세요.

다른 AWS 제품 요금에 대한 자세한 내용은 Amazon Web Services 요금 페이지를 참조하세요.

다른 AWS 서비스와의 통합

IAM은 많은 AWS 서비스와 통합되어 있습니다. IAM과 함께 사용할 수 있는 AWS 서비스 목록과 해당 서비스가 지원하는 IAM 기능은 AWS IAM으로 작업하는 서비스 섹션을 참조하세요.

IAM 개념에 대한 자세한 내용은 다음 주제를 참조하세요.

주제

- 권한 및 정책이 액세스 관리를 제공하는 방법
- ABAC 권한 부여를 통한 속성 기반 권한 정의

왜 IAM을 사용해야 하나요?

AWS Identity and Access Management는 AWS 리소스에 대한 액세스를 안전하게 관리하기 위한 강력한 도구입니다. IAM을 사용하여 얻을 수 있는 주요 이점 중 하나는 AWS 계정에 공유 액세스 권한을 부여할 수 있다는 것입니다. 또한 IAM을 사용하면 세분화된 권한을 할당할 수 있으므로 각 사용자가 특정 리소스에서 수행할 수 있는 작업을 정확히 제어할 수 있습니다. 이러한 수준의 액세스 제어는 AWS 환경의 보안을 유지하는 데 매우 중요합니다. IAM은 몇 가지 다른 보안 기능도 제공합니다. 추가 보호 계층을 위해 다중 인증(MFA)을 추가하고 ID 페더레이션을 활용하여 기업 네트워크 또는 다른 ID 공급자의 사용자를 원활하게 통합할 수 있습니다. 또한 IAM은 AWS CloudTrail과 통합되어 감사 및 규정 준수 요구 사항을 지원하는 상세한 로깅 및 ID 정보를 제공합니다. 이러한 기능을 활용하면 중요한 AWS 리소스에 대한 액세스를 엄격하게 제어하고 안전하게 보호할 수 있습니다.

AWS 계정에 대한 공유 액세스

암호나 액세스 키를 공유하지 않고도 AWS 계정의 리소스를 관리하고 사용할 수 있는 권한을 다른 사람에게 부여할 수 있습니다.

세분화된 권한

리소스에 따라 여러 사람에게 다양한 권한을 부여할 수 있습니다. 예를 들어 일부 사용자에게 Amazon Elastic Compute Cloud(Amazon EC2), Amazon Simple Storage Service(Amazon S3), Amazon DynamoDB, Amazon Redshift 및 기타 AWS 서비스에 대한 완전한 액세스를 허용할 수 있습니다. 다른 사용자에게는 일부 Amazon S3 버킷에 대한 읽기 전용 권한, 일부 Amazon EC2 인스턴스를 관리할 수 있는 권한 또는 결제 정보에만 액세스할 수 있는 권한을 허용할 수 있습니다.

Amazon EC2에서 실행되는 애플리케이션을 위한 보안 AWS 리소스 액세스

EC2 인스턴스에서 실행되는 애플리케이션의 경우 IAM 기능을 사용하여 자격 증명을 안전하게 제공할 수 있습니다. 이러한 자격 증명은 애플리케이션에 다른 AWS 리소스에 액세스할 수 있는 권한을 제공합니다. 예를 들면 이러한 리소스에는 S3 버킷 및 DynamoDB 테이블이 있습니다.

멀티 팩터 인증(MFA)

보안 강화를 위해 계정과 개별 사용자에게 2팩터 인증을 추가할 수 있습니다. MFA를 사용할 경우 계정 소유자나 사용자가 계정 작업을 위해 암호나 액세스 키뿐 아니라 특별히 구성된 디바이스의 코드도

제공해야 합니다. 이미 다른 서비스와 함께 FIDO 보안 키를 사용하고 있으며 FIDO 보안 키의 구성에 AWS가 지원되는 경우 MFA 보안을 위해 WebAutn을 사용할 수 있습니다. 자세한 내용은 패스키 및 보안 키 사용이 지원되는 구성 단원을 참조하십시오.

아이덴티티 페더레이션

기업 네트워크나 인터넷 ID 공급자와 같은 다른 곳에 이미 암호가 있는 사용자에게 AWS 계정에 대한 액세스를 허용할 수 있습니다. 이러한 사용자에게는 IAM 모범 사례 권장 사항을 준수하는 임시 자격 증명이 부여됩니다. ID 페더레이션을 사용하면 AWS 계정 보안이 강화됩니다.

보장을 위한 자격 증명 정보

AWS CloudTrail을 사용하는 경우 계정의 리소스를 요청한 사람에 대한 정보가 포함된 로그 레코드를 받게 됩니다. 이 정보는 IAM 자격 증명을 기반으로 합니다.

PCI DSS 준수

IAM에서는 판매자 또는 서비스 공급자에 의한 신용카드 데이터의 처리, 저장 및 전송을 지원하며, Payment Card Industry(PCI) Data Security Standard(DSS) 준수를 검증받았습니다. AWS PCI 규정 준수 패키지의 사본을 요청하는 방법 등 PCI DSS에 대해 자세히 알아보려면 PCI DSS 레벨 1을 참조하세요.

IAM을 사용하는 경우

AWS Identity and Access Management은 AWS 내의 ID에 따라 액세스 제어의 기반을 제공하는 핵심 인프라 서비스입니다. AWS 계정에 액세스할 때마다 IAM을 사용합니다. IAM을 사용하는 방법은 조직 내의 특정 책임과 직무에 따라 달라집니다. AWS 서비스 사용자는 IAM을 사용하여 일상 업무에 필요한 AWS 리소스에 액세스하고 관리자는 적절한 권한을 부여합니다. 반면 IAM 관리자는 IAM ID를 관리하고 리소스에 대한 액세스를 제어하는 정책을 작성할 책임이 있습니다. 역할과 관계없이 AWS 리소스에 대한 액세스를 인증하고 권한을 부여할 때마다 IAM과 상호 작용하게 됩니다. 여기에는 IAM 사용자로 로그인하거나, IAM 역할을 수임하거나, 원활한 액세스를 위해 ID 페더레이션을 활용하는 것이 포함될 수 있습니다. AWS 환경에 대한 보안 액세스를 효과적으로 관리하려면 다양한 IAM 기능과 사용 사례를 이해하는 것이 중요합니다. 정책 및 권한 생성과 관련하여 IAM은 유연하고 세분화된 접근 방식을 제공합니다. 사용자 또는 역할이 액세스할 수 있는 작업 및 리소스를 지정하는 자격 증명 기반 정책 외에 역할을 수임할 수 있는 보안 주체를 제어하는 신뢰 정책을 정의할 수 있습니다. 이러한 IAM 정책을 구성하면 사용자와 애플리케이션이 필요한 작업을 수행할 수 있는 적절한 수준의 권한을 갖도록 할 수 있습니다.

다른 직무를 수행하는 경우

AWS Identity and Access Management은 AWS 내의 ID에 따라 액세스 제어의 기반을 제공하는 핵심 인프라 서비스입니다. AWS 계정에 액세스할 때마다 IAM을 사용합니다.

IAM를 사용하는 방법은 AWS에서 수행하는 작업에 따라 달라집니다.

- 서비스 사용자 - AWS 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 고급 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다.
- 서비스 관리자 - 회사에서 AWS 리소스를 책임지고 있는 경우 IAM에 대한 전체 액세스 권한을 가지고 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 IAM 기능과 리소스를 결정합니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오.
- IAM 관리자 - IAM 관리자인 경우 IAM 자격 증명을 관리하고 IAM에 대한 액세스 권한을 관리하는 정책을 작성할 수 있습니다.

AWS 리소스에 액세스할 권한이 있는 경우

인증은 ID 보안 인증을 사용하여 AWS에 로그인하는 방식입니다. AWS 계정 루트 사용자(이)나, IAM 사용자로 또는 IAM 역할을 수임하여 인증(AWS에 로그인)되어야 합니다.

ID 소스를 통해 제공된 보안 인증 정보를 사용하여 페더레이션 ID로 AWS에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증, Google 또는 Facebook 보안 인증이 페더레이션 ID의 예입니다. 페더레이션 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 연동을 사용하여 AWS에 액세스하면 간접적으로 역할을 수임합니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. AWS에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법을 참조하십시오.

AWS에 프로그래밍 방식으로 액세스하는 경우, AWS에서는 보안 인증 정보를 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK) 및 명령줄 인터페이스(CLI)를 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청에 서명을 참조하십시오.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS는 다중 인증(MFA)을 사용하여 계정의 보안을 강화하는 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 다중 인증 및 IAM 사용 설명서의 AWS에서 다중 인증(MFA) 사용을 참조하십시오.

IAM 사용자로 로그인하는 경우

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 AWS 계정내 ID입니다. 가능하면 암호 및 액세스 키와 같은 장기 보안 인증이 있는 IAM 사용자를 생성하는 대신 임시 보안 인증을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 보안 인증이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체를 참조하십시오.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증 정보를 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자를 만들어야 하는 경우(역할이 아님)를 참조하십시오.

IAM 역할을 맡을 경우

IAM 역할은 특정 권한을 가지고 있는 AWS 계정 계정 내 ID입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할 전환하여 AWS Management Console에서 IAM 역할을 임시로 수임할 수 있습니다. AWS CLI 또는 AWS API 태스크를 직접적으로 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 사용을 참조하십시오.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 서드 파티 ID 공급자의 역할 생성 단원을 참조하십시오. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하십시오.

- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 태스크에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스를 사용하면 (역할을 프록시로 사용하는 대신) 리소스에 정책을 직접 연결할 수 있습니다. 크로스 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.
- 교차 서비스 액세스 - 일부 AWS 서비스는 다른 AWS 서비스의 기능을 사용합니다. 예를 들어 서비스에서 직접적 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 태스크를 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하십시오.
  - 서비스 연결 역할 – 서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수는 없습니다.
- Amazon EC2에서 실행 중인 애플리케이션 – IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 보안 인증을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS역할을 할당하고 해당 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하십시오.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하십시오.

정책 및 권한을 생성하는 경우

사용자에게 권한을 부여하려면 사용자가 수행할 수 있는 작업과 그러한 작업으로 인해 영향을 받을 수 있는 리소스를 나열하는 문서에 해당하는 정책을 만들어야 합니다. 명시적으로 허용되지 않은 작업 또는 리소스는 기본적으로 모두 거부됩니다. 정책을 생성하여 보안 주체(사용자, 사용자 그룹, 사용자가 맡는 역할, 리소스)에 연결할 수 있습니다.

다음 정책은 IAM 역할과 함께 사용할 수 있습니다.

- 신뢰 정책 – 역할을 수임할 수 있는 보안 주체와 역할 수임 조건을 정의합니다. 신뢰 정책은 IAM 역할에 대해 특정한 유형의 리소스 기반 정책 유형입니다. 역할은 하나의 신뢰 정책만 가질 수 있습니다.
- 자격 증명 기반 정책(인라인 및 관리형) – 이러한 정책은 해당 역할의 사용자가 수행할 수 있는 (또는 수행할 수 없도록 거부되는) 권한과 리소스 위치를 정의합니다.

IAM 자격 증명 기반 정책의 예 사용 시 IAM 자격 증명에 대한 권한을 정의하는 데 유용할 수 있습니다. 필요로 하는 정책을 찾은 다음에 View this policy(이 정책 보기)를 선택하여 정책의 JSON을 확인합니다. JSON 정책 문서를 자체 정책의 템플릿으로 활용할 수 있습니다.

> **Note**
>
> IAM Identity Center를 사용하여 사용자를 관리하는 경우 권한 정책을 보안 주체에 연결하는 대신 IAM Identity Center에서 권한 세트를 할당합니다. 그룹 또는 AWS IAM Identity Center의 사용자에 권한 세트를 할당하면 IAM Identity Center가 각 계정에 해당되는 IAM 역할을 생성하고 권한 세트에 지정된 정책을 해당 역할에 연결합니다. IAM Identity Center는 역할을 관리하고, 정의에 따라 인증된 사용자가 역할을 맡을 수 있도록 합니다. 권한 세트를 수정하면 IAM Identity Center에서 해당 IAM 정책 및 역할이 그에 따라 업데이트되도록 합니다.
> IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?(IAM Identity Center란 무엇인가요?) 섹션을 참조하세요.

IAM 관리 방법

AWS 환경 내에서 AWS Identity and Access Management를 관리하려면 다양한 도구와 인터페이스를 활용해야 합니다. 가장 일반적인 방법은 사용자 및 역할 생성부터 권한 구성에 이르기까지 광범위한

IAM 관리 작업을 수행할 수 있는 웹 기반 인터페이스인 AWS Management Console을 사용하는 것입니다.

명령줄 인터페이스에 더 익숙한 사용자를 위해 AWS는 두 가지 명령줄 도구 세트(AWS Command Line Interface 및 AWS Tools for Windows PowerShell)를 제공합니다. 이를 통해 터미널에서 직접 IAM 관련 명령을 실행할 수 있으며, 이는 콘솔을 탐색하는 것보다 훨씬 효율적입니다. 또한 AWS CloudShell은 콘솔 로그인과 관련된 권한을 사용하여 웹 브라우저에서 직접 CLI 또는 SDK 명령을 실행할 수 있습니다.

콘솔과 명령줄 외에도 AWS는 다양한 프로그래밍 언어를 위한 소프트웨어 개발 키트(SDK)를 제공하므로 IAM 관리 기능을 애플리케이션에 직접 통합할 수 있습니다. 또는 서비스로 직접 HTTPS 요청을 실행할 수 있는 IAM 쿼리 API를 사용하여 프로그래밍 방식으로 IAM에 액세스할 수 있습니다. 이러한 다양한 관리 접근 방식을 활용하면 IAM을 기존 워크플로 및 프로세스에 유연하게 통합할 수 있습니다.

AWS Management Console 사용

콘솔은 IAM 및 AWS 리소스를 관리하기 위한 브라우저 기반 인터페이스입니다. 콘솔을 통하여 IAM에 액세스하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 로그인 방법을 참조하세요.

AWS 콘솔

AWS Management Console은 AWS 리소스 관리를 위한 다양한 서비스 콘솔의 모음을 구성하고 참조하는 웹 애플리케이션입니다. 처음 로그인하면 콘솔 홈 페이지가 나타납니다. 홈 페이지는 각 서비스 콘솔에 대한 액세스를 제공하며, 이 단일 위치에서 AWS 관련 작업을 수행하는 데 필요한 정보에 액세스할 수 있습니다. 콘솔에 로그인한 후 사용할 수 있는 서비스와 애플리케이션은 액세스 권한이 있는 AWS 리소스에 따라 달라집니다. 역할을 수임하거나 권한이 부여된 그룹의 멤버가 되거나 명시적으로 권한을 받아 리소스에 대한 권한을 부여받을 수 있습니다. 독립 실행형 AWS 계정의 경우 루트 사용자 또는 IAM 관리자가 리소스에 대한 액세스를 구성합니다. AWS Organizations의 경우 관리 계정 또는 위임된 관리자가 리소스에 대한 액세스를 구성합니다.

AWS Management Console을 사용하여 AWS 리소스를 관리하게 하려면 보안 모범 사례로 임시 보안 인증 정보를 사용하여 사용자를 구성하는 것이 좋습니다. 역할을 수임한 IAM 사용자, 페더레이션 사용자 및 IAM Identity Center의 사용자는 임시 보안 인증을 갖고 있으며, IAM 사용자와 루트 사용자는 장기 보안 인증을 갖고 있습니다. 루트 사용자 보안 인증은 AWS 계정에 대한 전체 액세스 권한을 제공하는 반면, 다른 사용자는 IAM 정책에서 부여한 리소스에 대한 액세스 권한을 제공하는 보안 인증을 보유합니다.

로그인 환경은 AWS Management Console 사용자 유형마다 다릅니다.

- IAM 사용자와 루트 사용자는 기본 AWS 로그인 URL(https://signin.aws.amazon.com)에서 로그인합니다. 로그인한 후에 권한이 부여된 계정의 리소스에 액세스할 수 있습니다.

  루트 사용자로 로그인하려면 루트 사용자 이메일 주소와 암호가 있어야 합니다.

  IAM 사용자로 로그인하려면 AWS 계정 번호나 별칭, IAM 사용자 이름 및 IAM 사용자 암호가 있어야 합니다.

  계정의 IAM 사용자를 장기 보안 인증이 필요한 특정 상황(예: 긴급 액세스)으로 제한하고 루트 사용자 보안 인증이 필요한 작업에만 루트 사용자를 사용하는 것이 좋습니다.

  사용자 편의를 위해 AWS 로그인 페이지는 브라우저 쿠키를 사용하여 IAM 사용자 이름 및 계정 정보를 기억합니다. 다음에 사용자가 AWS Management Console의 아무 페이지로든 이동하면 콘솔이 쿠키를 사용하여 사용자를 사용자 로그인 페이지로 리디렉션합니다.

  세션이 끝나면 콘솔에서 로그아웃하여 이전 로그인을 다시 사용하지 않도록 합니다.
- IAM Identity Center 사용자는 조직에 고유한 특정 AWS 액세스 포털을 사용하여 로그인합니다. 로그인한 후에 액세스할 계정이나 애플리케이션을 선택할 수 있습니다. 계정에 액세스하려는 경우 관리 세션에 사용할 권한 세트를 선택합니다.
- 페더레이션 사용자는 사용자 지정 엔터프라이즈 액세스 포털을 사용하여 AWS 계정 로그인에 연결된 외부 ID 제공업체를 통해 관리합니다. 페더레이션 사용자가 사용할 수 있는 AWS 리소스는 해당 조직에서 선택한 정책에 따라 달라집니다.

> **Note**
>
> 추가적으로 보안을 강화하기 위해 루트 사용자, IAM 사용자 및 IAM Identity Center 사용자는 AWS 리소스에 대한 액세스 권한을 부여하기 전에 AWS에서 확인하는 다중 인증(MFA)을 이용할 수 있습니다. MFA가 활성화되는 경우 로그인하려면 MFA 디바이스에 대한 액세스 권한도 있어야 합니다.

다양한 사용자가 관리 콘솔에 로그인하는 방법에 대해 자세히 알아보려면 AWS 로그인 사용 설명서의 AWS Management Console에 로그인을 참조하세요.

AWS 명령줄 도구

AWS 명령줄 도구를 통해 시스템 명령줄에서 명령을 실행하여 IAM 및 AWS 작업을 수행할 수 있습니다. 명령줄을 사용하는 것이 콘솔을 사용하는 것보다 더 빠르고 편리할 수 있습니다. AWS 작업을 수행하는 스크립트를 작성할 때도 명령줄 도구가 유용합니다.

AWS에서는 AWS Command Line Interface(AWS CLI) 및 AWS Tools for Windows PowerShell라는 두 가지 명령줄 도구 세트를 제공합니다. AWS CLI 설치 및 사용에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서를 참조하세요. Tools for Windows PowerShell 도구 설치 및 사용에 대한 자세한 내용은 AWS Tools for Windows PowerShell 사용 설명서를 참조하세요.

콘솔에 로그인한 후 브라우저에서 AWS CloudShell을 사용하여 CLI 또는 SDK 명령을 실행할 수 있습니다. AWS 리소스 액세스 권한은 콘솔에 로그인할 때 사용하는 보안 인증에 기반합니다. 경험에 따라 AWS 계정을 관리하는 데 CLI가 더 효율적일 수 있습니다. 자세한 내용은 AWS CloudShell을 사용하여 AWS Identity and Access Management에서 작업 단원을 참조하세요.

AWS Command Line Interface(CLI) 및 소프트웨어 개발 키트(SDK)

IAM Identity Center 및 IAM 사용자는 CLI 또는 관련 SDK의 애플리케이션 인터페이스(API)를 통해 인증할 때 다양한 방법을 사용하여 보안 인증을 인증합니다.

보안 인증 및 구성 설정은 시스템 또는 사용자 환경 변수, 로컬 AWS 구성 파일 또는 명령줄에서 파라미터로 명시적으로 선언된 위치 등 다양한 장소에 있습니다. 특정 위치가 다른 위치보다 우선합니다.

IAM Identity Center 및 IAM 모두 CLI 또는 SDK에서 사용할 수 있는 액세스 키를 제공합니다. IAM Identity Center 액세스 키는 자동으로 새로 고칠 수 있는 임시 보안 인증이며 IAM 사용자에게 연결된 장기 액세스 키보다 권장됩니다.

CLI 또는 SDK를 사용하여 AWS 계정을 관리하려면 브라우저에서 AWS CloudShell을 사용할 수 있습니다. CloudShell을 사용하여 CLI 또는 SDK 명령을 실행하는 경우 먼저 콘솔에 로그인해야 합니다. AWS 리소스 액세스 권한은 콘솔에 로그인할 때 사용하는 보안 인증에 기반합니다. 경험에 따라 AWS 계정을 관리하는 데 CLI가 더 효율적일 수 있습니다.

애플리케이션 개발의 경우 CLI 또는 SDK를 컴퓨터에 다운로드하고 명령 프롬프트 또는 도커 창에서 로그인할 수 있습니다. 이 시나리오에서는 CLI 스크립트 또는 SDK 애플리케이션의 일부로 인증 및 액세스 보안 인증을 구성합니다. 환경 및 사용 가능한 액세스 권한에 따라 다양한 방식으로 리소스에 대한 프로그래밍 방식의 액세스를 구성할 수 있습니다.

- AWS 서비스를 통해 로컬 코드를 인증하는 경우 IAM Identity Center 및 IAM Roles Anywhere를 사용하는 것이 좋습니다.

- AWS 환경 내에서 실행되는 코드를 인증하는 경우 IAM 역할을 사용하거나 IAM Identity Center 보안 인증을 사용하는 것이 좋습니다.

AWS 액세스 포털을 사용하여 로그인할 때 권한 세트를 선택하는 시작 페이지에서 단기 보안 인증 정보를 얻을 수 있습니다. 이러한 보안 인증은 기간이 정의되어 있으며 자동으로 새로 고쳐지지 않습니다. 이러한 보안 인증을 사용하려면 AWS 포털에 로그인한 후 AWS 계정을 선택하고 권한 세트를 선택합니다. 명령줄 또는 프로그래밍 방식의 액세스를 선택하여 프로그래밍 방식으로 또는 CLI에서 AWS 리소스에 액세스하는 데 사용할 수 있는 옵션을 확인합니다. 이러한 방법에 대한 자세한 내용은 IAM Identity Center 사용 설명서의 Getting and refreshing temporary credentials를 참조하세요. 이러한 보안 인증은 애플리케이션 개발 중에 코드를 신속하게 테스트하기 위해 자주 사용됩니다.

AWS 리소스에 대한 액세스를 자동화하는 경우 자동으로 새로 고쳐지는 IAM Identity Center 보안 인증을 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 권한 세트를 구성한 경우 `aws configure sso` 명령을 사용하여 사용 가능한 보안 인증을 식별하고 프로파일에 저장하는 데 도움이 되는 명령줄 마법사를 사용합니다. 프로파일 구성에 대한 자세한 내용은 AWS Command Line Interface 버전 2 사용 설명서의 `aws configure sso` 마법사를 사용하여 프로파일 구성을 참조하세요.

> **Note**
>
> 많은 샘플 애플리케이션에서 IAM 사용자 또는 루트 사용자에 연결된 장기 액세스 키를 사용합니다. 장기 보안 인증은 샌드박스 환경에서 학습 체험의 일환으로만 사용해야 합니다. 장기 액세스 키의 대안을 검토하고 가능한 한 빨리 IAM Identity Center 보안 인증 또는 IAM 역할과 같은 대체 보안 인증을 사용하도록 코드를 전환할 계획을 수립합니다. 코드를 전환한 후에는 액세스 키를 삭제합니다.

CLI 구성에 대한 자세한 내용은 AWS Command Line Interface 버전 2 사용 설명서의 최신 버전의 AWS CLI 설치 또는 업데이트 및 AWS Command Line Interface 사용 설명서의 보안 인증 정보 인증 및 액세스를 참조하세요.

SDK 구성에 대한 자세한 내용은 AWS SDK 및 도구 참조 가이드의 IAM Identity Center authentication 및 AWS SDK 및 도구 참조 가이드의 IAM Roles Anywhere를 참조하세요.

AWS SDK 사용

AWS에서는 다양한 프로그래밍 언어 및 플랫폼(Java, Python, Ruby, .NET, iOS, Android 등)을 위한 라이브러리와 샘플 코드로 구성된 소프트웨어 개발 키트(SDK)를 제공합니다. SDK를 사용하면 편리하게

IAM 및 AWS에 프로그래밍 방식으로 액세스할 수 있습니다. 예를 들어 SDK는 요청에 암호화 방식으로 서명, 오류 관리 및 자동으로 요청 재시도와 같은 작업을 처리합니다. 다운로드 및 설치 방법을 비롯하여 AWS SDK에 대한 자세한 내용은 Amazon Web Services용 도구 페이지를 참조하세요.

IAM 쿼리 API 사용

서비스로 직접 HTTPS 요청을 실행할 수 있는 IAM 쿼리 API를 사용하여 프로그래밍 방식으로 IAM 및 AWS에 액세스할 수 있습니다. 쿼리 API를 사용할 때는 자격 증명을 사용하여 요청에 디지털 방식으로 서명하는 코드를 포함해야 합니다. 자세한 내용은 HTTP 쿼리 요청을 사용하여 IAM API 호출 및 IAM API 참조를 참조하세요.

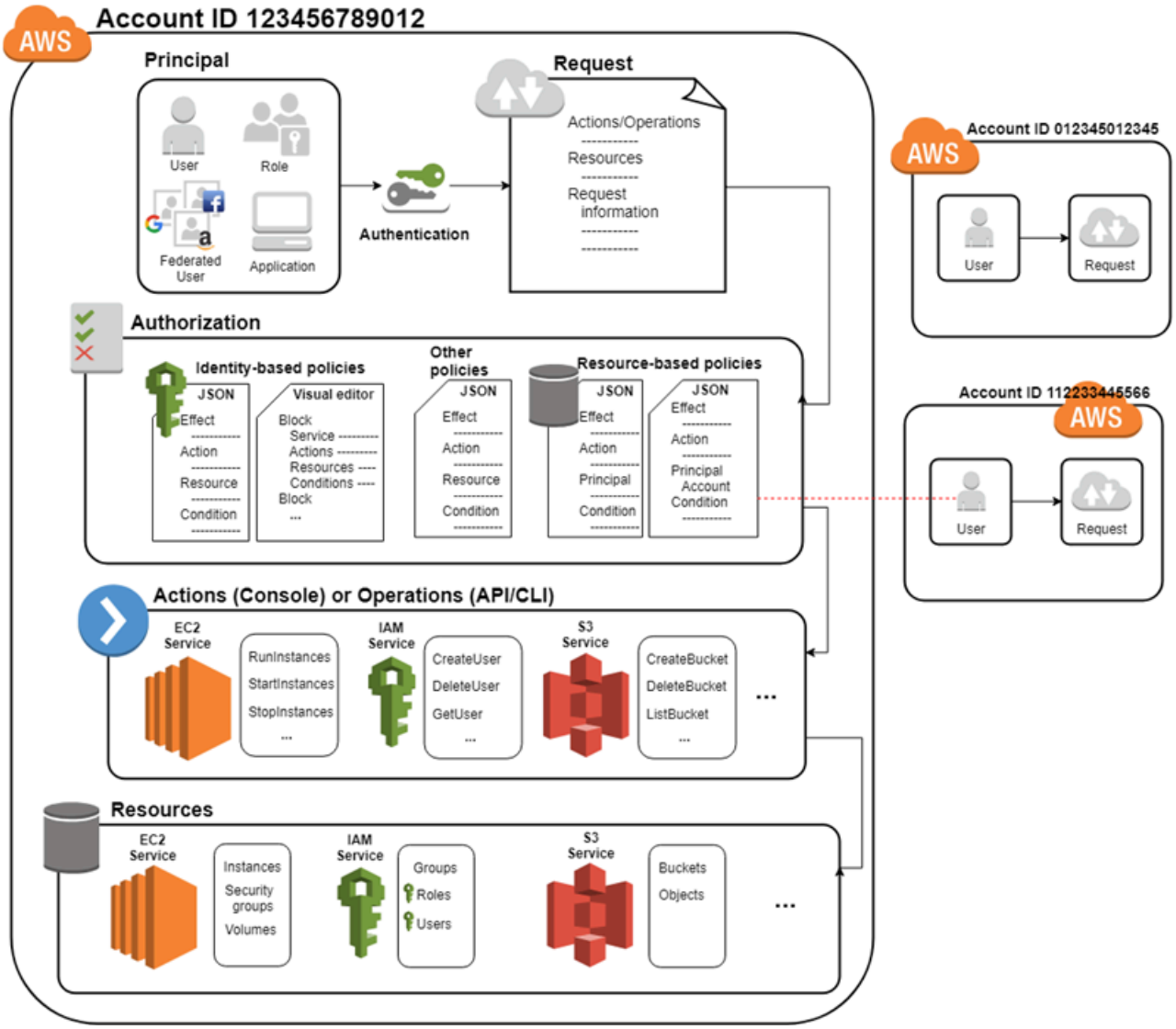
IAM 작동 방식

AWS Identity and Access Management는 AWS 계정에 대한 인증 및 권한 부여를 제어하는 데 필요한 인프라를 제공합니다.

먼저, 실제 사용자나 애플리케이션이 AWS를 통해 로그인 보안 인증을 사용하여 인증합니다. IAM은 로그인 보안 인증 정보를 AWS 계정이 신뢰하는 보안 주체(IAM 사용자, 페더레이션 사용자, IAM 역할 또는 애플리케이션)에 일치시키고 AWS에 액세스할 수 있는 권한을 인증합니다.

다음으로 IAM은 리소스 액세스 권한을 보안 주체에게 부여하도록 요청합니다. IAM은 권한 부여 요청에 대한 응답으로 액세스 권한을 부여하거나 거부합니다. 예를 들어, 콘솔에 처음 로그인하고 콘솔 홈 페이지에 있는 경우에는 특정 서비스에 액세스하고 있지 않습니다. 서비스를 선택하면 해당 서비스에 대한 권한 부여 요청이 IAM에 전송됩니다. IAM은 권한이 부여된 사용자 목록에 ID가 있는지 확인하고, 부여된 액세스 권한 수준을 제어하는 정책을 결정하고, 적용될 수 있는 다른 정책이 있는지 평가합니다. AWS 계정 내의 보안 주체나 신뢰할 수 있는 다른 AWS 계정의 보안 주체는 권한 부여를 요청할 수 있습니다.

권한이 부여된 후 보안 주체가 AWS 계정에서 리소스에 작업 또는 연산을 수행할 수 있습니다. 예를 들어, 보안 주체는 새 Amazon Elastic Compute Cloud 인스턴스를 시작하거나, IAM 그룹 멤버십을 수정하거나, Amazon Simple Storage Service 버킷을 삭제할 수 있습니다. 다음 다이어그램은 IAM 인프라를 통한 이 프로세스를 보여줍니다.



요청 구성 요소

보안 주체가 AWS Management Console, AWS API 또는 AWS CLI를 사용하려고 시도하면 해당 보안 주체가 요청을 AWS에 전송합니다. 이 요청에는 다음 정보가 포함되어 있습니다.

- 작업 또는 연산 - 보안 주체가 수행하려는 작업 또는 연산입니다(예: AWS Management Console에서의 작업이나 AWS CLI 또는 AWS API에서의 연산).
- 리소스 - 보안 주체가 작업 또는 연산 수행을 요청하는 AWS 리소스 객체입니다.
- 보안 주체 – 엔터티(사용자 또는 역할)를 사용하여 요청을 보내는 사람 또는 애플리케이션입니다. 보안 주체에 대한 정보에는 권한 정책이 포함됩니다.
- 환경 데이터 – IP 주소, 사용자 에이전트, SSL 사용 상태, 타임스탬프에 대한 정보입니다.

- 리소스 데이터 - DynamoDB 테이블 이름 또는 Amazon EC2 인스턴스의 태그 등 요청된 리소스와 관련된 데이터입니다.

AWS가 요청 정보를 요청 콘텍스트에 수집하면 IAM이 평가하여 요청 권한을 부여합니다.

보안 주체가 인증되는 방법

보안 주체는 보안 인증 정보를 사용하여 AWS에 로그인하고 IAM은 이 보안 인증 정보를 인증하여 보안 주체가 AWS에 요청을 보내도록 허용합니다. Amazon S3 및 AWS STS 등의 일부 서비스는 익명 사용자의 특정 요청을 허용합니다. 하지만 이는 규칙의 예외입니다. 각 유형의 사용자는 인증을 거칩니다.

- 루트 사용자 - 인증에 사용되는 로그인 보안 인증 정보는 AWS 계정을 만들 때 사용한 이메일 주소와 당시 지정한 암호입니다.
- 페더레이션 사용자 - 자격 증명 공급자가 사용자를 인증하고 자격 증명을 AWS에 전달하므로 AWS에 직접 로그인하지 않아도 됩니다. IAM Identity Center와 IAM 모두 페더레이션 사용자를 지원합니다.
- AWS IAM Identity Center 디렉터리의 사용자(페더레이션 아님) - IAM Identity Center 기본 디렉터리에서 직접 생성된 사용자는 AWS 액세스 포털을 사용하여 로그인하고 사용자 이름과 암호를 입력합니다.
- IAM 사용자 - 계정 ID 또는 별칭, 사용자 이름, 암호를 입력하여 로그인합니다. API 또는 AWS CLI에서 워크로드를 인증하려면 역할 수임을 통해 임시 보안 인증 정보를 사용하거나 액세스 키와 비밀 키를 제공하여 장기 보안 인증 정보를 사용합니다.

  IAM 엔터티에 대한 자세한 내용은 IAM 사용자 및 IAM 역할 섹션을 참조하세요.

AWS는 모든 사용자에 다중 인증(MFA)을 사용하여 계정의 보안을 강화하는 것을 권장합니다. MFA에 대한 자세한 내용은 IAM의 AWS 다중 인증 섹션을 참조하세요.

권한 부여 및 권한 정책 기본 사항

권한 부여란 보안 주체가 요청을 완료하는 데 필요한 권한을 갖는 것을 말합니다. IAM은 권한 부여 과정에서 요청 컨텍스트의 값을 사용하여 요청에 적용되는 정책을 식별합니다. 그런 다음 이것은 정책을 사용하여 요청을 허용하거나 거부할지 여부를 결정합니다. IAM은 대부분의 권한 정책을 보안 주체 엔터티의 권한을 지정하는 JSON 문서로 저장합니다.

권한 부여 요청에 영향을 미칠 수 있는 몇 가지 정책 유형이 있습니다. 계정의 AWS 리소스에 액세스할 수 있는 권한을 사용자에게 제공하려면 자격 증명 기반 정책을 사용할 수 있습니다. 리소스 기반 정책

은 크로스 계정 액세스 권한을 부여할 수 있습니다. 다른 계정에서 요청하려면 다른 계정의 정책에서 해당 리소스에 대한 액세스를 허용해야 하며, 또한 요청하는 데 사용하는 IAM 엔터티에 해당 요청을 허용하는 자격 증명 기반 정책이 있어야 합니다.

IAM은 요청 컨텍스트에 적용되는 각 정책을 확인합니다. IAM 정책 평가는 명시적 거부를 사용합니다. 즉, 단일 권한 정책에 거부된 작업이 포함된 경우 IAM은 전체 요청을 거부하고 평가를 중지합니다. 요청이 기본적으로 거부되므로 IAM이 요청 권한을 부여하려면 관련 권한 정책에서 요청의 모든 부분을 허용해야 합니다. 단일 계정 내 요청 평가 로직은 다음과 같은 기본 규칙을 따릅니다.

- 기본적으로 모든 요청을 거부합니다. (일반적으로, AWS 계정 루트 사용자 증명을 사용하여 해당 계정의 리소스를 요청하는 경우는 항상 허용됩니다.)
- 권한 정책(자격 증명 기반 또는 리소스 기반)에 포함된 명시적 허용은 이 기본 작동을 재정의합니다.
- 조직 SCP, IAM 권한 경계 또는 세션 정책이 있는 경우 허용이 재정의됩니다. 하나 이상의 이러한 정책 유형이 존재하는 경우 이들 정책 유형 모두가 해당 요청을 허용해야 합니다. 그렇지 않은 경우 묵시적으로 거부됩니다.
- 정책의 명시적 거부는 정책의 모든 허용을 무시합니다.

자세한 내용은 정책 평가 로직을 참조하십시오.

IAM이 보안 주체를 인증하고 권한을 부여한 후, IAM은 보안 주체에 적용되는 권한 정책을 평가하여 요청의 작업이나 연산을 승인합니다. 각 AWS 서비스는 지원하는 작업(연산)을 정의하며, 리소스 보기, 생성, 편집, 삭제 등 리소스에 대해 수행할 수 있는 작업을 포함합니다. 보안 주체에 적용되는 권한 정책에는 연산을 수행하는 데 필요한 작업이 포함되어야 합니다. IAM이 권한 정책을 평가하는 방법에 대한 자세한 내용은 the section called “정책 평가 로직” 섹션을 참조하세요.

서비스는 각 리소스에서 보안 주체가 수행할 수 있는 일련의 작업을 정의합니다. 권한 정책을 생성할 때는 사용자가 수행할 수 있게 하려는 작업을 포함해야 합니다. 예를 들어 IAM은 다음 기본 작업을 비롯하여 사용자 리소스에 대한 40개 이상의 작업을 지원합니다.

- `CreateUser`
- `DeleteUser`
- `GetUser`
- `UpdateUser`

또한 요청이 조건을 충족할 때 리소스 액세스 권한을 제공하는 조건을 권한 정책에 지정할 수 있습니다. 예를 들어 특정 날짜 이후에만 정책문이 적용되도록 하거나 API에 특정 값이 표시되는 경우에 액세스를 제어하려 할 수 있습니다. 조건을 지정하려면 정책문의 ???`Condition` 요소를 사용합니다.

IAM이 요청의 연산을 승인하면 보안 주체가 계정 내 관련 리소스에서 해당 작업을 수행할 수 있습니다. 리소스는 서비스 내에 존재하는 객체입니다. 예를 들어 Amazon EC2 인스턴스, IAM 사용자 및 Amazon S3 버킷이 있습니다. 보안 주체가 권한 정책에 포함되지 않은 리소스에 대한 작업을 수행하기 위한 요청을 생성하면 서비스는 요청을 거부합니다. 예를 들어 IAM 역할을 삭제할 권한이 있지만 IAM 그룹 삭제를 요청하는 경우 IAM 그룹을 삭제할 권한이 없으면 요청이 실패합니다. 다양한 AWS 서비스가 지원하는 작업, 리소스, 조건 키에 대한 자세한 내용은 Actions, Resources, and Condition Keys for AWS Services를 참조하세요.

IAM ID 및 자격 증명 비교

AWS Identity and Access Management에서 관리되는 ID는 IAM 사용자, IAM 역할 및 IAM 그룹입니다. 이러한 ID는 AWS가 AWS 계정과 함께 생성한 루트 사용자 외에 사용됩니다.

일상적인 작업, 심지어 관리 작업의 경우에도 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 대신 추가 사용자를 제공해서 필요한 작업을 수행하는 데 필요한 권한을 해당 사용자에게 부여합니다. IAM Identity Center 디렉터리에 사람을 추가하거나, 외부 ID 공급자를 IAM Identity Center 또는 IAM과 페더레이션하거나, 최소 권한 IAM 사용자를 생성하여 사용자를 추가할 수 있습니다.

사용자를 설정한 후에는 특정 사용자에게 AWS 계정 액세스 권한을 부여하고 리소스에 액세스할 수 있는 권한을 제공할 수 있습니다.

모범 사례로, AWS는 사람이 IAM 역할을 맡아 AWS에 액세스하도록 하여 임시 자격 증명을 사용하도록 하는 것을 권장합니다. IAM Identity Center 디렉터리에서 ID를 관리하거나 ID 공급자와의 페더레이션을 사용하는 경우 모범 사례를 따르고 있는 것입니다.

용어

IAM 자격 증명 작업 시 다음과 같은 용어가 일반적으로 사용됩니다.

IAM 리소스

IAM 서비스는 다음 리소스를 저장합니다. 이 리소스는 IAM 콘솔에서 추가, 편집, 제거할 수 있습니다.

- IAM 사용자
- IAM 그룹
- IAM 역할
- 권한 정책
- 자격 증명 공급자 객체

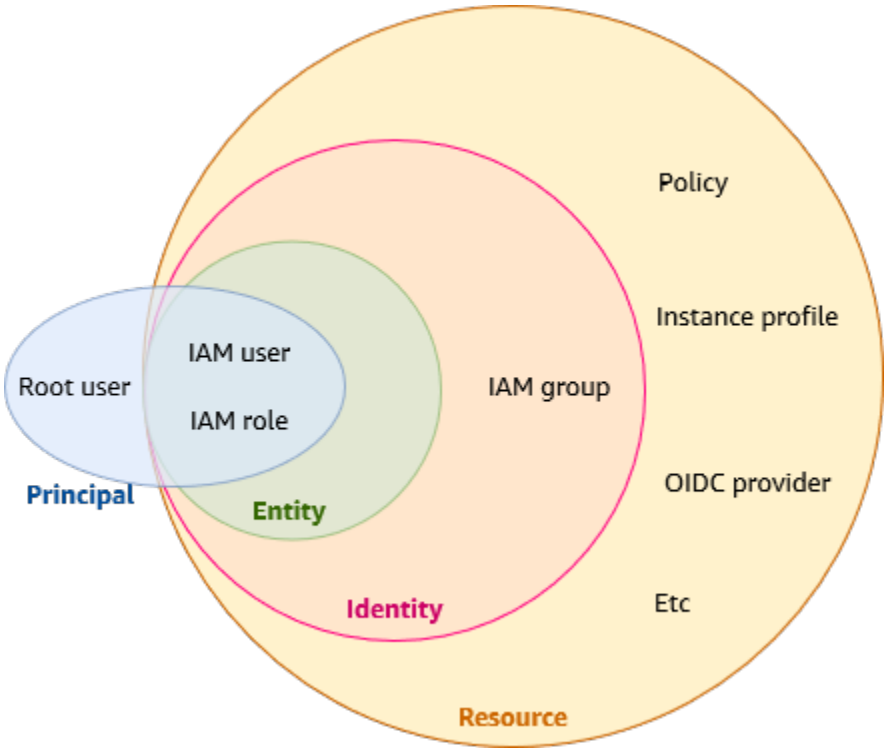
IAM 엔터티

AWS이(가) 인증에 사용하는 IAM 리소스. 리소스 기반 정책에서 엔터티를 보안 주체로 지정합니다.

- IAM 사용자
- IAM 역할

IAM 자격 증명

정책에서 권한을 부여받아 작업을 수행하고 리소스에 액세스할 수 있는 IAM 리소스입니다. ID에는 IAM 사용자, IAM 그룹 및 IAM 역할이 포함됩니다.



보안 주체

AWS 리소스에 대한 작업 또는 연산을 요청할 수 있는 AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할입니다. 보안 주체에는 인간 사용자, 워크로드, 페더레이션 사용자 및 수임된 역할이 포함됩니다. 인증 후 IAM은 보안 주체 유형에 따라 AWS에 요청할 수 있는 영구 또는 임시 보안 인증 정보를 보안 주체에게 부여합니다.

인간 사용자는 인간 자격 증명이라고도 하며, 애플리케이션의 사용자, 관리자, 개발자, 운영자, 소비자 등입니다.

워크로드는 애플리케이션, 프로세스, 운영 도구 및 기타 구성 요소 같이 비즈니스 가치를 창출하는 리소스 및 코드 모음입니다.

페더레이션 사용자는 Active Directory, Okta 또는 Microsoft Entra 같은 다른 자격 증명 공급자가 자격 증명과 보안 인증 정보를 관리하는 사용자입니다.

IAM 역할은 계정에서 생성할 수 있고 해당 자격 증명이 수행할 수 있는 작업과 수행할 수 없는 작업을 결정하는 특정 권한을 지닌 IAM 자격 증명입니다. 그러나 역할은 한 사람하고만 연관되지 않고 해당 역할이 필요한 사람이라면 누구든지 맡을 수 있어야 합니다.

IAM은 IAM 사용자와 루트 사용자에게 장기 보안 인증 정보를 부여하고 IAM 역할에 임시 보안 인증 정보를 부여합니다. 페더레이션 사용자와 AWS IAM Identity Center의 사용자는 AWS에 로그인할 때 IAM 역할을 수임하고 임시 보안 인증 정보를 부여받습니다. 가장 좋은 방법은 인간 사용자와 워크로드가 임시 보안 인증을 사용하여 AWS 리소스에 액세스하도록 하는 것입니다.

IAM 사용자와 IAM Identity Center의 사용자의 차이점

IAM 사용자는 별도의 계정이 아니라 계정 내의 사용자입니다. 각 사용자에게는 AWS Management Console에 액세스하기 위한 자체 암호가 있습니다. 또한 사용자가 계정의 리소스를 사용하기 위한 프로그래밍 방식의 요청을 할 수 있도록 각 사용자에 대한 개별 액세스 키를 생성할 수 있습니다.

IAM 사용자와 해당 액세스 키에는 AWS 리소스에 대한 장기 보안 인증 정보가 있습니다. IAM 사용자의 주요 용도는 IAM 역할을 사용할 수 없는 워크로드에 API 또는 CLI를 사용하여 AWS 서비스에 프로그래밍 방식으로 요청하는 기능을 제공하는 것입니다.

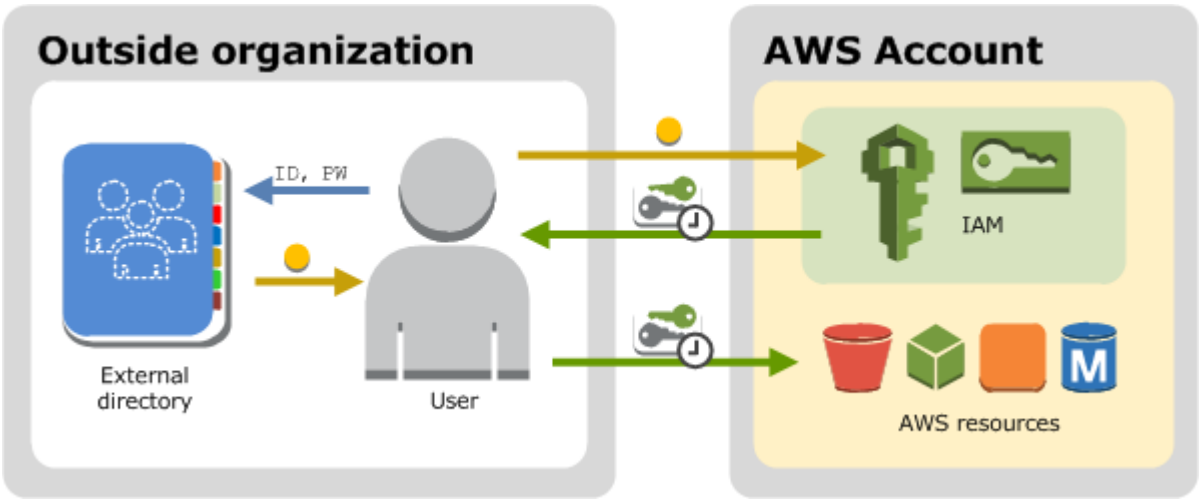
> **Note**
>
> 프로그래밍 방식 액세스 권한과 장기 보안 인증이 있는 IAM 사용자가 필요한 시나리오의 경우 필요할 때 액세스 키를 업데이트하는 것이 좋습니다. 자세한 내용은 액세스 키 업데이트 단원을 참조하십시오.

작업 인력 자격 증명(사람)은 수행하는 역할에 따라 권한 요구 사항이 다르고 조직 전체의 다양한 AWS 계정에서 작업할 수 있는 AWS IAM Identity Center의 사용자입니다. 액세스 키가 필요한 사용 사례가 있는 경우 AWS IAM Identity Center의 사용자를 사용하여 해당 사용 사례를 지원할 수 있습니다. AWS 액세스 포털을 통해 로그인하는 사용자는 AWS 리소스에 대한 단기 보안 인증 정보가 포함된 액세스 키를 얻을 수 있습니다. 중앙 액세스 관리를 위해 AWS IAM Identity Center(IAM Identity Center)를 사용하여 계정에 대한 액세스 권한과 해당 계정 내 권한을 관리하는 것이 좋습니다. IAM Identity Center는 사용자 및 그룹을 추가하고 AWS 리소스에 대한 액세스 수준을 할당할 수 있는 기본 ID 소스로 Identity Center 디렉터리를 사용하여 자동으로 구성됩니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 AWS IAM Identity Center란 무엇입니까? 단원을 참조하세요.

이 두 유형의 사용자 간의 주요 차이점은 IAM Identity Center의 사용자는 관리 콘솔 또는 AWS 리소스에 액세스하기 전에 AWS에 로그인할 때 자동으로 IAM 역할을 수임한다는 것입니다. IAM 역할은 사용자가 AWS에 로그인할 때마다 임시 보안 인증 정보를 부여합니다. IAM 사용자가 IAM 역할을 사용하여 로그인하려면 역할을 수임하고 전환할 수 있는 권한이 있어야 하며, AWS 계정에 액세스한 후 수임하려는 역할로의 전환을 명시적으로 선택해야 합니다.

기존 자격 증명 소스의 사용자 페더레이션

조직 내 사용자가 회사 네트워크에 로그인할 때 이미 인증된 경우 해당 사용자를 위해 별도의 IAM 사용자 또는 IAM Identity Center 사용자를 생성할 필요가 없습니다. 대신 IAM이나 AWS IAM Identity Center 사용을 통해 이러한 사용자 자격 증명을 AWS에 페더레이션할 수 있습니다. 페더레이션 사용자는 특정 리소스에 액세스할 수 있는 권한을 부여하는 IAM 역할을 수임합니다. 역할에 관한 자세한 내용은 역할 용어 및 개념 단원을 참조하세요.



페더레이션은 다음과 같은 경우에 유용합니다.

- 사용자가 이미 기업 디렉터리에 있는 경우.

  기업 디렉터리가 SAML 2.0(Security Assertion Markup Language 2.0)과 호환되는 경우, 기업 디렉터리를 구성하여 사용자에게 AWS Management Console에 대한 Single-Sign On(SSO) 액세스를 제공할 수 있습니다. 자세한 내용은 임시 자격 증명과 관련된 일반적인 시나리오 단원을 참조하십시오.

  기업 디렉터리가 SAML 2.0과 호환되지 않는 경우, ID 브로커 애플리케이션을 생성하여 사용자에게 AWS Management Console에 대한 Single-Sign On(SSO) 액세스를 제공할 수 있습니다. 자세한 내용은 사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기 단원을 참조하십시오.

기업 디렉토리가 Microsoft Active Directory인 경우, AWS IAM Identity Center를 사용하여 Active Directory의 자체 관리형 디렉토리 또는 AWS Directory Service의 디렉토리에 연결해 기업 디렉토리와 AWS 계정 간의 신뢰를 설정할 수 있습니다.

Okta 또는 Microsoft Entra와 같은 외부 ID 제공업체(IdP)를 사용하여 사용자를 관리하는 경우 AWS IAM Identity Center을 사용하여 IdP와 AWS 계정 간에 신뢰를 설정할 수 있습니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 외부 ID 제공업체에 연결을 참조하세요.

- 사용자가 이미 인터넷 자격 증명을 보유한 경우

  사용자가 Login with Amazon, Facebook, Google 또는 OpenID Connect(OIDC) 호환 자격 증명 공급자 등의 인터넷 자격 증명 공급자를 통해 자신을 식별할 수 있도록 모바일 앱 또는 웹 기반 앱을 만들면, 해당 앱에서 연동을 통해 AWS에 액세스할 수 있습니다. 자세한 내용은 OIDC 페더레이션 단원을 참조하십시오.

> **Tip**
>
> 인터넷 자격 증명 공급자를 통해 ID 페더레이션을 사용하려면 Amazon Cognito를 사용하는 것이 좋습니다.

사용자 액세스를 제공하는 다양한 방법

다음은 AWS 리소스에 대한 액세스 권한을 제공할 수 있는 방법입니다.

사용자 액세스 유형	언제 사용되나요?	자세한 정보는 어디에 있나요?
작업 인력 사용자 등 사람이 IAM Identity Center를 사용하여 AWS 리소스에 액세스하기 위한 Single	IAM Identity Center는 사용자의 관리와 AWS 계정 및 클라우드 애플리케이션에 대한 액세스를 통합하는 중앙 위치를 제공합니다. IAM Identity Center 내에서 ID 스토어를 설정하거나 기존 ID 제공업체(idP)와의 페더레이션을 구성할 수 있습니다. 보안상	IAM Identity Center 설정에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 Getting Started(시작하기)를 참조하세요. IAM Identity Center의 MFA 사용에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 Multi-factor authentication(다중 인증)을 참조하세요.

사용자 액세스 유형	언제 사용되나요?	자세한 정보는 어디에 있나요?
Sign-On 액세스	가장 좋은 방법은 인간 사용자에게 AWS 리소스에 대한 제한된 보안 인증 정보를 부여하는 것입니다. 사용자는 보다 쉽게 로그인할 수 있으며 단일 시스템에서 리소스에 대한 액세스를 제어할 수 있습니다. IAM Identity Center는 추가 계정 보안을 위해 다중 인증(MFA)을 지원합니다.	
작업 인력 사용자 등 인간 사용자가 IAM ID 제공업체(IdP)를 사용하여 AWS 서비스에 액세스하기 위한 페더레이션 액세스	IAM은 OpenID Connect(OIDC) 또는 SAML 2.0(Security Assertion Markup Language 2.0)과 호환되는 IdP를 지원합니다. IAM ID 제공업체를 생성한 후 페더레이션 사용자에게 동적으로 할당할 수 있는 IAM 역할을 하나 이상 생성합니다.	IAM ID 제공업체 및 페더레이션에 대한 자세한 내용은 자격 증명 공급자 및 페더레이션 섹션을 참조하세요.

사용자 액세스 유형	언제 사용되나요?	자세한 정보는 어디에 있나요?
AWS 계정 간 크로스 계정 액세스	일부 AWS 리소스에 대한 액세스를 AWS 계정의 사용자와 공유하려고 합니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스는 (역할을 프록시로 사용하는 대신) 정책을 리소스에 직접 연결할 수 있는 리소스 기반 정책을 지원합니다.	IAM 역할에 대한 자세한 내용은 IAM 역할 섹션을 참조하세요. 서비스 연결 역할에 대한 자세한 내용은 서비스 연결 역할 생성를 참조하세요. 서비스 연결 역할 사용을 지원하는 서비스에 대한 자세한 내용은 AWS IAM으로 작업하는 서비스 섹션을 참조하세요. 서비스 연결 역할 열에 예가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 해당 열의 예와 연결된 링크를 선택하세요.

사용자 액세스 유형	언제 사용되나요?	자세한 정보는 어디에 있나요?
AWS 계정의 지정된 IAM 사용자를 위한 장기 보안 인증	AWS에서 IAM 사용자의 장기 보안 인증이 필요한 특정 사용 사례가 있을 수 있습니다. IAM을 사용하여 AWS 계정에서 이러한 IAM 사용자를 생성하고 IAM을 사용하여 해당 권한을 관리할 수 있습니다. 일부 사용 사례는 다음과 같습니다. • IAM 역할을 사용할 수 없는 워크로드 • 액세스 키를 통한 프로그래밍 방식 액세스가 필요한 타사 AWS 클라이언트 • AWS CodeCommit 또는 Amazon Keyspaces의 서비스별 자격 증명 • 사용자 계정에 AWS IAM Identity Center을 사용할 수 없으며 다른 ID 공급자가 없음 프로그래밍 방식 액세스 및 장기 보안 인증이 있는 IAM 사용자가 필요한 시나리오에서는 모범 사례로, 필요할 때 액세스 키를 업데이트하는 것이 좋습니다. 자세한 내용은 액세스 키 업데이트 단원을 참조하십시오.	IAM 사용자 설정에 대한 자세한 내용은 AWS 계정에서 IAM 사용자 생성 섹션을 참조하세요. IAM 사용자 액세스 키에 대한 자세한 내용은 IAM 사용자의 액세스 키 관리 섹션을 참조하세요. AWS CodeCommit 또는 Amazon Keyspaces에 대한 서비스별 보안 인증에 대한 자세한 내용은 CodeCommit용 IAM 자격 증명: Git 자격 증명, SSH 키 및 AWS 액세스 키 및 Amazon Keyspaces(Apache Cassandra용)와 함께 IAM 사용 섹션을 참조하세요.

프로그래밍 방식의 사용자 액세스 지원

사용자가 AWS Management Console 외부에서 AWS와 상호 작용하려면 프로그래밍 방식의 액세스가 필요합니다. 프로그래밍 방식으로 액세스를 부여하는 방법은 AWS에 액세스하는 사용자 유형에 따라 다릅니다.

- IAM Identity Center에서 ID를 관리하는 경우 AWS API에는 프로필이 필요하고 AWS Command Line Interface에는 프로필이나 환경 변수가 필요합니다.
- IAM 사용자가 있는 경우 AWS API 및 AWS Command Line Interface에는 액세스 키가 필요합니다. 가능한 경우 액세스 키 ID, 비밀 액세스 키 및 보안 인증 정보가 만료되는 시간을 나타내는 보안 토큰으로 구성된 임시 보안 인증 정보를 만듭니다.

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	옵션	추가 정보
작업 인력 ID (IAM Identity Center에서 관리되는 사람 및 사용자)	단기 보안 인증 정보를 사용하여 AWS CLI 또는 AWS API에 대한 프로그래밍 방식 요청에 직접 또는 AWS SDK를 사용하여 서명하려 합니다.	AWS CLI의 경우 AWS IAM Identity Center 사용 설명서의 CLI 액세스를 위한 IAM 역할 보안 인증 정보 가져오기에 나와 있는 지침을 따르세요. AWS API의 경우 AWS SDK 및 도구 참조 가이드의 SSO 보안 인증 정보에 나와 있는 지침을 따르세요.
IAM 사용자	단기 보안 인증 정보를 사용하여 AWS CLI 또는 AWS API에 대한 프로그래밍 방식 요청에 직접 또는 AWS SDK를 사용하여 서명하려 합니다.	AWS 리소스로 임시 보안 인증 정보 사용에 나와 있는 지침을 따르세요.
IAM 사용자	장기 보안 인증 정보를 사용하여 AWS CLI 또는 AWS API에 대한 프로그래밍 방식 요청에	IAM 사용자 액세스 키 관리에 나와 있는 지침을 따르세요.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	옵션	추가 정보
	직접 또는 AWS SDK를 사용하여 서명할 수 있습니다. (권장되지 않음)	
페더레이션 사용자	AWS STS API 작업을 사용하여 액세스 키 페어 및 세션 토큰을 포함하는 임시 보안 인증 정보로 새 세션을 생성합니다.	API 작업에 대한 설명은 the section called “임시 보안 자격 증명 요청” 섹션을 참조하세요.

권한 및 정책이 액세스 관리를 제공하는 방법

AWS Identity and Access Management(IAM)의 액세스 관리를 통해 계정에서 보안 주체 엔터티에 허용된 권한을 정의할 수 있습니다. 보안 주체 엔터티란 IAM 엔터티(IAM 사용자 또는 IAM 역할)를 사용하여 인증된 사람 또는 애플리케이션입니다. 액세스 관리를 흔히 권한 부여라고 합니다. 정책을 생성하고 IAM ID(IAM 사용자, IAM 그룹 또는 IAM 역할) 또는 AWS 리소스에 연결하여 AWS에서 액세스를 관리합니다. 정책은 ID 또는 리소스와 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. AWS는 보안 주체가 IAM 엔터티(IAM 사용자 또는 IAM 역할)를 사용하여 요청할 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 AWS에 JSON 문서로 저장됩니다. 정책 유형 및 활용에 대한 자세한 정보는 IAM의 정책 및 권한 섹션을 참조하세요.

정책 및 계정

AWS에서 하나의 계정을 관리하려면 정책을 사용하여 해당 계정 내 권한을 정의합니다. 여러 계정 전반의 권한을 관리하고자 한다면 IAM 사용자에 대한 권한을 관리하기가 더 어렵습니다. 크로스 계정 권한에 대해 IAM 역할, 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 사용할 수 있습니다. 하지만 여러 계정을 소유하는 경우에는 이러한 권한을 쉽게 관리할 수 있도록 ACL 대신에 AWS Organizations 서비스를 사용하는 것이 좋습니다. 자세한 내용은 Organizations 사용 설명서에서 AWS Organizations란 무엇인가요?를 참조하세요.

정책 및 사용자

IAM 사용자는 AWS 계정의 ID입니다. IAM 사용자를 생성할 경우, 권한을 부여하지 않는 한 사용자는 계정 내에서 어떠한 것으로도 액세스할 수 없습니다. IAM 사용자 또는 IAM 사용자가 속한 IAM 그룹에 연결된 정책인 자격 증명 기반 정책을 생성하여 IAM 사용자에게 권한을 부여합니다. 다음 예제

는 IAM 사용자가 us-east-2 리전 내의 123456789012 계정에서 Books 테이블의 모든 Amazon DynamoDB 작업(dynamodb:*)을 수행할 수 있도록 허용하는 JSON 정책을 보여줍니다.

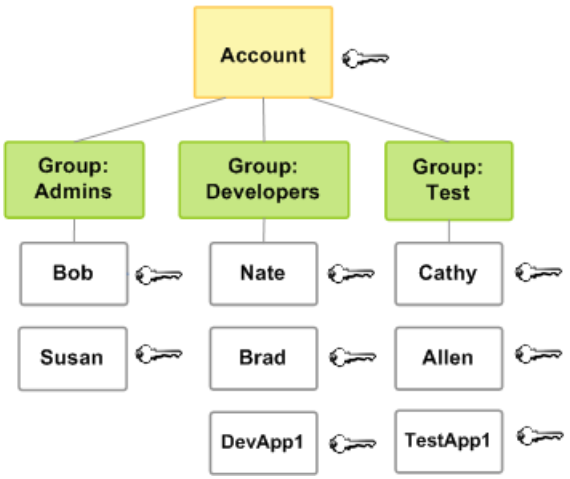
```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "dynamodb:*",
    "Resource": "arn:aws:dynamodb:us-east-2:123456789012:table/Books"
  }
}
```

이 정책을 IAM 사용자에게 연결하면 해당 사용자는 DynamoDB 인스턴스의 Books 테이블에서 모든 작업을 수행할 권한을 갖습니다. 대부분의 IAM 사용자에게는 부여된 전체 권한을 나타내는 여러 정책의 조합이 있습니다.

정책에 의해 명시적으로 허용되지 않은 작업 또는 리소스는 기본적으로 모두 거부됩니다. 예를 들어 앞의 정책이 사용자에게 연결된 단일 정책이라면 이 사용자는 Books 테이블에서 DynamoDB 작업을 수행할 수 있지만 다른 테이블에서는 작업을 수행할 수 없습니다. 마찬가지로 이 사용자는 Amazon EC2, Amazon S3 또는 기타 다른 AWS 서비스에서 어떠한 작업도 수행할 수 없습니다. 이러한 서비스에서 작업할 수 있는 권한이 정책에 포함되지 않기 때문입니다.

정책 및 IAM 그룹

IAM 사용자를 IAM 그룹으로 구성하고 IAM 그룹에 정책을 연결할 수 있습니다. 이 경우 각 IAM 사용자는 별도의 자격 증명을 갖고 있지만 IAM 그룹의 모든 IAM 사용자에게는 IAM 그룹에 연결된 권한이 있습니다. IAM 그룹을 사용하여 간편하게 권한을 관리합니다.



IAM 사용자 또는 IAM 그룹에 여러 정책을 연결하여 다양한 권한을 부여할 수 있습니다. 이 경우 정책의 조합이 보안 주체의 유효 권한을 결정합니다. 특정 작업과 리소스 모두에 대한 명시적 `Allow` 권한이 없는 보안 주체에게는 이러한 권한이 없는 것입니다.

연동 사용자 및 역할

페더레이션 사용자에게는 IAM 사용자처럼 AWS 계정에서 영구적인 ID가 부여되지 않습니다. 연동 사용자에게 권한을 부여하려면 역할이라고 하는 개체를 만들어 해당 역할의 권한을 정의할 수 있습니다. 연동 사용자가 AWS에 로그인하면 사용자가 역할과 연결되고 역할에 정의된 권한이 부여됩니다. 자세한 내용은 타사 ID 제공업체의 역할 생성(페더레이션) 단원을 참조하십시오.

자격 증명 기반 정책 및 리소스 기반 정책

자격 증명 기반 정책은 IAM 사용자, 그룹 또는 역할과 같은 IAM 자격 증명에 연결할 수 있는 권한 정책입니다. 리소스 기반 정책은 Amazon S3 버킷 또는 IAM 역할 신뢰 정책과 같은 리소스에 연결하는 권한 정책입니다.

자격 증명 기반 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. 자격 증명 기반 정책을 추가로 분류할 수 있습니다.

- 관리형 정책 - AWS 계정에 속한 다수의 사용자, 그룹 및 역할에 독립적으로 연결할 수 있는 자격 증명 기반 정책입니다. 사용할 수 있는 관리형 정책은 두 가지가 있습니다.
  - AWS 관리형 정책 - AWS에서 생성 및 관리하는 관리형 정책입니다. 정책 사용이 처음이라면 AWS 관리형 정책 사용을 먼저 권장합니다.
  - 고객 관리형 정책 - 사용자가 자신의 AWS 계정에서 생성 및 관리하는 관리형 정책입니다. 고객 관리형 정책은 AWS 관리형 정책보다 정책에 대해 더욱 정밀하게 제어할 수 있습니다. 시각적 편집기에서 또는 JSON 정책 문서를 직접 생성하여 IAM 정책을 생성, 편집 및 검증할 수 있습니다. 자세한 내용은 고객 관리형 정책으로 사용자 지정 IAM 권한 정의 및 IAM 정책 편집을(를) 참조하세요.
- 인라인 정책 - 자신이 생성 및 관리하며, 단일 사용자, 그룹 또는 역할에 직접 포함되는 정책입니다. 대부분의 경우 인라인 정책을 사용하지 않는 것이 좋습니다.

리소스 기반 정책은 지정된 보안 주체가 해당 리소스에 대해 수행할 수 있는 작업 및 이에 관한 조건을 제어합니다. 리소스 기반 정책은 인라인 정책이며, 관리형 리소스 기반 정책은 없습니다. 크로스 계정 액세스를 활성화하는 경우, 전체 계정이나 다른 계정의 IAM 엔터티를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다.

IAM 서비스는 역할 신뢰 정책이라고 하는 리소스 기반 정책 유형 하나를 지원하며, 이 정책을 IAM 역할에 연결합니다. IAM 역할은 리소스 기반 정책을 지원하는 자격 증명이자 리소스이므로 신뢰 정책과 자격 증명 기반 정책 모두 IAM 역할에 연결해야 합니다. 신뢰 정책은 역할을 수임할 수 있는 보안 주체 엔터티(계정, 사용자, 역할 및 페더레이션 사용자)를 정의합니다. IAM 역할과 다른 리소스 기반 정책 간의 차이에 대해 알아보려면 IAM의 크로스 계정 리소스 액세스 섹션을 참조하세요.

리소스 기반 정책을 지원하는 서비스를 보려면 AWS IAM으로 작업하는 서비스 단원을 참조하십시오. 리소스 기반 정책에 대해 자세히 알아보려면 자격 증명 기반 정책 및 리소스 기반 정책 섹션을 참조하세요.

ABAC 권한 부여를 통한 속성 기반 권한 정의

ABAC(속성 기반 액세스 제어)는 속성을 기준으로 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이 속성을 태그라고 합니다. IAM 엔터티(IAM 사용자 또는 IAM 역할)를 포함하는 IAM 리소스와 AWS 리소스에 태그를 연결할 수 있습니다. IAM 보안 주체에 대해 단일 ABAC 정책 또는 작은 정책 세트를 생성할 수 있습니다. 보안 주체의 태그가 리소스 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계할 수 있습니다. ABAC의 속성 시스템은 높은 사용자 컨텍스트와 세분화된 액세스 제어를 모두 제공합니다. ABAC는 속성 기반이므로 실시간으로 액세스 권한을 부여하거나 취소하는 데이터 또는 애플리케이션에 대해 동적 권한 부여를 수행할 수 있습니다. ABAC는 확장 중인 환경과 자격 증명 또는 리소스 정책 관리가 복잡해진 상황에서 유용합니다.

예를 들어, `access-project` 태그 키를 사용하여 세 개의 IAM 역할을 생성할 수 있습니다. 첫 번째 IAM 역할의 태그 값을 `Heart`로, 두 번째 역할의 태그 값을 `Star`로, 세 번째 역할의 태그 값을 `Lightning`으로 설정합니다. 그런 다음 IAM 역할과 AWS 리소스에 태그 값 `access-project`가 있을 때 액세스를 허용하는 단일 정책을 사용할 수 있습니다. AWS에서 ABAC를 사용하는 방법을 보여주는 자세한 자습서는 IAM 튜토리얼: 태그를 기반으로 AWS 리소스에 액세스할 수 있는 권한 정의 섹션을 참조하세요. ABAC를 지원하는 서비스에 대해 알아보려면 AWS IAM으로 작업하는 서비스을 참조하세요.

ABAC와 기존 RBAC 모델 비교

IAM에 사용되는 기존 권한 부여 모델은 RBAC(역할 기반 액세스 제어)입니다. RBAC는 IAM 역할과 구별되는 개인의 직무나 역할에 따라 권한을 정의합니다. IAM에는 직무에 관한 관리형 정책이 있어 RBAC 모델의 작업 기능에 대한 사용 권한을 정렬할 수 있습니다.

IAM에서는 다양한 직무에 대해 서로 다른 정책을 생성하여 RBAC를 구현합니다. 그런 다음 정책을 자격 증명(IAM 사용자, IAM 그룹 또는 IAM 역할)에 연결합니다. 가장 좋은 방법은 직무에 필요한 최소 권한을 부여하는 것입니다. 이로 인한 결과가 최소 권한 액세스입니다. 각 직능 정책에는 해당 정책이 할

당된 자격 증명이 액세스할 수 있는 특정 리소스가 나열되어 있습니다. 기존 RBAC 모델을 사용하면 사용자가 환경에 새 리소스를 추가할 때 해당 리소스에 액세스할 수 있도록 정책을 업데이트해야 한다는 단점이 있습니다.

예를 들어, 직원이 작업 중인 세 개의 프로젝트 `Heart`, `Star` 및 `Lightning`이 있다고 가정합니다. 각 프로젝트에 대한 IAM 역할을 생성합니다. 그런 다음 각 IAM 역할에 정책을 연결하여 IAM 역할을 맡을 수 있는 모든 사람이 액세스할 수 있는 리소스를 정의합니다. 직원이 회사 내에서 직무를 변경하는 경우 다른 IAM 역할에 해당 직무를 할당합니다. 사람이나 프로그램을 둘 이상의 IAM 역할에 할당할 수 있습니다. 그러나 `Star` 프로젝트에 새 Amazon EC2 컨테이너와 같은 추가 리소스가 필요할 수 있습니다. 이 경우 `Star` IAM 역할에 연결된 정책을 업데이트하여 새 컨테이너 리소스를 지정해야 합니다. 그렇지 않으면 `Star` 프로젝트 멤버가 새 컨테이너에 액세스할 수 없습니다.

ABAC는 전통적인 RBAC 모델에 비해 다음과 같은 이점을 제공합니다.

- ABAC 권한은 혁신적으로 확장됩니다. 관리자가 새 리소스에 액세스할 수 있도록 기존 정책을 업데이트할 필요가 없습니다. 예를 들어, `access-project` 태그로 ABAC 전략을 설계했다고 가정합니다. 개발자는 `access-project = Heart` 태그와 함께 IAM 역할을 사용합니다. `Heart` 사용자에게 추가 Amazon EC2 리소스가 필요한 경우 개발자는 `access-project = Heart` 태그를 사용하여 새 Amazon EC2 인스턴스를 생성할 수 있습니다. 그러면 `Heart` 프로젝트의 모든 사용자가 태그 값이 일치하기 때문에 해당 인스턴스를 시작하고 중지할 수 있습니다.
- ABAC를 사용하면 필요한 정책 수가 적어집니다. 각 직무에 대해 서로 다른 정책을 생성할 필요가 없기 때문에 생성해야 하는 정책이 더 적습니다. 그러므로 정책을 관리하기가 더 쉽습니다.
- ABAC를 사용하면 팀이 변화와 성장에 동적으로 대응할 수 있습니다. 새 리소스에 대한 권한이 속성에 따라 자동으로 부여되므로 자격 증명에 정책을 수동으로 할당할 필요가 없습니다. 예를 들어, 회사에서 이미 ABAC를 사용하여 `Heart` 및 `Star` 프로젝트를 지원하는 경우 새 `Lightning` 프로젝트를 쉽게 추가할 수 있습니다. IAM 관리자는 `access-project = Lightning` 태그와 함께 새 IAM 역할을 생성합니다. 새 프로젝트를 지원하기 위해 정책을 변경할 필요는 없습니다. IAM 역할을 맡을 권한이 있는 모든 사용자는 `access-project = Lightning` 태그가 지정된 인스턴스를 생성하고 볼 수 있습니다. 또 다른 시나리오는 팀원이 `Heart` 프로젝트에서 `Lightning` 프로젝트로 옮기는 경우입니다. 팀원에게 `Lightning` 프로젝트에 대한 액세스 권한을 부여하기 위해 IAM 관리자는 팀원에게 다른 IAM 역할을 할당합니다. 권한 정책을 변경할 필요가 없습니다.
- ABAC를 사용하여 세분화된 권한을 사용할 수 있습니다. 정책을 생성할 때는 최소 권한을 부여하는 것이 가장 좋습니다. 기존 RBAC를 사용하는 경우에는 특정 리소스에 대한 액세스를 허용하는 정책을 작성합니다. 그러나 ABAC를 사용하는 경우 리소스의 태그가 보안 주체의 태그와 일치하는 경우에만 모든 리소스에 대한 작업을 허용할 수 있습니다.
- ABAC를 사용하여 회사 디렉터리의 직원 속성을 사용합니다. 세션 태그를 IAM에 전달하도록 SAML 또는 OIDC 공급자를 구성할 수 있습니다. 직원이 AWS에 페더레이션되면 IAM은 해당 속성을 결과

보안 주체에 적용합니다. 그런 다음 ABAC를 사용하여 이러한 속성에 따라 권한을 허용하거나 거부할 수 있습니다.

AWS에서 ABAC를 사용하는 방법을 보여 주는 자세한 자습서는 IAM 튜토리얼: 태그를 기반으로 AWS 리소스에 액세스할 수 있는 권한 정의 섹션을 참조하세요.

IAM 시작하기

AWS Identity and Access Management(IAM)으로 Amazon Web Services(AWS) 및 계정의 리소스에 대한 액세스 권한을 안전하게 제어할 수 있습니다. IAM은 로그인 보안 인증 정보를 비공개로 유지할 수도 있습니다. 하지만 IAM을 사용하려고 따로 가입할 필요는 없습니다. IAM 사용에는 요금이 부과되지 않습니다.

IAM을 사용하여 사용자 및 역할과 같은 ID에 계정의 리소스에 대한 액세스 권한을 부여합니다. 예를 들어AWS 외부에서 관리하는 회사 디렉터리의 기존 사용자와 함께 IAM을 사용하거나 AWS IAM Identity Center를 사용하여 AWS에서 사용자를 생성할 수 있습니다. 페더레이션형 ID는 정의된 IAM 역할을 수임하여 필요한 리소스에 액세스합니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?(IAM Identity Center란 무엇인가요?) 섹션을 참조하세요.

> **Note**
>
> IAM은 여러 AWS 제품에 통합됩니다. IAM 지원 서비스 목록은 AWS IAM으로 작업하는 서비스 섹션을 참조하세요.

AWS를 시작하고, 관리 사용자와 Organizations를 생성하고, 여러 서비스를 사용하여 프로젝트 구축 및 개시 등과 같은 문제를 해결하는 방법을 알아보려면 리소스 센터 시작하기 섹션을 참조하세요.

AWS 계정 설정

IAM 작업을 시작하기 전에 AWS 환경의 초기 설정을 완료했는지 확인하세요.

AWS 계정이 없는 경우 다음 절차에 따라 계정을 생성합니다.

AWS 계정에 가입

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   AWS 계정 루트 사용자에 가입하면 AWS 계정 루트 사용자가 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관

리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS는 가입 절차 완료된 후 사용자에게 확인 이메일을 전송합니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

서비스에 등록할 때 이메일 주소와 암호를 사용하여 AWS 계정을 생성했습니다. 그 주소와 암호가 바로 AWS 루트 사용자 자격 증명입니다. 일상 작업을 위해 AWS에 액세스할 때 루트 사용자 자격 증명을 사용하지 않는 것이 가장 좋습니다. 루트 사용자 자격 증명이 필요한 작업을 수행할 때만 루트 사용자 자격 증명을 사용하세요. 또한 자격 증명을 다른 사람과 공유하지 마세요. 대신 디렉터리에 사람을 추가하고 AWS 계정 액세스 권한을 부여하세요.

AWS 계정 루트 사용자 보안 유지

1. 루트 사용자를 선택하고 AWS 계정이메일 주소를 입력하여 AWS Management Console에 계정 소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 루트 사용자로 로그인을 참조하십시오.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자용 가상 MFA 디바이스 활성화(콘솔)를 참조하십시오.

빌링 콘솔에 대한 액세스 권한 부여

IAM 사용자와 AWS 계정 내의 역할은 기본적으로 과금 정보 및 비용 관리 콘솔 페이지에 액세스할 수 없습니다. 이는 특정 청구 기능에 대한 액세스 권한을 부여하는 IAM 정책을 보유한 경우에도 마찬가지입니다. 액세스 권한을 부여하려면 AWS 계정 루트 사용자가 먼저 IAM 액세스를 활성화해야 합니다.

> **Note**
>
> 보안 모범 사례로 AWS IAM Identity Center과의 ID 페더레이션을 통해 리소스에 대한 액세스를 제공하는 것이 좋습니다. AWS Organizations와 함께 IAM Identity Center를 활성화하면 과금 정보 및 비용 관리 콘솔이 기본적으로 조직 내 모든 AWS 계정에 통합 결제와 함께 활성화됩니다. 자세한 내용은 과금 정보 및 비용 관리 사용 설명서에서 AWS Organizations 통합 결제를 참조하세요.

1. 루트 사용자 보안 인증(AWS 계정을 만들 때 사용한 이메일 주소 및 암호)을 사용하여 AWS Management Console에 로그인합니다.
2. 탐색 표시줄에서 계정 이름을 선택한 다음 계정을 선택합니다.
3. 결제 정보에 대한 IAM 사용자 및 역할 액세스 섹션이 나타날 때까지 페이지를 아래로 스크롤한 다음 편집을 선택합니다.
4. IAM 액세스 활성화(Activate IAM Access) 확인란을 선택하여 Billing and Cost Management 콘솔 페이지에 대한 액세스를 활성화합니다.
5. 업데이트를 선택합니다.

   페이지에 결제 정보에 대한 IAM 사용자/역할 액세스가 활성화됨 메시지가 표시됩니다.

   > **Important**
   >
   > IAM 액세스 활성화만으로는 사용자 및 역할에 과금 정보 및 비용 관리 콘솔 페이지를 볼 권한이 부여되지 않습니다. 또한 결제 콘솔에 대한 액세스를 부여하려면 필요한 ID 기반 정책을 IAM 역할에 연결해야 합니다. 역할은 사용자가 필요할 때 수임할 수 있는 임시 자격 증명을 제공합니다.

6. AWS Management Console을 사용하여 결제 콘솔에 액세스하기 위해 수임할 수 있는 역할을 생성합니다.
7. 역할의 권한 추가 페이지에서 AWS 계정의 결제 콘솔에 관한 세부 정보를 나열하고 볼 수 있는 권한을 추가합니다.

   AWS 관리형 정책 Billing은 사용자에게 과금 정보 및 비용 관리 콘솔을 보고 편집할 수 있는 권한을 부여합니다. 여기에는 계정 사용량 보기 권한, 예산 및 결제 방법 수정 권한이 포함됩니다. IAM 역할에 연결하여 계정의 결제 정보에 대한 액세스를 제어할 수 있는 추가 정책 예제는 과금 정보 및 비용 관리 사용 설명서의 결제 AWS Billing 정책 예제를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center 설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   IAM Identity Center 디렉터리를 ID 소스로 사용하는 방법에 대한 자습서는 AWS IAM Identity Center 사용 설명서의 기본 IAM Identity Center 디렉터리로 사용자 액세스 구성을 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM IDentity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자로 로그인하는 데 도움이 필요한 경우 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하십시오.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

계정에 초기 IAM 서비스 설정

AWS Identity and Access Management는 AWS 리소스에 대한 액세스를 안전하게 관리하는 데 도움이 되는 기본 AWS 서비스입니다. IAM 관리는 다양한 사용자 유형 지원부터 암호, 권한 및 보안 자격 증명 관리에 이르기까지 사용자 액세스 및 권한을 제어할 수 있는 광범위한 책임을 제공합니다. AWS 환경을 처음 설정할 때는 다음과 같은 결정을 내립니다.

- AWS 연결에 사용하는 URL
- 환경에서 ID를 구성하는 방법
- 다양한 작업을 수행하는 데 필요한 권한
- 환경에서 지원할 역할

ID 및 액세스 관리 시스템의 다양한 구성 요소를 생성할 때 해당 환경에서 작업한 다른 항목을 다시 참조해야 할 수도 있습니다. IAM은 항목을 빠르고 쉽게 찾을 수 있는 검색 기능을 제공합니다.

AWS 계정 ID 보기

콘솔에 로그인한 경우 다음 방법을 사용하여 AWS 계정의 계정 ID를 볼 수 있습니다.

AWS 계정 ID 보기

IAM console

AWS 계정 ID는 AWS 계정 섹션의 IAM 대시보드로 이동하면 표시됩니다. 콘솔에서 계정 ID를 보는 방법에는 사용자 유형에 따라 추가적인 방법이 있습니다. 역할을 수임한 경우 보안 인증을 사용할 수 없습니다.

사용자 유형	절차
루트 사용자	오른쪽 상단의 탐색 모음에서 사용자 이름을 선택한 다음 보안 인증을 선택합니다. 계정 번호가 계정 식별자 아래에 표시됩니다.
IAM 사용자	오른쪽 상단의 탐색 모음에서 사용자 이름을 선택합니다. 사용자 이름 위에 계정 ID가 표시됩니다. Security credentials(보안 자격 증명)를 선택합니다. 계정 번호가 계정 세부 정보 아래에 표시됩니다.
페더레이션 사용자	오른쪽 상단의 탐색 모음에서 사용자 이름을 선택합니다. 사용자 이름 위에 계정 ID가 표시됩니다.
수임된 역할	오른쪽 상단에 있는 탐색 모음에서 지원 아이콘을 선택한 후 목록에서 지원 센터를 선택합니다. 현재 로그인한 12자리 계정 번호(ID)는 지원 센터 탐색 창에 나타납니다.

AWS CLI

다음 명령을 사용하여 사용자 ID, 계정 ID 및 사용자 ARN을 봅니다.

- aws sts get-caller-identity

API

다음 API를 사용하여 사용자 ID, 계정 ID 및 사용자 ARN을 봅니다.

- GetCallerIdentity

AWS 계정 ID에 별칭 사용

계정 ID는 계정을 고유하게 식별하는 12자리 숫자입니다. 기본적으로 계정의 IAM 사용자는 계정 ID가 포함된 웹 URL을 사용하여 로그인합니다. URL이 없는 경우 로그인할 때 AWS 로그인 페이지에서 계정 ID를 제공할 수 있습니다.

로그인 페이지 URL의 형식은 기본적으로 다음과 같습니다.

```
https://Your_Account_ID.signin.aws.amazon.com/console/
```

많은 사람들이 숫자보다 단어를 기억하기 쉽다고 생각하므로 계정 ID의 별칭을 만들면 IAM 사용자가 더 쉽게 로그인할 수 있습니다.

AWS 계정 ID의 AWS 계정 별칭을 만드는 경우 로그인 페이지 URL이 다음 예제와 같습니다.

```
https://Your_Account_Alias.signin.aws.amazon.com/console/
```

계정 별칭 생성 시 고려 사항

- AWS 계정은 별칭을 하나만 가질 수 있습니다. AWS 계정의 새 별칭을 생성하면 새 별칭이 이전 별칭을 덮어쓰며 이전 별칭을 포함하는 URL이 작동하지 않습니다.
- 계정 별칭은 숫자, 소문자 및 하이픈만 포함해야 합니다. AWS 계정 엔터티에 대한 자세한 내용은 IAM 및 AWS STS 할당량 섹션을 참조하세요.
- 계정 별칭은 지정된 네트워크 파티션 내 모든 Amazon Web Services 제품에서 고유해야 합니다

  파티션은 AWS 리전의 그룹입니다. 각 AWS 계정은 하나의 파티션으로 범위가 지정됩니다.

  지원되는 파티션은 다음과 같습니다.

  - `aws` - AWS 리전
  - `aws-cn` - 중국 리전
  - `aws-us-gov` - AWS GovCloud (US) 리전

> **Note**
>
> 계정 별칭은 비밀이 아니며 공개 로그인 페이지 URL에 표시됩니다. 계정 별칭에 민감한 정보를 포함하지 마세요.
> AWS 계정 별칭을 만든 후에도 AWS 계정 ID를 포함하는 원래 URL은 활성 상태로 유지되며 사용할 수 있습니다.

계정 별칭 생성

다음 단계를 수행하려면 적어도 다음과 같은 IAM 권한이 있어야 합니다.

- `iam:ListAccountAliases`
- `iam:CreateAccountAlias`

AWS 계정 별칭 생성

계정 별칭 생성을 위해 따르려는 방법에 해당하는 탭 선택:

IAM console

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 대시보드를 선택합니다.
3. AWS 계정 섹션의 계정 별칭 옆에 있는 생성을 선택합니다. 별칭이 이미 있는 경우 편집(Edit)을 선택합니다.
4. 대화 상자에서 별칭에 사용할 이름을 입력한 후 변경사항 저장을 선택합니다.

AWS CLI

다음 명령 실행:

- `aws iam create-account-alias`

API

AWS Management Console 로그인 페이지 URL의 별칭을 만들려면 다음 연산을 호출합니다.

- CreateAccountAlias

계정 별칭 삭제

다음 단계를 수행하려면 적어도 다음과 같은 IAM 권한이 있어야 합니다.

- iam:ListAccountAliases
- iam:DeleteAccountAlias

계정 별칭 삭제

계정 별칭 삭제를 위해 따르려는 방법에 해당하는 탭 선택:

IAM console

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 대시보드를 선택합니다.
3. AWS 계정 섹션의 계정 별칭 옆에 있는 삭제를 선택합니다.

AWS CLI

AWS 계정 ID 별칭을 삭제하려면 다음 명령을 실행합니다.

- aws iam delete-account-alias

다음 명령을 실행하여 AWS 계정 ID 별칭을 표시를 시도하고 계정 별칭이 삭제되었는지 확인:

- aws iam list-account-aliases

API

AWS 계정 ID 별칭을 삭제하려면 다음 연산을 호출합니다.

- DeleteAccountAlias

다음 작업을 호출하여 AWS 계정 ID 별칭을 표시를 시도하고 계정 별칭이 삭제되었는지 확인:

- ListAccountAliases

> **Note**
>
> 계정 별칭을 삭제한 이후 계정의 유일한 로그인 URL은 계정 ID를 기반으로 합니다. 별칭 URL에 연결하려는 모든 시도는 실패하고 리디렉션되지 않습니다.

AWS 계정에 대한 액세스 계획

AWS 설정 시 사용자가 AWS 계정과 리소스에 어떻게 액세스할지 계획하여 잘 설계되고 안전한 ID 관리 솔루션을 설정하세요.

ID 원본

IAM 모범 사례에 따르면 사용자 및 워크로드는 AWS 리소스에 액세스할 때 임시 자격 증명을 사용해야 합니다. IAM 역할을 사용하여 리소스에 액세스하는 ID에는 임시 자격 증명이 부여됩니다. IAM으로 페더레이션된 사용자와 IAM Identity Center의 사용자(IAM Identity Center 디렉터리에 페더레이션 또는 생성된 사용자) 모두 IAM 역할을 사용하여 리소스에 액세스합니다.

AWS 사용을 시작하기 전에 다음을 통해 ID 설정 방법을 계획:

- Organizations에서 IAM Identity Center를 활성화하고 IAM Identity Center의 사용자를 조직 디렉터리에 직접 추가합니다.

  IAM Identity Center 조직 디렉터리에 사용자를 직접 추가하는 방법을 알아보려면 사용자 추가를 참조하세요.
- 기존 외부 ID 제공업체와 IAM Identity Center 또는 IAM과 페더레이션합니다.

  외부 ID 제공업체를 IAM Identity Center 조직 디렉터리에 페더레이션하는 방법을 알아보려면 해당 시작하기 자습서를 사용하세요.

액세스 관리

사용자가 액세스할 AWS 리소스와 서비스를 식별하고 각 사용자, 그룹 또는 역할에 필요한 액세스 권한 및 정책을 정의합니다.

- IAM Identity Center를 사용하는 경우 IAM ID 제공업체와 IAM 역할 및 권한 정책이 조직의 각 AWS 계정에 자동으로 생성됩니다. 이러한 역할 및 권한은 특정 애플리케이션 또는 AWS 계정에 사람이나 그룹을 할당할 때 지정하는 권한과 연계됩니다.

  자세한 내용은 사용자 액세스 할당 및 애플리케이션에 대한 Single Sign-On 액세스 설정을 참조하세요.
- ID 제공업체를 AWS 계정의 IAM에 직접 페더레이션하는 경우 사용자가 수임할 역할과 두 개의 정책, 즉 역할을 수임할 수 있는 사람을 지정하는 트러스트 정책과 역할을 수임하는 사람에게 액세스가 허용 또는 거부되는 AWS 작업 및 리소스를 지정하는 권한 정책을 만들어야 합니다.

  자세한 내용은 자격 증명 공급자 및 페더레이션 섹션을 참조하세요.

IAM 사용자 사용 사례

AWS 계정에서 생성한 IAM 사용자는 직접 관리하는 장기 자격 증명을 보유합니다.

AWS에서 액세스를 관리할 때는 일반적으로 IAM 사용자가 최선의 선택이 아닙니다. 대부분의 사용 사례에서 IAM 사용자에 의존하지 않아야 하는 몇 가지 중요한 이유가 있습니다.

먼저, IAM 사용자는 개별 계정용으로 설계되므로 조직의 규모가 커지더라도 규모 조정이 원활하지 않습니다. 다수의 IAM 사용자에 대한 권한 및 보안 관리가 순식간에 어려운 작업이 될 수 있습니다.

또한 IAM 사용자는 다른 AWS ID 관리 솔루션에서 제공하는 중앙 집중식 가시성 및 감사 기능이 부족합니다. 이로 인해 보안 및 규제 준수 유지가 더 까다로울 수 있습니다.

마지막으로 확장 가능한 ID 관리 접근 방식을 사용하면 다중 인증, 암호 정책, 역할 분리와 같은 보안 모범 사례를 훨씬 쉽게 구현할 수 있습니다.

IAM 사용자에 의존하는 대신 IAM Identity Center를 갖춘 Organizations 또는 외부 제공업체의 페더레이션 ID와 같은 보다 강력한 솔루션을 사용하는 것이 좋습니다. 이러한 옵션을 사용하면 AWS 환경의 확장에 따라 제어, 보안 및 운영 효율성 개선을 제공합니다.

결과적으로 페더레이션 사용자가 지원하지 않는 사용 사례에는 IAM 사용자만 사용하는 것이 좋습니다.

다음 목록은 AWS에서 IAM 사용자의 장기 보안 자격 증명이 필요한 특정 사용 사례를 보여줍니다. IAM을 사용하여 AWS 계정 내에서 이러한 IAM 사용자를 생성하고 IAM을 사용하여 해당 권한을 관리할 수 있습니다.

- AWS 계정에 대한 비상 액세스

- IAM 역할을 사용할 수 없는 워크로드
  - AWS CodeCommit 액세스
  - Amazon Keyspaces(Apache Cassandra용) 액세스
- 타사 AWS 클라이언트
- 사용자 계정에 AWS IAM Identity Center을 사용할 수 없으며 다른 ID 공급자가 없음

비상 액세스를 위한 IAM 사용자 생성

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 AWS 계정내 ID입니다.

비상 액세스를 위한 IAM 사용자를 두는 것은 ID 제공업체에 액세스할 수 없는 경우에도 AWS 계정에 액세스할 수 있도록 IAM 사용자를 생성하는 것이 권장되는 이유 중 하나입니다.

> **Note**
>
> 보안 모범 사례로 IAM 사용자를 생성하는 대신 ID 페더레이션을 통해 리소스에 대한 액세스를 제공하는 것이 좋습니다. IAM 사용자가 필요한 특정 상황에 대한 자세한 내용은 IAM 사용자(역할이 아님)를 생성해야 하는 경우를 참조하세요.

비상 액세스를 위한 IAM 사용자 생성

IAM 사용자 생성을 위해 따르려는 방법에 해당하는 탭 선택:

IAM console

1. AWS 로그인 사용 설명서의 AWS에 로그인하는 방법 항목에 설명된 대로 사용자 유형에 맞는 로그인 절차를 따르세요.
2. 콘솔 홈 페이지에서 IAM 서비스를 선택합니다.
3. 탐색 창에서 사용자와 사용자 추가를 차례로 선택합니다.

> **Note**
>
> IAM Identity Center를 활성화한 경우 IAM Identity Center에서 사용자 액세스를 관리하는 것이 가장 좋다는 알림이 AWS Management Console에 표시됩니다. 이 절차에서 생성한 IAM 사용자는 ID 제공업체를 사용할 수 없는 경우에만 사용하도록 설계되었습니다.

4. 사용자 이름에 **EmergencyAccess**를 입력합니다. 이름에는 공백이 있어서는 안 됩니다.
5. 사용자에게 AWS Management Console 액세스 권한 제공- 선택 사항 옆의 확인란을 선택한 다음 IAM 사용자를 생성하고 싶음을 선택합니다.
6. 콘솔 암호에서 자동 생성된 암호(권장)을 선택합니다.
7. 다음 로그인 시 사용자가 새 암호를 생성해야 함(권장) 옆의 확인란을 선택 해제합니다. 이 IAM 사용자는 긴급 액세스용이므로 신뢰할 수 있는 관리자가 암호를 보관하고 필요할 때만 암호를 제공합니다.
8. 권한 설정 페이지의 권한 옵션에서 그룹에 사용자 추가를 선택합니다. 그런 다음 사용자 그룹에서 그룹 생성을 선택합니다.
9. 사용자 그룹 생성 페이지의 사용자 그룹 이름에 **EmergencyAccessGroup**을 입력합니다. 그런 다음 권한 정책에서 AdministratorAccess를 선택합니다.
10. 권한 설정 페이지로 돌아가려면 사용자 그룹 생성을 선택합니다.
11. 사용자 그룹에서 이전에 만든 **EmergencyAccessGroup** 이름을 선택합니다.
12. 다음을 선택하여 검토 및 생성 페이지로 이동합니다.
13. 검토 및 생성 페이지에서 새 사용자에 추가될 사용자 그룹 멤버십의 목록을 검토합니다. 계속 진행할 준비가 되었으면 사용자 생성을 선택합니다.
14. 암호 검색 페이지에서 .csv 파일 다운로드를 선택하여 사용자 자격 증명 정보(연결 URL, 사용자 이름, 암호)가 포함된 .csv 파일을 저장합니다.
15. IAM에 로그인해야 하고 ID 제공업체에 액세스할 수 없는 경우 이 파일을 저장하여 사용하세요.

새 IAM 사용자가 사용자 목록에 표시됩니다. 사용자 세부 정보를 보려면 사용자 이름 링크를 선택합니다.

AWS CLI

1. **EmergencyAccess**라는 사용자를 생성합니다.

   - aws iam create-user

```
aws iam create-user \
--user-name EmergencyAccess
```

2. (선택 사항) 사용자에게 AWS Management Console에 대한 액세스 권한 부여. 이를 위해서는 암호가 필요합니다. IAM 사용자의 암호를 생성하기 위해 `--cli-input-json` 파라미터를 사

용하여 암호가 포함된 JSON 파일을 전달할 수 있습니다. 사용자에게 계정 로그인 페이지의 URL도 제공해야 합니다.

- aws iam create-login-profile

```
aws iam create-login-profile \
--generate-cli-skeleton > create-login-profile.json
```

- 텍스트 편집기에서 `create-login-profile.json` 파일을 열고 암호 정책을 준수하는 암호를 입력한 다음 파일을 저장합니다. 예:

```
{
 "UserName": "EmergencyAccess",
 "Password": "Ex@3dRA0djs",
 "PasswordResetRequired": false
}
```

- 다시 `aws iam create-login-profile` 명령을 사용하여 `--cli-input-json` 파라미터를 전달하고 JSON 파일을 지정합니다.

```
 aws iam create-login-profile \
                                --cli-input-json file://create-login-
profile.json
```

> **Note**
>
> JSON 파일에 제공한 암호가 계정의 암호 정책을 위반하는 경우 `PassworPolicyViolation` 오류가 발생합니다. 이 경우 계정의 암호 정책을 검토하고 요구 사항에 맞게 JSON 파일의 암호를 업데이트합니다.

3. **`EmergencyAccessGroup`**을 생성하고, AWS 관리형 정책 `AdministratorAccess`를 그룹에 연결하고, **`EmergencyAccess`** 사용자를 그룹에 추가합니다.

> **Note**
>
> AWS 관리형 정책은 AWS에서 생성 및 관리하는 독립적인 정책입니다. 각 정책에는 정책 이름이 포함된 Amazon 리소스 이름(ARN)이 있습니다. 예를 들어 `arn:aws:iam::aws:policy/IAMReadOnlyAccess`는 AWS 관리형 정책입니다. ARN에 대한 자세한 내용은 IAM ARN 섹션을 참조하세요. AWS에 대한 AWS 서비스 관리형 정책의 목록은 AWS 관리형 정책을 참조하세요.

- aws iam create-group

```
aws iam create-group \
--group-name EmergencyAccessGroup
```

- aws iam attach-group-policy

```
aws iam attach-group-policy \
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
--group-name >EmergencyAccessGroup
```

- aws iam add-user-to-group

```
aws iam add-user-to-group \
--user-name EmergencyAccess \
--group-name EmergencyAccessGroup
```

- aws iam get-group 명령을 실행하여 **EmergencyAccessGroup** 및 구성원을 나열합니다.

```
aws iam get-group \
--group-name EmergencyAccessGroup
```

IAM 역할을 사용할 수 없는 워크로드용 IAM 사용자 생성

> **Important**
>
> 모범 사례로, 인간 사용자가 AWS에 액세스할 때 임시 자격 증명을 사용하는 것을 권장합니다. 임시 보안 인증을 제공하는 역할을 가정하여 인간 사용자가 AWS 계정에 페더레이션 액세스를 제공하도록 ID 공급자를 사용할 수 있습니다. 중앙 액세스 관리를 위해 AWS IAM Identity Center(IAM Identity Center)를 사용하여 계정에 대한 액세스 권한과 해당 계정 내 권한을 관리하는 것이 좋습니다. IAM Identity Center를 사용하여 관리자 사용자를 포함한 사용자 ID를 생성하고 관리할 수 있습니다. 외부 ID 공급자를 사용하는 경우 IAM Identity Center에서 사용자 ID에 대한 액세스 권한을 구성할 수도 있습니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 AWS IAM Identity Center란 무엇입니까? 단원을 참조하세요.

프로그래밍 방식 액세스 및 장기 자격 증명이 있는 IAM 사용자가 필요한 사용 사례에서는 필요할 때 액세스 키를 업데이트하는 절차를 수립하는 것이 좋습니다. 자세한 내용은 액세스 키 업데이트 단원을 참조하십시오.

일부 계정 및 서비스 관리 작업을 수행하려면 루트 사용자 자격 증명을 사용하여 로그인해야 합니다. 루트 사용자로 로그인해야 하는 작업을 보려면 루트 사용자 보안 인증이 필요한 작업 항목을 참조하세요.

IAM 역할을 사용할 수 없는 워크로드용 IAM 사용자 생성

워크로드에 대한 IAM 사용자 생성을 위해 따르려는 방법에 해당하는 탭 선택:

IAM console

1. AWS 로그인 사용 설명서의 AWS에 로그인하는 방법 항목에 설명된 대로 사용자 유형에 맞는 로그인 절차를 따르세요.
2. 콘솔 홈 페이지에서 IAM 서비스를 선택합니다.
3. 탐색 창에서 사용자를 선택한 다음, 사용자 생성을 선택합니다.
4. 사용자 세부 정보 지정 페이지에서 다음 작업을 수행합니다.
   a. 사용자 이름에 ***WorkloadName***을 입력합니다. ***WorkloadName***을 계정을 사용할 워크로드의 이름으로 바꿉니다.
   b. Next(다음)를 선택합니다.
5. (선택 사항) 권한 설정 페이지에서 다음을 수행합니다.

a. 사용자를 그룹에 추가를 선택합니다.

b. 그룹 생성을 선택합니다.

c. 사용자 그룹 생성 대화 상자에서 사용자 그룹 이름에 그룹 내 워크로드 사용을 나타내는 이름을 입력합니다. 이 예에서는 **`Automation`** 이름을 사용합니다.

d. 권한 정책에서 PowerUserAccess 관리형 정책의 확인란을 선택합니다.

> **Tip**
>
> 권한 정책 검색 상자에 파워를 입력하면 관리형 정책을 빠르게 찾을 수 있습니다.

e. 사용자 그룹 만들기를 선택합니다.

f. 사용자 그룹 목록이 있는 페이지로 돌아가 새 사용자 그룹 옆의 확인란을 선택합니다. 목록에 새 사용자 그룹이 표시되지 않으면 새로 고침(Refresh)을 선택합니다.

g. Next(다음)를 선택합니다.

6. (선택 사항) 태그 섹션에서 태그를 키 값 페어로 연결하여 메타데이터를 사용자에게 추가합니다. 자세한 내용은 AWS Identity and Access Management 리소스용 태그 단원을 참조하십시오.

7. 새 사용자에 대한 사용자 그룹 멤버십을 확인합니다. 계속 진행할 준비가 되었으면 사용자 생성을 선택합니다.

8. 사용자가 성공적으로 생성되었다는 상태 알림이 표시됩니다. 사용자 세부 정보 페이지로 이동하려면 사용자 보기를 선택합니다.

9. 보안 자격 증명 탭을 선택합니다. 그런 다음 워크로드에 필요한 자격 증명을 생성합니다.

- 액세스 키-액세스 키 생성을 선택하여 사용자에 대한 액세스 키를 생성하고 다운로드합니다.

> **Important**
>
> 보안 액세스 키는 이 때만 확인 및 다운로드가 가능하기 때문에 사용자에게 AWS API를 사용하도록 하려면 이 정보를 제공해야 합니다. 사용자의 새 액세스 키 ID와 보안 액세스 키를 안전한 장소에 보관하세요. 이 단계 이후에는 보안 키에 다시 액세스할 수 없습니다.

- AWS CodeCommit에 대한 SSH 퍼블릭 키-SSH 퍼블릭 키 업로드를 선택하여 사용자가 SSH를 통해 CodeCommit 리포지토리와 통신할 수 있도록 SSH 퍼블릭 키를 업로드합니다.

- AWS CodeCommit에 대한 HTTPS Git 자격 증명-자격 증명 생성을 선택하여 Git 리포지토리와 함께 사용할 고유한 사용자 자격 증명 세트를 생성합니다. 사용자 이름과 암호를.csv 파일에 저장하려면 자격 증명 다운로드를 선택합니다. 이 시점에만 해당 정보를 사용할 수 있습니다. 암호를 잊어버리거나 분실한 경우 재설정해야 합니다.
- Amazon Keyspaces(Apache Cassandra용) 자격 증명-Amazon Keyspaces와 함께 사용할 서비스별 사용자 자격 증명을 생성하려면 자격 증명 생성을 선택합니다. 사용자 이름과 암호를 .csv 파일에 저장하려면 자격 증명 다운로드를 선택합니다. 이 시점에만 해당 정보를 사용할 수 있습니다. 암호를 잊어버리거나 분실한 경우 재설정해야 합니다.

  > **Important**
  >
  > 서비스별 자격 증명은 특정 IAM 사용자와 연결되는 장기 자격 증명이며, 해당 자격 증명을 생성한 서비스에만 사용할 수 있습니다. 임시 자격 증명을 사용하여 모든 AWS 리소스에 액세스할 수 있는 IAM 역할 또는 페더레이션 ID 권한을 부여하려면 Amazon Keyspaces용 SigV4 인증 플러그인을 통한 AWS 인증을 사용합니다. 자세한 내용은 Amazon Keyspaces(Apache Cassandra용) 개발자 안내서의 임시 자격 증명을 사용하여 IAM 역할 및 SigV4 플러그인을 통해 Amazon Keyspaces(Apache Cassandra용)에 연결을 참조하세요.

- X.509 서명 인증서-안전한 SOAP 프로토콜 요청을 해야 하고 AWS Certificate Manager에서 지원되지 않는 리전에 있는 경우 X.509 인증서 생성을 선택합니다. ACM은 서버 인증서를 프로비저닝, 관리 및 배포하기 위한 기본 도구입니다. ACM 사용에 대한 자세한 내용은 AWS Certificate Manager 사용 설명서를 참조하세요.

프로그래밍 방식으로 액세스할 수 있는 사용자를 생성하고 PowerUserAccess 작업 기능을 사용하여 구성했습니다. 이 사용자 권한 정책은 IAM 및 AWS Organizations를 제외한 모든 서비스에 대한 전체 액세스 권한을 부여합니다.

워크로드가 IAM 역할을 맡을 수 없는 경우 이 동일한 프로세스를 사용하여 추가 워크로드에 프로그래밍 방식으로 AWS 계정 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 이 절차에서는 PowerUserAccess 관리형 정책을 사용하여 권한을 할당했습니다. 최소 권한의 모범 사례를 따르려면, 보다 제한적인 정책을 사용하거나, 프로그램에 필요한 리소스만 액세스하도록 제한하는 사용자 지정 정책을 만드는 것이 좋습니다. 사용자 권한을 특정 AWS 리소스로 제한하는 정책을 사용하는 방법은 AWS리소스에 대한 액세스 관리 및 IAM 자격 증명 기반 정책의 예 단원을 참조하십시오. 사용자 그룹을 생성한 후 추가로 사용자를 추가하려면 IAM 사용자 그룹의 사용자 편집 섹션을 참조하세요.

AWS CLI

1. **Automation**라는 사용자를 생성합니다.

   - aws iam create-user

```
aws iam create-user \
                                --user-name Automation
```

2. **AutomationGroup**이라는 IAM 사용자 그룹을 생성하고, AWS 관리형 정책 PowerUserAccess를 그룹에 연결한 다음 **Automation** 사용자를 그룹에 추가합니다.

> **Note**
>
> AWS 관리형 정책은 AWS에서 생성 및 관리하는 독립적인 정책입니다. 각 정책에는 정책 이름이 포함된 Amazon 리소스 이름(ARN)이 있습니다. 예를 들어 arn:aws:iam::aws:policy/IAMReadOnlyAccess는 AWS 관리형 정책입니다. ARN에 대한 자세한 내용은 IAM ARN 섹션을 참조하세요. AWS에 대한 AWS 서비스 관리형 정책의 목록은 AWS 관리형 정책을 참조하세요.

   - aws iam create-group

```
aws iam create-group \
    --group-name AutomationGroup
```

   - aws iam attach-group-policy

```
aws iam attach-group-policy \
    --policy-arn arn:aws:iam::aws:policy/PowerUserAccess \
    --group-name AutomationGroup
```

   - aws iam add-user-to-group

```
aws iam add-user-to-group \
    --user-name Automation \
    --group-name AutomationGroup
```

- aws iam get-group 명령을 실행하여 **AutomationGroup** 및 구성원을 나열합니다.

```
aws iam get-group \
     --group-name AutomationGroup
```

3. 워크로드에 필요한 봔 자격 증명을 생성합니다.

- 테스트용 액세스 키 생성 - aws iam create-access-key

```
aws iam create-access-key \
    --user-name Automation
```

이 명령의 출력에는 비밀 액세스 키와 액세스 키 ID가 표시됩니다. 안전한 위치에 이 정보를 기록하고 저장합니다. 이러한 자격 증명을 손실하면 복구할 수 없으며, 새 액세스 키를 생성해야 합니다.

> **Important**
>
> 이러한 IAM 사용자 액세스 키는 계정에 보안 위험을 초래하는 장기 자격 증명입니다. 테스트를 완료한 후에는 이러한 액세스 키를 삭제하는 것이 좋습니다. 액세스 키를 고려하고 있는 경우 워크로드 IAM 사용자에 대해 MFA를 활성화할 수 있는지 여부를 조사하고 IAM 액세스 키를 사용하는 대신 aws sts get-session-token을 사용하여 세션에 대한 임시 자격 증명을 획득하세요.

- AWS CodeCommit에 대한 SSH 퍼블릭 키 업로드 – aws iam upload-ssh-public-key

다음 예제에서는 SSH 퍼블릭 키가 `sshkey.pub` 파일에 저장되어 있다고 가정합니다.

```
aws upload-ssh-public-key \
    --user-name Automation \
    --ssh-public-key-body file://sshkey.pub
```

- X.509 서명 인증서 업로드 – aws iam upload-signing-certificate

AWS Certificate Manager에서 지원되지 않는 리전에 있는 경우 보안 SOAP 프로토콜 요청을 해야 한다면 X.509 인증서를 업로드하세요. ACM은 서버 인증서를 프로비저닝, 관리 및 배포하기 위한 기본 도구입니다. ACM 사용에 대한 자세한 내용은 AWS Certificate Manager 사용 설명서를 참조하세요.

다음 예제에서는 X.509 서명 인증서가 `certificate.pem` 파일에 저장되어 있다고 가정합니다.

```
aws iam upload-signing-certificate \
--user-name Automation \
--certificate-body file://certificate.pem
```

워크로드가 IAM 역할을 맡을 수 없는 경우 이 동일한 프로세스를 사용하여 추가 워크로드에 프로그래밍 방식으로 AWS 계정 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 이 절차에서는 PowerUserAccess 관리형 정책을 사용하여 권한을 할당했습니다. 최소 권한의 모범 사례를 따르려면, 보다 제한적인 정책을 사용하거나, 프로그램에 필요한 리소스만 액세스하도록 제한하는 사용자 지정 정책을 만드는 것이 좋습니다. 사용자 권한을 특정 AWS 리소스로 제한하는 정책을 사용하는 방법은 AWS리소스에 대한 액세스 관리 및 IAM 자격 증명 기반 정책의 예 단원을 참조하십시오. 사용자 그룹을 생성한 후 추가로 사용자를 추가하려면 IAM 사용자 그룹의 사용자 편집 섹션을 참조하세요.

ID에 다중 인증 추가

ID에 대한 다중 인증(MFA)을 추가하는 것 역시 또 다른 권장 사항입니다. MFA는 사용자가 ID를 확인하기 위해 사용자 이름과 암호를 입력한 후 추가 인증 요소를 제공해야 하는 추가 보안 계층입니다. 사용자의 암호가 침해되더라도 공격자의 무단 액세스를 훨씬 더 어렵게 만들어 보안을 크게 강화합니다. MFA는 온라인 계정, 클라우드 서비스 및 기타 민감한 리소스에 대한 액세스를 보호하는 모범 사례로 널리 채택되고 있습니다. AWS는 루트 사용자, IAM 사용자, IAM Identity Center의 사용자, Builder ID, 페더레이션 사용자에 대해 MFA를 지원합니다. 보안을 강화하기 위해 사용자의 리소스 액세스 또는 특정 조치를 허용하기 전에 MFA를 요구하는 정책을 생성하고 이를 IAM 역할에 연결할 수 있습니다.

자세한 내용은 IAM Identity Center에서 MFA 활성화 및 IAM의 AWS 다중 인증의 내용을 참조하세요.

최소 권한 준비

최소 권한을 사용하는 것이 IAM 모범 권장 사항입니다. 최소 권한이라는 개념은 작업을 수행하는 데 필요한 권한만 부여하고 다른 추가 권한을 없는 것입니다. 설정을 완료할 때 최소 권한을 어떻게 지원할 것인지 고려하세요. 루트 사용자, 관리 사용자 및 비상 액세스 IAM 사용자에게는 일상적인 작업에 필요하지 않은 강력한 권한이 주어집니다. AWS에 대해 알아보고 다양한 서비스를 테스트하는 동안 여러 가지 시나리오에서 사용할 수 있는 권한이 적은 추가 사용자를 IAM Identity Center에 한 명 이상 생성하는 것이 좋습니다. IAM 정책을 사용하여 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의한 다음 이러한 리소스와 최소 권한 계정을 연결합니다.

IAM Identity Center를 사용하는 경우 IAM Identity Center 권한 세트를 사용하여 시작하는 것을 고려해 보세요. 자세한 내용은 IAM Identity Center 사용 설명서의 권한 세트 생성을 참조하세요.

IAM Identity Center를 사용하지 않는 경우 IAM 역할을 사용하여 다양한 IAM 엔터티에 대한 권한을 정의합니다. 자세한 내용은 IAM 역할 생성을 참조하십시오.

IAM 역할과 IAM Identity Center 권한 세트 모두 직무에 따른 AWS 관리형 정책을 사용할 수 있습니다. 이러한 정책에서 부여하는 권한에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 섹션을 참조하세요.

> **Important**
>
> AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있기 때문에 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 설정을 마친 후에는 IAM Access Analyzer를 사용하여 AWS CloudTrail에 로그인한 액세스 활동을 기반으로 최소 권한을 생성하는 것이 좋습니다. 정책 생성에 대한 자세한 내용은 IAM Access Analyzer 정책 생성을 참조하세요.

시작할 때는 AWS 관리형 정책을 사용하여 권한을 부여하는 것이 좋습니다. 사전 정의된 활성 샘플 기간(예: 90일)이 지난 후에는 사용자와 워크로드에서 액세스한 서비스를 검토할 수 있습니다. 그런 다음 권한이 축소된 새 고객 관리형 정책을 생성하여 AWS 관리형 정책을 대체할 수 있습니다. 새 정책에는 샘플 기간 동안 액세스한 서비스만 포함되어야 합니다. 권한을 업데이트하여 AWS 관리형 정책을 제거하고 생성한 새 고객 관리형 정책을 연결합니다.

AWS 계정에 대해 마지막으로 액세스한 정보 검토

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 IAM에 대해 마지막으로 액세스한 서비스 정보를 볼 수 있습니다. 데이터, 필요 권한, 문제 해결, 지원되는 리전에 대한 중요 정보는 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 섹션을 참조하세요.

IAM에서 다음 리소스 유형에 대한 정보를 볼 수 있습니다. 각각의 경우 정보에는 지정된 보고 기간 동안 허용된 서비스가 포함됩니다.

- IAM 사용자 - 사용자가 허용된 각 서비스에 액세스하려고 시도한 마지막 시간을 표시합니다.
- IAM 그룹 - IAM 그룹 구성원이 허용된 각 서비스에 액세스하려고 시도한 마지막 시간에 대한 정보를 표시합니다. 또한 이 보고서에는 액세스를 시도한 총 멤버 수가 포함됩니다.
- IAM 역할 - 해당 역할이 허용된 각 서비스에 액세스하려고 시도한 마지막 시간을 표시합니다.
- 정책 - 사용자 또는 역할이 허용된 각 서비스에 액세스하려고 시도한 마지막 시간에 대한 정보를 표시합니다. 또한 이 보고서에는 액세스를 시도한 총 엔터티 수가 포함됩니다.

> **Note**
>
> IAM의 리소스에 대한 액세스 정보를 보려면 먼저 보고 기간, 보고된 엔터티 및 정보에 대해 평가된 정책 유형을 이해해야 합니다. 자세한 내용은 the section called “마지막으로 액세스한 정보에 대해 알아야 할 사항”를 참조하세요.

마지막으로 액세스한 정보에 대한 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 섹션을 참조하세요.

AWS 계정에 대해 마지막으로 액세스한 정보 검토

마지막으로 액세스한 정보를 검토하기 위해 따르려는 방법에 해당하는 탭 선택:

IAM console

1. AWS 로그인 사용 설명서의 AWS에 로그인하는 방법 항목에 설명된 대로 사용자 유형에 맞는 로그인 절차를 따르세요.
2. 콘솔 홈 페이지에서 IAM 서비스를 선택합니다.
3. 탐색 창에서 사용자 그룹, 사용자, 역할 또는 정책을 선택합니다.
4. 사용자, 사용자 그룹, 역할 또는 정책 이름을 선택하여 요약 페이지를 열고 액세스 관리자(Access Advisor) 탭을 선택합니다. 선택한 리소스를 기반으로 다음 정보를 확인합니다.

    - 사용자 그룹 - 사용자 그룹 멤버가 액세스할 수 있는 서비스 목록을 봅니다. 또한 멤버가 서비스에 마지막으로 액세스한 시기, 사용한 사용자 그룹 정책 및 요청한 사용자 그룹 멤버를 볼 수 있습니다. 정책의 이름을 선택하여 정책이 관리형 정책인지 아니면 인라인 사용자 그룹

정책인지 확인합니다. 사용자 그룹 멤버의 이름을 선택하여 사용자 그룹의 모든 멤버를 확인하고 마지막으로 서비스에 액세스한 시간을 확인합니다.

- 사용자 - 사용자가 액세스할 수 있는 서비스 목록을 봅니다. 또한 서비스에 마지막으로 액세스한 시기 및 사용한 정책을 볼 수도 있습니다. 정책의 이름을 선택하여 정책이 관리형 정책인지, 인라인 사용자 정책인지, 아니면 사용자 그룹의 인라인 정책인지 확인합니다.
- 역할 - 역할이 액세스할 수 있는 서비스 목록, 서비스에 마지막으로 액세스한 역할 및 사용된 정책 목록을 표시합니다. 정책의 이름을 선택하여 정책이 관리형 정책인지 아니면 인라인 역할 정책인지 확인합니다.
- 정책 - 정책에서 허용된 작업이 있는 서비스 목록을 봅니다. 서비스에 액세스하는 데 정책이 마지막으로 사용된 시기와 해당 정책을 사용한 엔터티(사용자 또는 역할)도 볼 수 있습니다. 그 마지막 액세스 날짜에는 다른 정책을 통해 이 정책에 대한 액세스 권한이 부여된 시기도 포함됩니다. 엔터티의 이름을 선택하여 어떤 엔터티에 이 정책이 연결되어 있는지 그리고 마지막으로 서비스에 액세스한 시간을 확인합니다.

5. 테이블의 서비스 열을 선택하고 마지막으로 액세스한 작업 정보가 포함된 서비스 중 하나의 이름을 선택하여 IAM 엔터티가 액세스하려고 시도한 관리 작업 목록을 확인합니다. 사용자가 마지막으로 작업을 수행하려고 시도한 시점을 보여주는 AWS 리전 및 타임스탬프를 볼 수 있습니다.
6. 마지막으로 액세스한 작업 정보가 포함된 서비스의 서비스 및 관리 작업에 대해 마지막 액세스 열이 표시됩니다. 이 열에 반환되는 다음과 같은 가능한 결과를 검토합니다. 이러한 결과는 서비스나 작업이 허용되는지, 액세스되는지, 마지막으로 액세스한 정보에 대해 AWS가 추적하는지 여부에 따라 달라집니다.

   <number of days>일 전

   추적 기간에 서비스 또는 작업이 사용된 이후의 일 수입니다. 서비스에 대한 추적 기간은 지난 400일입니다. Amazon S3 작업에 대한 추적 기간은 2020년 4월 12일에 시작되었습니다. Amazon EC2, IAM 및 Lambda 작업에 대한 추적 기간은 2021년 4월 7일에 시작되었습니다. 다른 모든 서비스의 추적 기간은 2023년 5월 23일에 시작되었습니다. 각 AWS 리전의 추적 시작 날짜에 대한 자세한 내용은 AWS에서 마지막으로 액세스한 정보를 추적하는 위치 섹션을 참조하세요.

   추적 기간에 액세스하지 않음

   추적된 서비스 또는 작업은 추적 기간 동안 엔터티에서 사용되지 않았습니다.

목록에 나타나지 않는 작업에 대한 권한이 있을 수 있습니다. 작업에 대한 추적 정보가 현재 AWS에 포함되지 않는 경우 이 문제가 발생할 수 있습니다. 추적 정보가 없는 경우 권한을 결정

해서는 안 됩니다. 대신 이 정보를 사용하여 최소 권한 부여에 대한 전반적인 전략을 알리고 지원하는 것이 좋습니다. 정책을 확인하여 액세스 수준이 적절한지 확인합니다.

AWS CLI

AWS CLI를 사용하여 AWS 서비스 및 Amazon S3, Amazon EC2, IAM 및 Lambda 작업에 액세스하기 위해 AWS 계정의 IAM 리소스가 사용된 마지막 시간에 대한 정보를 검색할 수 있습니다. IAM 리소스는 사용자, 사용자 그룹, 역할 또는 정책입니다.

- AWS 계정에서 IAM 리소스에 대한 보고서를 생성합니다. 요청에는 보고서가 필요한 IAM 리소스(사용자, 사용자 그룹, 역할 또는 정책)의 ARN이 포함되어야 합니다. 보고서에 생성할 세부 수준을 지정하여 서비스 또는 서비스 및 작업 모두에 대한 액세스 세부 정보를 볼 수 있습니다. 이 요청은 작업이 완료될 때까지 `get-service-last-accessed-details` 및 `get-service-last-accessed-details-with-entities` 작업에서 `job-status`를 모니터링하기 위해 사용할 수 있는 `job-id`를 반환합니다.

  - aws iam generate-service-last-accessed-details

  a. 이전 단계의 `job-id` 파라미터를 사용하여 보고서에 대한 세부 정보를 검색합니다.

     - aws iam get-service-last-accessed-details

     이 작업은 `generate-service-last-accessed-details` 작업에서 요청한 리소스 유형 및 세부 수준에 따라 다음 정보를 반환합니다.

     - 사용자 - 지정한 사용자가 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 사용자의 마지막 시도 날짜 및 시간과 사용자의 ARN을 반환합니다.
     - 사용자 그룹 - 사용자 그룹에 연결된 정책을 사용하여 지정된 사용자 그룹의 멤버가 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 사용자 그룹 멤버가 마지막으로 시도한 날짜와 시간을 반환합니다. 또한 해당 사용자의 ARN과 서비스에 액세스하려고 시도한 사용자 그룹 멤버의 총 수를 반환합니다. 모든 멤버 목록을 반환하려면 GetServiceLastAccessedDetailsWithEntities 작업을 사용합니다.
     - 역할 - 지정한 역할이 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 역할의 마지막 시도 날짜 및 시간과 역할의 ARN을 반환합니다.
     - 정책 - 지정된 정책으로 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 엔터티(사용자 또는 역할)가 정책을 사용하여 마지막으로 서비스에 액세스하려

고 시도한 날짜와 시간을 반환합니다. 또한 엔터티의 ARN과 액세스를 시도한 엔터티의 총 수를 반환합니다.

b. 특정 서비스에 액세스하기 위해 사용자 그룹 또는 정책 권한을 사용하는 엔터티에 대해 자세히 알아봅니다. 이 작업은 각 엔터티의 ARN, ID, 이름, 경로, 유형(사용자 또는 역할) 및 마지막으로 서비스에 액세스하려고 시도한 엔터티의 목록을 반환합니다. 사용자와 역할에 대해 이 작업을 사용할 수도 있지만 해당 엔터티에 대한 정보만 반환합니다.

   - aws iam get-service-last-accessed-details-with-entities

c. 특정 서비스에 액세스하기 위해 자격 증명(사용자, 사용자 그룹 또는 역할)이 사용하는 자격 기반 정책에 대해 자세히 알아봅니다. 자격 증명 및 서비스를 지정한 경우 이 작업은 해당 자격 증명이 지정된 서비스에 액세스하는 데 사용할 수 있는 권한 정책 목록을 반환합니다. 이 작업은 정책의 현재 상태를 제공하며 생성된 보고서에 의존하지 않습니다. 또한 리소스 기반 정책, 액세스 제어 목록, AWS Organizations 정책, IAM 권한 경계 또는 세션 정책 등의 다른 정책 유형을 반환하지 않습니다. 자세한 내용은 정책 유형 또는 단일 계정 내에서 정책 평가을 참조하세요.

   - aws iam list-policies-granting-service-access

API

AWS를 사용하여 AWS API 서비스 및 Amazon S3, Amazon EC2, IAM 및 Lambda 작업에 액세스하기 위해 IAM 리소스가 사용된 마지막 시간에 대한 정보를 검색할 수 있습니다. IAM 리소스는 사용자, 사용자 그룹, 역할 또는 정책입니다. 보고서에 생성할 세부 수준을 지정하여 서비스 또는 서비스 및 작업 모두에 대한 세부 정보를 볼 수 있습니다.

1. 보고서를 생성합니다. 요청에는 보고서가 필요한 IAM 리소스(사용자, 사용자 그룹, 역할 또는 정책)의 ARN이 포함되어야 합니다. 작업이 완료될 때까지 `GetServiceLastAccessedDetails` 및 `GetServiceLastAccessedDetailsWithEntities` 작업에서 `JobStatus`를 모니터링하기 위해 사용할 수 있는 `JobId`를 반환합니다.

   - GenerateServiceLastAccessedDetails

2. 이전 단계의 `JobId` 파라미터를 사용하여 보고서에 대한 세부 정보를 검색합니다.

   - GetServiceLastAccessedDetails

이 작업은 `GenerateServiceLastAccessedDetails` 작업에서 요청한 리소스 유형 및 세부 수준에 따라 다음 정보를 반환합니다.

- 사용자 - 지정한 사용자가 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 사용자의 마지막 시도 날짜 및 시간과 사용자의 ARN을 반환합니다.
- 사용자 그룹 - 사용자 그룹에 연결된 정책을 사용하여 지정된 사용자 그룹의 멤버가 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 사용자 그룹 멤버가 마지막으로 시도한 날짜와 시간을 반환합니다. 또한 해당 사용자의 ARN과 서비스에 액세스하려고 시도한 사용자 그룹 멤버의 총 수를 반환합니다. 모든 멤버 목록을 반환하려면 GetServiceLastAccessedDetailsWithEntities 작업을 사용합니다.
- 역할 - 지정한 역할이 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 역할의 마지막 시도 날짜 및 시간과 역할의 ARN을 반환합니다.
- 정책 - 지정된 정책으로 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 엔터티(사용자 또는 역할)가 정책을 사용하여 마지막으로 서비스에 액세스하려고 시도한 날짜와 시간을 반환합니다. 또한 엔터티의 ARN과 액세스를 시도한 엔터티의 총 수를 반환합니다.

3. 특정 서비스에 액세스하기 위해 사용자 그룹 또는 정책 권한을 사용하는 엔터티에 대해 자세히 알아봅니다. 이 작업은 각 엔터티의 ARN, ID, 이름, 경로, 유형(사용자 또는 역할) 및 마지막으로 서비스에 액세스하려고 시도한 엔터티의 목록을 반환합니다. 사용자와 역할에 대해 이 작업을 사용할 수도 있지만 해당 엔터티에 대한 정보만 반환합니다.

   - GetServiceLastAccessedDetailsWithEntities

4. 특정 서비스에 액세스하기 위해 자격 증명(사용자, 사용자 그룹 또는 역할)이 사용하는 자격 기반 정책에 대해 자세히 알아봅니다. 자격 증명 및 서비스를 지정한 경우 이 작업은 해당 자격 증명이 지정된 서비스에 액세스하는 데 사용할 수 있는 권한 정책 목록을 반환합니다. 이 작업은 정책의 현재 상태를 제공하며 생성된 보고서에 의존하지 않습니다. 또한 리소스 기반 정책, 액세스 제어 목록, AWS Organizations 정책, IAM 권한 경계 또는 세션 정책 등의 다른 정책 유형을 반환하지 않습니다. 자세한 내용은 정책 유형 또는 단일 계정 내에서 정책 평가을 참조하세요.

   - ListPoliciesGrantingServiceAccess

액세스 활동을 기반으로 정책 생성

IAM 사용자 또는 IAM 역할에 대해 AWS CloudTrail에 기록된 액세스 활동을 사용하여 IAM 액세스 분석기에서 특정 사용자 및 역할에 필요한 서비스에만 액세스할 수 있도록 허용하는 고객 관리형 정책을 생성하도록 할 수 있습니다.

IAM Access Analyzer에서 IAM 정책을 생성하면 정책을 추가로 사용자 지정하는 데 도움이 되는 정보가 반환됩니다. 정책이 생성되면 다음 두 가지 범주의 정보가 반환될 수 있습니다.

- 작업 수준 정보가 포함된 정책 - Amazon EC2와 같은 일부 AWS 서비스의 경우 IAM Access Analyzer는 CloudTrail 이벤트에서 발견된 작업을 식별하고 생성된 정책에 사용된 작업을 나열할 수 있습니다. 지원되는 서비스의 목록은 IAM Access Analyzer 정책 생성 서비스 단원을 참조하세요. 일부 서비스의 경우 생성된 정책에 서비스에 대한 작업을 추가하라는 IAM Access Analyzer 메시지가 표시됩니다.
- 서비스 수준 정보 –이(가) 포함된 정책 IAM Access Analyzer는 마지막으로 액세스한 정보를 사용하여 최근에 사용한 모든 서비스가 포함된 정책 템플릿을 생성합니다. AWS Management Console을(를) 사용할 때 서비스를 검토하고 정책을 완료하기 위한 작업을 추가하라는 메시지가 표시됩니다.

액세스 활동을 기반으로 정책 생성

다음 절차에서는 사용자의 사용에 맞게 역할에 부여된 권한을 줄입니다. 사용자를 선택할 때는 사용이 해당 역할의 예제로 적합한 사용자를 선택하세요. 많은 고객이 PowerUser 권한으로 테스트 사용자 계정을 설정한 다음 짧은 기간 동안 특정 작업을 수행하도록 하여 해당 작업을 수행하는 데 필요한 액세스 권한을 결정합니다.

새 권한 정책을 생성할 때 따르려는 방법에 해당하는 탭 선택:

IAM console

1. AWS 로그인 사용 설명서의 AWS에 로그인하는 방법 항목에 설명된 대로 사용자 유형에 맞는 로그인 절차를 따르세요.
2. 콘솔 홈 페이지에서 IAM 서비스를 선택합니다.
3. 탐색 창에서 사용자를 선택한 다음 사용자 이름을 선택하여 사용자 세부 정보 페이지로 이동합니다.
4. 권한 탭의 CloudTrail 이벤트 기반 정책 생성에서 정책 생성을 선택합니다.
5. 정책 생성 페이지에서 다음 항목을 구성합니다.

    - 기간 선택에서 지난 7일을 선택합니다.

- 분석할 CloudTrail 트레일의 경우 이 사용자의 활동이 기록되는 리전 및 트레일을 선택합니다.
- 새 서비스 역할 생성 및 사용을 선택합니다.

6. 정책 생성을 선택한 다음 역할이 생성될 때까지 기다립니다. 정책 생성 진행 중 알림 메시지가 나타날 때까지 콘솔 페이지를 새로 고치거나 다른 페이지로 이동하지 마세요.
7. 정책이 생성된 후 필요에 따라 정책을 검토하고 리소스의 계정 ID 및 ARN을 사용하여 이를 사용자 지정해야 합니다. 또한 자동으로 생성된 정책에는 정책을 완료하는 데 필요한 작업 수준 정보가 포함되지 않았을 수 있습니다. 자세한 내용은 IAM Access Analyzer 정책 생성을 참조하세요.

   예를 들어, `Allow` 효과와 `NotAction` 요소를 포함하는 첫 번째 명령문을 편집하여 Amazon EC2 및 Amazon S3 작업만 허용할 수 있습니다. 이 작업을 수행하려면 `FullAccessToSomeServices` ID가 있는 명령문으로 바꿉니다. 새 정책은 다음 예제 정책과 유사할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Sid": "FullAccessToSomeServices",
          "Effect": "Allow",
          "Action": [
              "ec2:*",
              "s3:*"
          ],
          "Resource": "*"
      },
      {
          "Effect": "Allow",
          "Action": [
              "iam:CreateServiceLinkedRole",
              "iam:DeleteServiceLinkedRole",
              "iam:ListRoles",
              "organizations:DescribeOrganization"
          ],
          "Resource": "*"
      }
  ]
}
```

8. 최소 권한 부여의 모범 사례를 지원하기 위해 정책 검증에서 반환된 오류, 경고 또는 제안을 검토하고 수정합니다.
9. 특정 작업 및 리소스에 대한 정책 권한을 추가로 줄이려면 CloudTrail 이벤트 기록(Event history)에서 이벤트를 확인합니다. 여기에서 사용자가 액세스한 특정 작업 및 리소스에 대한 자세한 정보를 볼 수 있습니다. 자세한 내용은 AWS CloudTrail사용 설명서의 CloudTrail 콘솔에서 CloudTrail 이벤트 보기를 참조하세요.
10. 정책을 검토 및 검증한 후에는 정책을 설명할 수 있는 이름으로 저장하세요.
11. 역할 페이지로 이동하여 새 정책에서 허용하는 작업을 수행할 때 사용자가 수임할 역할을 선택합니다.
12. 권한 탭을 선택한 다음 권한 추가, 정책 연결을 선택합니다.
13. 권한 정책 연결 페이지의 기타 권한 정책 목록에서 만들어 둔 정책을 선택하고 정책 연결을 선택합니다.
14. 역할 세부 정보 페이지로 돌아갑니다. 이제 역할에 앞선 AWS 관리형 정책(예: PowerUserAccess)과 새 정책을 추가했습니다. AWS 관리형 정책의 확인란을 선택한 후 제거를 선택합니다. 제거를 확인하라는 메시지가 표시되면 제거를 선택합니다.

이제 새로 만든 정책에 따라 IAM 사용자, 페더레이션 사용자, 워크로드에 대한 액세스 권한이 축소됩니다.

AWS CLI

다음 명령을 사용하여 AWS CLI을(를) 사용하는 정책을 생성할 수 있습니다.

정책 생성

- aws accessanalyzer start-policy-generation

생성된 정책 보기

- aws accessanalyzer get-generated-policy

정책 생성 요청 취소

- aws accessanalyzer cancel-policy-generation

정책 생성 요청 목록 보기

- aws accessanalyzer list-policy-generations

API

다음 작업을 사용하여 AWS API를 사용하는 정책을 생성할 수 있습니다.

정책 생성

- StartPolicyGeneration

생성된 정책 보기

- GetGeneratedPolicy

정책 생성 요청 취소

- CancelPolicyGeneration

정책 생성 요청 목록 보기

- ListPolicyGenerations

검색을 사용하여 IAM 리소스 찾기

액세스 조사 결과를 살펴보면서 IAM 콘솔 검색 페이지를 사용하여 더 빠르게 IAM 리소스를 찾을 수 있습니다. 부분 리소스 이름 또는 ARN을 사용하여 리소스를 검색할 수 있습니다.

IAM console

IAM 콘솔 검색 기능으로 찾을 수 있는 항목은 다음과 같습니다.

- 검색 키워드(예: 사용자, 그룹, 역할, 자격 증명 공급자, 정책)와 일치하는 IAM 엔터티 이름
- 검색 키워드와 일치하는 작업

IAM 콘솔 검색 기능은 IAM Access Analyzer에 대한 정보를 반환하지 않습니다.

검색 결과의 각 행은 활성 링크입니다. 예를 들어 검색 결과에서 사용자 이름을 선택할 수 있습니다. 그러면 사용자 세부 정보 페이지로 이동합니다. 또는 예를 들어 사용자 만들기 활성 링크를 선택하여 사용자 생성 페이지로 이동할 수 있습니다.

> **Note**
>
> 액세스 키 검색에서는 검색 상자에 전체 액세스 키 ID를 입력해야 합니다. 검색 결과는 해당 키와 연결된 사용자를 보여줍니다. 여기에서 해당 사용자의 액세스 키를 관리할 수 있는 사용자 페이지로 이동할 수 있습니다.

IAM 콘솔의 검색 페이지를 사용하여 해당 계정과 관련된 항목을 찾습니다.

IAM 콘솔에서 항목을 검색하는 방법

1. AWS 로그인 사용 설명서의 AWS에 로그인하는 방법 항목에 설명된 대로 사용자 유형에 맞는 로그인 절차를 따르세요.
2. 콘솔 홈 페이지에서 IAM 서비스를 선택합니다.
3. 탐색 창에서 검색을 선택합니다.
4. 검색 상자에 검색 키워드를 입력합니다.
5. 검색 결과 목록에서 링크를 선택하여 콘솔의 해당 부분으로 이동합니다.

다음 아이콘은 검색으로 찾을 수 있는 항목의 유형을 식별합니다.

아이콘	설명
	IAM 사용자
	IAM 그룹
	IAM 역할
	IAM 정책

아이콘	설명
	"사용자 만들기" 또는 "정책 연결"과 같은 작업
	키워드 **delete**의 결과

샘플 검색 문구

IAM 검색 시 다음과 같은 문구를 사용할 수 있습니다. 기울임꼴로 표시된 용어를 찾으려는 실제 IAM 사용자, 그룹, 역할, 액세스 키, 정책 또는 ID 제공업체의 이름으로 대체합니다.

- ***user_name*** 또는 ***group_name*** 또는 ***role_name*** 또는 ***policy_name*** 또는 ***identity_provider_name***
- ***access_key***
- **add user *user_name* to groups** 또는 **add users to group *group_name***
- **remove user *user_name* from groups**
- **delete *user_name*** 또는 **delete *group_name*** 또는 **delete *role_name*** 또는 **delete *policy_name*** 또는 **delete *identity_provider_name***
- **manage access keys *user_name***
- **manage signing certificates *user_name***
- **users**
- **manage MFA for *user_name***
- **manage password for *user_name***
- **create role**
- **password policy**
- **edit trust policy for role *role_name***
- **show policy document for role *role_name***
- **attach policy to *role_name***
- **create managed policy**
- **create user**
- **create group**

- **attach policy to *group_name***
- **attach entities to *policy_name***
- **detach entities from *policy_name***

AWS Identity and Access Management의 보안 모범 사례 및 사용 사례

AWS Identity and Access Management(IAM)은 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용해 주세요.

IAM의 이점을 극대화하기 위해 권장 모범 사례에 대해 알아보시기 바랍니다. 이를 위한 한 가지 방법은 실제 시나리오에서 다른 AWS 서비스와 함께 IAM을 어떻게 사용하는지 알아보는 것입니다.

주제

IAM의 보안 모범 사례

Follow us on Twitter

> **Important**
>
> AWS Identity and Access Management 모범 사례는 2022년 7월 14일에 업데이트되었습니다.

AWS 리소스를 보호하려면 AWS Identity and Access Management(IAM)에 대한 다음 모범 사례를 따르세요.

주제

- 최소 권한 적용
- AWS 관리형 정책으로 시작하고 최소 권한을 향해 나아갑니다.
- IAM 액세스 분석기를 사용하여 액세스 활동을 기반으로 최소 권한 정책 생성
- 사용하지 않는 사용자, 역할, 권한, 정책 및 보안 인증은 정기적으로 검토하고 제거합니다.
- IAM 정책의 조건을 사용하여 액세스 추가 제한
- IAM Access Analyzer를 사용하여 리소스에 대한 퍼블릭 및 크로스 계정 액세스 확인
- IAM Access Analyzer를 사용하여 IAM 정책을 검증하여 안전하고 기능적인 권한을 보장합니다.
- 여러 계정에 권한 가드레일 설정
- 권한 경계를 사용하여 계정 내에서 권한 관리 위임

임시 보안 인증을 사용하여 AWS에 액세스하려면 인간 사용자가 ID 공급자와의 페더레이션을 사용하도록 요구합니다.

인간 사용자(인간 ID라고도 함)는 애플리케이션의 사용자, 관리자, 개발자, 운영자 및 소비자입니다. AWS 환경 및 애플리케이션에 액세스하려면 ID가 있어야 합니다. 조직의 구성원인 인간 사용자는 작업 인력 ID라고도 합니다. 인간 사용자는 여러분과 협업하고 AWS 리소스와 상호 작용하는 외부 사용자일 수도 있습니다. 인간 사용자는 웹 브라우저, 클라이언트 애플리케이션, 모바일 앱 또는 대화형 명령줄 도구를 통해 이렇게 할 수 있습니다.

AWS에 액세스할 때 인간 사용자에게 임시 보안 인증을 사용하도록 요구합니다. 임시 보안 인증을 제공하는 역할을 가정하여 인간 사용자가 AWS 계정에 페더레이션 액세스를 제공하도록 ID 공급자를 사용할 수 있습니다. 중앙 액세스 관리를 위해 AWS IAM Identity Center(IAM Identity Center)를 사용하여 계정에 대한 액세스 권한과 해당 계정 내 권한을 관리하는 것이 좋습니다. IAM Identity Center를 사용하여 사용자 자격 증명을 관리하거나 외부 자격 증명 공급자의 IAM Identity Center에서 사용자 자격 증명에 대한 액세스 권한을 관리할 수 있습니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 AWS IAM Identity Center란 무엇입니까? 섹션을 참조하세요.

역할에 대한 자세한 내용은 역할 용어 및 개념 단원을 참조하세요.

AWS에 액세스하려면 워크로드에 IAM 역할이 있는 임시 자격 증명을 사용하도록 요구합니다.

워크로드란 애플리케이션이나 백엔드 프로세스같이 비즈니스 가치를 창출하는 리소스 및 코드 모음을 말합니다. 워크로드에는 AWS 서비스에 요청하기 위해 자격 증명이 필요한 애플리케이션, 운영 도구

및 구성 요소가 있을 수 있습니다(예: 데이터 읽기 요청). 이러한 자격 증명에는 AWS 환경에서 실행되는 컴퓨터가 포함됩니다(예: Amazon EC2 인스턴스 또는 AWS Lambda 함수).

또한 액세스 권한이 필요한 외부 당사자를 위해 시스템 자격 증명을 관리할 수도 있습니다. 시스템 자격 증명에 대한 액세스 권한을 부여하기 위해 IAM 역할을 사용할 수 있습니다. IAM 역할에는 특정 권한이 있으며 역할 세션과 함께 임시 보안 인증을 사용하여 AWS에 대한 액세스 방법을 제공합니다. 또한 AWS 외부에 AWS 환경에 대한 액세스 권한이 필요한 시스템이 있을 수 있습니다. AWS 외부에서 실행되는 시스템의 경우 AWS Identity and Access Management Roles Anywhere를 사용할 수 있습니다. 역할에 대한 자세한 내용은 IAM 역할 섹션을 참조하세요. 역할을 사용하여 AWS 계정 전체에서 액세스 권한을 위임하는 방법에 대한 자세한 내용은 튜토리얼: IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임을(를) 참조하세요.

다중 인증(MFA) 필요

임시 보안 인증을 사용할 수 있도록 AWS 리소스에 액세스하는 인간 사용자 및 워크로드에는 IAM 역할을 사용하는 것이 좋습니다. 하지만 계정에 IAM 사용자 또는 루트 사용자가 필요한 시나리오의 경우 추가 보안을 위해 MFA가 필요합니다. MFA에는 인증 문제에 응답을 생성하는 디바이스가 있습니다. 로그인 과정을 완료하려면 각 사용자의 보안 인증과 디바이스에서 생성한 응답이 필요합니다. 자세한 내용은 IAM의 AWS 다중 인증 단원을 참조하십시오.

인간 사용자에 대한 중앙 집중식 액세스 관리에 IAM Identity Center를 사용하는 경우 자격 증명 소스가 IAM Identity Center 자격 증명 스토어, AWS 관리형 Microsoft AD 또는 AD Connector와 함께 구성되면 IAM Identity Center MFA 기능을 사용할 수 있습니다. IAM Identity Center의 MFA에 관한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 다중 인증을 참조하세요.

장기 보안 인증이 필요한 사용 사례에 필요한 경우 액세스 키 업데이트

가능하면 액세스 키와 같은 장기 자격 증명을 만드는 대신 임시 보안 인증을 사용하는 것이 좋습니다. 하지만 프로그래밍 방식 액세스 권한과 장기 보안 인증이 있는 IAM 사용자가 필요한 시나리오의 경우 필요할 때(예: 직원 퇴사) 액세스 키를 업데이트하는 것이 좋습니다. IAM 액세스에서 마지막으로 사용한 정보를 사용하여 액세스 키를 안전하게 업데이트 및 제거하는 것이 좋습니다. 자세한 내용은 액세스 키 업데이트 단원을 참조하십시오.

AWS에서 IAM 사용자의 장기 보안 인증이 필요한 특정 사용 사례가 있습니다. 일부 사용 사례는 다음과 같습니다.

- IAM 역할을 사용할 수 없는 워크로드 - AWS에 액세스해야 하는 위치에서 워크로드를 실행할 수 있습니다. 일부 상황에서는 예를 들어, WordPress 플러그인에 대한 임시 보안 인증을 제공하기 위해

IAM 역할을 사용할 수 없습니다. 이러한 상황에서는 해당 워크로드에 IAM 사용자 장기 액세스 키를 사용하여 AWS에 대해 인증합니다.

- 타사 AWS 클라이언트 – IAM Identity Center를 사용한 액세스를 지원하지 않는 도구를 사용하는 경우(예: AWS에서 호스팅되지 않은 타사 AWS 클라이언트 또는 공급업체) IAM 사용자 장기 액세스 키를 사용합니다.
- AWS CodeCommit 액세스 – CodeCommit을 사용하여 코드를 저장하는 경우 CodeCommit에 대한 SSH 키 또는 서비스별 보안 인증이 있는 IAM 사용자를 사용하여 리포지토리를 인증할 수 있습니다. 일반 인증에 IAM Identity Center 사용자를 사용하는 것 외에 이렇게 하는 것이 좋습니다. IAM Identity Center 사용자는 AWS 계정 또는 클라우드 애플리케이션에 대한 액세스 권한이 필요한 인력의 사용자입니다. IAM 사용자를 구성하지 않고 CodeCommit 리포지토리에 대한 액세스 권한을 사용자에게 부여하기 위해 git-remote-codecommit 유틸리티를 구성할 수 있습니다. IAM 및 CodeCommit에 대한 자세한 내용은 CodeCommit용 IAM 자격 증명: Git 자격 증명, SSH 키 및 AWS 액세스 키 단원을 참조하세요. git-remote-codecommit 유틸리티 구성에 대한 자세한 내용은 AWS CodeCommit 사용 설명서의 보안 인증을 교체하여 AWS CodeCommit 리포지토리에 연결을 참조하세요.
- Amazon Keyspaces(Apache Cassandra용) 액세스 – IAM Identity Center 사용자를 사용할 수 없는 상황(예: Cassandra 호환성 테스트 목적)의 경우 서비스별 보안 인증이 있는 IAM 사용자를 사용하여 Amazon Keyspaces로 인증할 수 있습니다. IAM Identity Center 사용자는 AWS 계정 또는 클라우드 애플리케이션에 대한 액세스 권한이 필요한 인력의 사용자입니다. 임시 보안 인증을 사용하여 Amazon Keyspaces 연결할 수도 있습니다. 자세한 내용은 Amazon Keyspaces(Apache Cassandra용) 개발자 안내서의 임시 보안 인증과 IAM 역할 및 SigV4 플러그인을 사용하여 Amazon Keyspaces에 연결을 참조하세요.

모범 사례를 따라 루트 사용자 보안 인증을 보호하세요

AWS 계정을(를) 생성할 때 AWS Management Console에 로그인하기 위한 루트 사용자 보안 인증 정보를 설정합니다. 다른 민감한 개인 정보를 보호하는 것과 같은 방식으로 루트 사용자 보안 인증을 보호합니다. 루트 사용자 프로세스를 보호하고 확장하는 방법을 더 잘 이해하려면 AWS 계정에 대한 루트 사용자 모범 사례을(를) 참조하세요.

최소 권한 적용

IAM 정책을 사용하여 권한을 설정하는 경우 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. 워크로드 또는 사용 사례에 필요한 권한을 탐색하는 동안 광범위한 권한으로 시작할 수 있습니다. 사용 사례가 발전함에 따라 최소 권한을 향해 나아가도록 부여하는 권한을 줄이기 위해 노력할 수 있습니다. IAM을 사용하여 권한 적용에 대한 자세한 내용은 IAM의 정책 및 권한을(를) 참조하세요.

AWS 관리형 정책으로 시작하고 최소 권한을 향해 나아갑니다.

사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 관리형 정책은 AWS 계정에서 사용할 수 있습니다. AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있기 때문에 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 따라서 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다. 자세한 내용은 AWS 관리형 정책 단원을 참조하십시오. 특정 작업 기능을 위해 설계된 AWS 관리형 정책에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 섹션을 참조하세요.

IAM 액세스 분석기를 사용하여 액세스 활동을 기반으로 최소 권한 정책 생성

작업을 수행하는 데 필요한 권한만 부여하기 위해 AWS CloudTrail에 로깅된 액세스 활동에 따라 정책을 생성할 수 있습니다. IAM Access Analyzer는 IAM 역할이 사용하는 서비스와 작업을 분석한 다음 사용할 수 있는 세분화된 정책을 생성합니다. 생성된 각 정책을 테스트한 후 프로덕션 환경에 해당 정책을 배포할 수 있습니다. 이렇게 하면 워크로드에 필요한 권한만 부여할 수 있습니다. 정책 생성에 대한 자세한 내용은 IAM Access Analyzer 정책 생성을 참조하세요.

사용하지 않는 사용자, 역할, 권한, 정책 및 보안 인증은 정기적으로 검토하고 제거합니다.

AWS 계정에서 더 이상 필요하지 않은 IAM 사용자, 역할, 권한, 정책 또는 자격 증명이 있을 수 있습니다. IAM에서는 마지막으로 액세스한 정보를 제공하여 더 이상 필요하지 않은 사용자, 역할, 권한, 정책 및 자격 증명을 식별하여 제거할 수 있도록 도와줍니다. 이렇게 하면 모니터링해야 하는 사용자, 역할, 권한, 정책 및 보안 인증의 수를 줄일 수 있습니다. 이 정보를 사용하여 최소 권한을 더욱 철저하게 준수하도록 IAM 정책을 구체화할 수 있습니다. 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 단원을 참조하십시오.

IAM 정책의 조건을 사용하여 액세스 추가 제한

정책 명령문 적용 조건을 지정할 수 있습니다. 이렇게 하면 작업 및 리소스에 대한 액세스 권한을 부여할 수 있지만 액세스 요청이 특정 조건을 충족하는 경우에만 가능합니다. 예를 들어 SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있지만 특정 AWS 서비스(예: AWS CloudFormation)를 통해 사용되는 경우에만 서비스 작업에 대한 액세스 권한을 부여할 수 있습니다. 자세한 내용은 IAM JSON 정책 요소: Condition 단원을 참조하십시오.

IAM Access Analyzer를 사용하여 리소스에 대한 퍼블릭 및 크로스 계정 액세스 확인

AWS에서 퍼블릭 또는 크로스 계정 액세스에 대한 권한을 부여하기 전에 해당 액세스 권한이 필요한지 여부를 확인하는 것이 좋습니다. IAM Access Analyzer를 사용하면 지원되는 리소스 유형에 대한 퍼블릭 및 크로스 계정 액세스를 미리 보고 분석할 수 있습니다. IAM Access Analyzer에서 생성하는 결과를 검토하여 이렇게 할 수 있습니다. 이러한 결과는 리소스 액세스 제어가 예상한 액세스 권한을 부여하는지 확인하는 데 도움이 됩니다. 또한 퍼블릭 및 크로스 계정 권한을 업데이트할 때 리소스에 대한 새로운 액세스 제어를 배포하기 전에 변경 사항의 영향을 확인할 수 있습니다. 또한 IAM Access Analyzer는 지원되는 리소스 유형을 지속적으로 모니터링하고 퍼블릭 또는 크로스 계정 액세스를 허용하는 리소스에 대한 결과를 생성합니다. 자세한 내용은 IAM Access Analyzer API를 사용하여 액세스 미리 보기를 참조하세요.

IAM Access Analyzer를 사용하여 IAM 정책을 검증하여 안전하고 기능적인 권한을 보장합니다.

생성한 정책을 검증하여 IAM 정책 언어(JSON) 및 IAM 모범 사례를 준수하는지 확인합니다. IAM Access Analyzer 정책 검증을 사용하여 정책을 검증할 수 있습니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 권장 사항을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 콘솔에서 새 정책을 작성하거나 기존 정책을 편집할 때 IAM Access Analyzer는 정책을 저장하기 전에 정책을 구체화하고 검증하는 데 도움이 되는 권장 사항을 제공합니다. 또한 기존 정책을 모두 검토하고 검증하는 것이 좋습니다. 자세한 내용은 IAM Access Analyzer 정책 검증을 참조하세요. IAM Access Analyzer가 제공하는 정책 확인 사항에 대한 자세한 내용은 IAM Access Analyzer 정책 확인 참조를 참조하세요.

여러 계정에 권한 가드레일 설정

워크로드를 확장할 때 AWS Organizations에서 관리하는 여러 계정을 사용하여 워크로드를 분리합니다. Organizations 서비스 제어 정책(SCP)을 사용하여 계정 전체의 모든 IAM 사용자 및 역할에 대한 액세스를 제어하는 권한 가드레일을 설정합니다. SCP는 AWS 조직, OU 또는 계정 수준에서 조직의 권한을 관리하는 데 사용할 수 있는 조직 정책 유형입니다. 설정한 권한 가드레일은 해당 계정 내의 모든 사용자 및 역할에 적용됩니다. 그러나 SCP만으로는 조직 내 계정에 권한을 부여하기에 충분하지 않습니다. 이렇게 하려면 관리자는 실제로 권한을 부여하기 위해 여전히 자격 증명 기반 또는 리소스 기반 정책을 IAM 사용자, IAM 역할 또는 계정의 리소스에 연결해야 합니다. 자세한 내용은 AWS Organizations, 계정 및 IAM 가드레일을 참조하세요.

권한 경계를 사용하여 계정 내에서 권한 관리 위임

일부 시나리오에서는 한 계정 내의 권한 관리를 다른 사용자에게 위임하려고 할 수 있습니다. 예를 들어 개발자가 워크로드에 대한 역할을 만들고 관리하도록 허용할 수 있습니다. 다른 사람에게 권한을 위임하는 경우 권한 경계를 사용하여 위임하는 최대 권한을 설정합니다. 권한 경계는 관리형 정책을 사용하여 자격 증명 기반 정책을 통해 IAM 역할에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 권한 경계는 자신에게는 권한을 부여하지 않습니다. 자세한 내용은 IAM 엔터티의 권한 범위 단원을 참조하십시오.

AWS 계정에 대한 루트 사용자 모범 사례

AWS 계정을(를) 처음 생성하면 계정의 모든 AWS 리소스에 대한 완전한 액세스 권한이 있는 기본 보안 인증 세트로 시작합니다. 이 자격 증명을 AWS 계정 루트 사용자라고 합니다. 루트 사용자 보안 인증이 필요한 작업이 있는 경우가 아니면 AWS 계정 루트 사용자에게 액세스하지 않는 것이 좋습니다. 권한이 높은 보안 인증이 무단 사용에 노출되지 않도록 하려면 루트 사용자 보안 인증과 계정 복구 메커니즘을 안전하게 보호해야 합니다.

루트 사용자에게 액세스하는 대신 일상적인 작업을 위한 관리 사용자를 생성하세요.

- 새 AWS 계정이 있는 경우 AWS 계정 설정 섹션을 참조하세요.
- AWS Organizations을(를) 통해 여러 AWS 계정을(를) 관리하는 경우 IAM Identity Center 관리 사용자의 AWS 계정 액세스 설정을 참조하세요.

그런 다음, 관리 사용자를 활용하면 AWS 계정 내 리소스에 액세스해야 하는 사용자를 위한 추가 ID를 생성할 수 있습니다. AWS에 액세스하는 경우 임시 보안 인증을 사용하여 인증하라고 요구하는 것이 좋습니다.

- 단일 독립 실행형 AWS 계정의 경우, 계정에서 특정 권한을 가진 ID를 생성하는 데 IAM 역할을(를) 사용하세요. 역할은 역할이 필요한 사용자라면 누구나 맡을 수 있습니다. 또한 역할에는 그와 연관된 암호 또는 액세스 키와 같은 표준 장기 보안 인증이 없습니다. 대신에 역할을 맡은 사람에게는 해당 역할 세션을 위한 임시 보안 자격 증명이 제공됩니다. IAM 역할과 달리 IAM 사용자에는 암호 및 액세스 키와 같은 장기 보안 인증이 있습니다. 모범 사례로서, 가능하면 암호 및 액세스 키와 같은 장기 보안 인증 정보가 있는 IAM 사용자를 생성하는 대신 임시 보안 인증을 사용하는 것이 좋습니다.
- 조직을 통해 여러 AWS 계정을(를) 관리하는 경우 IAM Identity Center 인력 사용자를 사용하세요. IAM Identity Center를 사용하면 전체 AWS 계정 및 해당 계정 내 권한을 중앙에서 관리할 수 있습니다. IAM Identity Center 또는 외부 ID 공급자를 통해 사용자 ID를 관리합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 AWS IAM Identity Center란 무엇입니까? 단원을 참조하세요.

주제

루트 사용자 보안 인증을 보호하여 무단 사용 방지

루트 사용자 보안 인증을 보호하고 해당 보안 인증이 필요한 작업에만 사용하세요. 무단 사용을 방지하려면 루트 사용자 암호, MFA, 액세스 키, CloudFront 키 페어 또는 서명 인증서를 누구와도 공유하지 마세요. 단, 루트 사용자에 액세스해야 하는 업무상 엄격한 수요를 가진 사람은 예외입니다.

동일한 암호를 사용하여 액세스하는 계정의 AWS 서비스에 신뢰하는 도구와 함께 루트 사용자 암호를 저장하지 마세요. 루트 사용자 암호를 분실하거나 잊어버리면 이러한 도구에 액세스할 수 없습니다. 복원력을 우선시하고 두 명 이상의 사용자에게 스토리지 위치에 대한 액세스 권한을 부여하라고 요구하는 것이 좋습니다. 암호 또는 암호 보관 위치에 대한 액세스를 로그에 기록하고 모니터링해야 합니다.

액세스를 보호하려면 강력한 루트 사용자 암호 사용

강력하고 고유한 암호를 사용하는 것이 좋습니다. 강력한 암호 생성 알고리즘을 갖춘 암호 관리자와 같은 도구를 사용하면 이러한 목표를 달성할 수 있습니다. AWS에 따르면 암호가 다음 조건을 충족해야 합니다.

- 최소 8자, 최대 128자여야 합니다.
- 대문자, 소문자, 숫자, 기호(! @ # $ % ^ & * () <> [] {} | _+-=) 중 적어도 세 가지 문자 유형을 혼합하여 포함해야 합니다.
- AWS 계정 이름 또는 이메일 주소와 동일하지 않아야 합니다.

자세한 내용은 AWS 계정 루트 사용자의 암호 변경 단원을 참조하세요.

다중 인증(MFA)을 통한 루트 사용자 로그인 보호

루트 사용자는 권한 있는 작업을 수행할 수 있으므로 로그인 보안 인증으로 이메일 주소 및 암호 외에도 루트 사용자를 위한 MFA를 두 번째 인증 요소로 추가하는 것이 중요합니다. 보안 전략의 유연성과 탄력성을 높이기 위해 루트 사용자 보안 인증에 여러 MFA를 활성화하는 것이 좋습니다. 현재 지원되는 MFA 유형을 조합하여 최대 8개의 MFA 디바이스를 AWS 계정 루트 사용자에게 등록할 수 있습니다.

- FIDO 인증 하드웨어 보안 키는 타사 제공업체에서 제공합니다. 자세한 내용은 AWS 계정 루트 사용자의 FIDO 보안 키 활성화를 참조하세요.
- 시간 기반 일회용 암호(TOTP) 알고리즘에 따라 6자리 숫자 코드를 생성하는 하드웨어 디바이스입니다. 자세한 내용은 AWS 계정 루트 사용자의 하드웨어 TOTP 토큰 활성화를 참조하세요.
- 전화 또는 기타 디바이스에서 실행되고 물리적 디바이스를 에뮬레이트하는 가상 인증 애플리케이션입니다. 자세한 내용은 AWS 계정 루트 사용자를 위한 가상 MFA 디바이스 활성화를 참조하세요.

루트 사용자에 대해 액세스 키를 생성하지 않음

액세스 키를 사용하면 명령줄 인터페이스(AWS CLI) 에서 AWS 명령을 실행하거나 AWS SDK 중 하나에서 API 작업을 사용할 수 있습니다. 루트 사용자는 결제 정보를 포함하여 계정 내 모든 AWS 서비스 및 리소스에 대한 전체 액세스 권한을 가지므로 루트 사용자용 액세스 키 쌍을 만들지 않는 것이 좋습니다.

일부 작업에만 루트 사용자가 필요하고 이러한 작업은 대개 자주 수행하지 않으므로 AWS Management Console에 로그인하여 루트 사용자 작업을 수행하는 것이 좋습니다. 액세스 키를 생성하기 전에 장기 액세스 키의 대안을 검토합니다.

가능하면 루트 사용자 로그인에 여러 사람 승인 사용

한 사람이 루트 사용자의 MFA와 암호에 모두 액세스할 수 없도록 복수 사용자 승인을 사용하는 것을 고려해 보세요. 일부 회사는 암호에 액세스할 수 있는 관리자 그룹 하나와 MFA에 액세스할 수 있는 다른 관리자 그룹을 설정하여 보안 계층을 추가합니다. 루트 사용자으로 로그인하려면 각 그룹에서 한 명의 구성원이 함께 모여야 합니다.

루트 사용자 보안 인증에 그룹 이메일 주소 사용

회사에서 관리하며 받은 메시지를 관리하여 사용자 그룹에 직접 전달하는 이메일 주소를 사용합니다. AWS이이(가) 계정 소유자에게 연락해야 하는 경우, 이 접근 방식을 사용하면 개인이 휴가 중이거나, 아프거나, 회사를 떠난 경우에도 응답이 지연될 위험이 감소합니다. 루트 사용자가 사용하는 이메일 주소는 다른 용도로 사용할 수 없습니다.

계정 복구 메커니즘에 대한 액세스 제한

관리자 계정 탈취와 같은 긴급 상황에서 루트 사용자 보안 인증 복구 메커니즘에 액세스해야 하는 경우에 대비하여 해당 메커니즘을 관리하는 프로세스를 개발해야 합니다.

- 분실하거나 잊어버린 루트 사용자 암호를 재설정할 수 있도록 루트 사용자 이메일 수신함에 액세스할 수 있는지 확인하세요.
- AWS 계정 루트 사용자에 대한 MFA가 분실되었거나, 손상되었거나, 작동하지 않는 경우 동일한 루트 사용자 보안 인증에 등록된 다른 MFA 디바이스를 사용하여 로그인할 수 있습니다. 모든 MFA에 액세스할 수 없는 경우 전화번호와 이메일이 계정을 등록하는 데 사용되었으며 모두 최신 상태이고 액세스 가능해야 MFA를 복구할 수 있습니다. 자세한 내용은 루트 사용자 MFA 디바이스 복구를 참조하세요.
- 루트 사용자 암호와 MFA를 저장하지 않기로 선택한 경우 계정에 등록된 전화번호를 루트 사용자 보안 인증을 복구하는 다른 방법으로 사용할 수 있습니다. 연락처 전화번호에 액세스할 수 있는지 확인하고, 전화번호를 최신 상태로 유지하고, 전화번호를 관리할 수 있는 액세스 권한을 제한하세요.

둘 다 루트 사용자 암호를 복구할 수 있는 확인 채널이므로 어느 누구도 이메일 수신함과 전화번호 모두에 액세스할 수 없어야 합니다. 이러한 채널을 관리하는 두 그룹의 개인이 있어야 한다는 것이 중요합니다. 한 그룹은 기본 이메일 주소에 액세스할 수 있고 다른 그룹은 기본 전화번호에 액세스하여 루트 사용자로서 계정에 대한 액세스 권한을 복구할 수 있습니다.

조직 계정 루트 사용자 보안 인증 보호

조직을 통한 다중 계정 전략으로 전환할 때는 AWS 계정마다 보호해야 하는 고유한 루트 사용자 보안 인증이 있습니다. 조직을 만들 때 사용하는 계정은 관리 계정이고 조직의 나머지 계정은 구성원 계정입니다.

구성원 계정의 루트 사용자 보안 인증 보호

조직을 사용하여 여러 계정을 관리하는 경우 조직에서 루트 사용자 액세스를 보호하기 위해 취할 수 있는 두 가지 전략이 있습니다.

- MFA를 사용하여 조직 계정의 루트 사용자 보안 인증을 보호합니다.
- 계정의 루트 사용자 암호를 재설정하지 말고 필요할 때만 암호 재설정 프로세스를 사용하여 액세스 권한을 복구하세요. 조직에 구성원 계정을 생성하면 조직은 관리 계정이 구성원 계정에 임시로 액세스하도록 하는 IAM 역할을 구성원 계정에 자동으로 생성합니다.

자세한 내용은 조직 사용 설명서의 조직 내 구성원 계정 액세스를 참조하세요.

서비스 제어 정책(SCP)을 사용하여 조직에서 예방적 보안 제어 설정

조직을 사용하여 여러 계정을 관리하는 경우 SCP를 적용하여 구성원 계정 루트 사용자에 대한 액세스를 제한할 수 있습니다. 특정 루트 전용 작업을 제외하고 구성원 계정에서 모든 루트 사용자 작업을 거부하면 무단 액세스를 방지하는 데 도움이 됩니다. 자세한 내용은 SCP를 사용하여 구성원 계정의 루트 사용자가 수행할 수 있는 작업 제한하기를 참조하세요.

액세스 및 사용 모니터링

현재 추적 메커니즘을 사용하여 루트 사용자 로그인 및 사용량을 알리는 알림을 포함하여 루트 사용자 보안 인증의 로그인 및 사용을 모니터링, 경고 및 보고하는 것이 좋습니다. 다음 서비스는 루트 사용자 보안 인증 사용량을 추적하고 무단 사용을 방지하는 데 도움이 되는 보안 검사를 수행하는 데 도움이 될 수 있습니다.

- 계정의 루트 사용자 로그인 활동에 대한 알림을 받으려면 Amazon CloudWatch를 활용하여 루트 사용자 보안 인증이 사용되는 시기를 탐지하고 보안 관리자에게 알림을 트리거하는 이벤트 규칙을 생성할 수 있습니다. 자세한 내용은 AWS 계정 루트 사용자 활동의 모니터링 및 통지를 참조하세요.
- 승인된 루트 사용자 작업을 알리도록 알림을 설정하려면 Amazon SNS와 함께 Amazon EventBridge를 활용하여 특정 작업에 대한 루트 사용자 사용량을 추적하고 Amazon SNS 주제를 사용하여 알리는 EventBridge 규칙을 작성할 수 있습니다. 예시는 Amazon S3 객체 생성 시 알림 보내기를 참조하세요.
- 이미 GuardDuty를 위협 탐지 서비스로 사용하고 있다면 계정에서 루트 사용자 보안 인증이 사용될 때 알림을 받도록 기능을 확장할 수 있습니다.

알림에는 루트 사용자에 대한 이메일 주소가 포함되나 이에 국한되지는 않습니다. 루트 사용자 액세스 경고를 받은 담당자는 루트 사용자 액세스가 정상적인지 확인하는 방법을 이해하고, 보안 인시던트가 진행 중이라고 판단되는 경우 에스컬레이션하는 방법을 이해할 수 있도록 경고에 대응하는 절차를 마련합니다. 경고를 구성하는 방법에 대한 예시는 AWS 계정 루트 사용자 활동의 모니터링 및 통지를 참조하세요.

루트 사용자 MFA 준수 평가

- AWS Config은(는) 규칙을 사용하여 루트 사용자 모범 사례를 적용하도록 지원합니다. AWS 관리형 규칙을 사용하여 루트 사용자에게 다중 인증(MFA)를 활성화하라고 요구할 수 있습니다. 또한, AWS Config은(는) 루트 사용자의 액세스 키를 식별할 수 있습니다.

- Security Hub는 AWS에서 보안 상태를 포괄적으로 볼 수 있으며 루트 사용자에게 MFA를 적용하고 루트 사용자 액세스 키를 사용하지 않는 등 보안 업계 표준 및 모범 사례와 비교하여 AWS 환경을 평가하는 데 도움이 됩니다. 사용 가능한 규칙에 대한 자세한 내용은 Security Hub 사용 설명서의 AWS Identity and Access Management 제어를 참조하세요.
- Trusted Advisor은(는) 루트 사용자 계정에서 MFA가 활성화되지 않았는지 알 수 있도록 보안 검사를 제공합니다. 자세한 내용은 AWS 지원 사용 설명서의 루트 계정에 대한 MFA를 참조하세요.

계정의 보안 문제를 신고해야 하는 경우 의심스러운 이메일 신고 또는 취약성 신고를 참조하세요. 또는 AWS에 문의하여 지원 및 추가 안내 요청을 받을 수도 있습니다.

IAM에 대한 기업 사용 사례

IAM의 간단한 기업 사용 사례를 통해 사용자의 AWS 액세스 권한을 제어하기 위한 서비스 구현의 기본적인 방법을 이해할 수 있습니다. 사용 사례는 일반적인 용어로 서술되며 원하는 결과를 달성하기 위해 IAM API를 사용하는 방법에 대한 기술적인 내용을 다루지 않습니다.

이 사용 사례에서는 Example Corp라는 가상의 회사가 IAM을 사용하는 2가지 일반적인 방법에 대해 살펴보겠습니다. 첫 번째 시나리오는 Amazon Elastic Compute Cloud(Amazon EC2)를 고려합니다. 두 번째는 Amazon Simple Storage Service(Amazon S3)를 고려합니다.

다른 AWS 서비스와 함께 IAM을 사용하는 방법에 대한 자세한 정보는 AWS IAM으로 작업하는 서비스 섹션 섹션을 참조하세요.

주제

- Example Corp의 초기 설정
- Amazon EC2에서의 IAM 사용 사례
- Amazon S3에서의 IAM의 사용 사례

Example Corp의 초기 설정

Nikki Wolf와 Mateo Jackson은 Example Corp의 창립자입니다. 회사를 시작할 때 그들은 AWS 계정을 생성하고 AWS IAM Identity Center(IAM Identity Center)를 설정하여 AWS 리소스와 함께 사용할 관리 계정을 생성합니다. 관리 사용자에 대한 계정 액세스를 설정하면 IAM Identity Center에서 해당 IAM 역할을 생성합니다. IAM Identity Center에서 제어하는 이 역할은 관련 AWS 계정에 생성되며 AdministratorAccess 권한 세트에 지정된 정책이 역할에 연결됩니다.

이제 관리자 계정이 있으므로 Nikki와 Mateo는 더 이상 루트 사용자를 사용하여 AWS 계정에 액세스할 필요가 없습니다. 이들은 루트 사용자만 수행할 수 있는 태스크를 완료하는 데 루트 사용자만 사용할 계획입니다. 보안 모범 사례를 검토한 후 루트 사용자 보안 인증에 대한 다중 인증(MFA)을 구성하고 루트 사용자 보안 인증을 보호하는 방법을 결정합니다.

회사가 성장함에 따라 개발자, 관리자, 테스터, 관리자 및 시스템 관리자로 일할 직원을 고용합니다. Nikki는 운영을 담당하고 Mateo는 엔지니어링 팀을 관리합니다. 이들은 Active Directory 도메인 서버를 설정하여 직원 계정과 회사 내부 리소스에 대한 액세스를 관리합니다.

직원에게 AWS 리소스에 대한 액세스 권한을 부여하기 위해 IAM Identity Center를 사용하여 회사의 Active Directory를 AWS 계정에 연결합니다.

Active Directory를 IAM Identity Center에 연결했기 때문에 사용자, 그룹 및 그룹 멤버십이 동기화되고 정의됩니다. AWS 리소스에 대한 올바른 수준의 액세스 권한을 사용자에게 부여하기 위해 서로 다른 그룹에 권한 세트와 역할을 할당해야 합니다. 이들은 AWS Management Console에서 직무에 관한 AWS 관리형 정책을 사용하여 다음과 같은 권한 세트를 생성합니다.

- 관리자
- 결제
- 개발자
- 네트워크 관리자
- 데이터베이스 관리자
- 시스템 관리자
- 지원 사용자

그런 다음 이러한 권한 세트를 Active Directory 그룹에 할당된 역할에 할당합니다.

IAM Identity Center의 초기 구성을 설명하는 단계별 가이드는 AWS IAM Identity Center 사용 설명서의 Getting started(시작하기)를 참조하세요. IAM Identity Center 사용자 액세스 권한 프로비저닝에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 Single sign-on access to AWS accounts(AWS 계정에 대한 Single Sign-On 액세스)를 참조하세요.

Amazon EC2에서의 IAM 사용 사례

Example Corp와 같은 회사는 일반적으로 IAM을 사용하여 Amazon EC2와 같은 서비스와 상호 작용합니다. 이 부분의 사용 사례를 이해하기 위해서는 Amazon EC2에 대한 기본적인 지식이 필요합니다. Amazon EC2에 대한 자세한 내용은 Amazon EC2 사용 설명서를 참조하세요.

그룹에 대한 Amazon EC2 권한

'경계' 제어를 제공하기 위해 Nikki는 정책을 AllUsers 사용자 그룹에 연결합니다. 이 정책은 Example Corp 회사 네트워크의 IP 주소가 아닌 주소에서 시작된 모든 사용자의 AWS 요청을 거부합니다.

Example Corp는 다음과 같이 사용자 그룹에 따라 서로 다른 권한을 부여했습니다.

- 시스템 관리자 - AMI, 인스턴스, 스냅샷, 볼륨, 보안 그룹 등을 생성하고 관리하기 위한 권한이 필요합니다. Nikki는 그룹 멤버에게 모든 Amazon EC2 작업을 사용할 수 있는 권한을 부여하는 `AmazonEC2FullAccess` AWS 관리형 정책을 SysAdmins 사용자 그룹에 연결합니다.
- 개발자 - 인스턴스를 사용한 작업 권한만 필요합니다. 따라서 Mateo는 개발자가 `DescribeInstances`, `RunInstances`, `StopInstances`, `StartInstances`, `TerminateInstances`를 호출할 수 있는 권한을 부여하는 정책을 생성하고 Developers 사용자 그룹에 연결합니다.

  > **Note**
  >
  > Amazon EC2는 SSH 키, Windows 암호 및 보안 그룹을 사용하여 특정 Amazon EC2 인스턴스의 운영 체제에 액세스할 사용자를 제어합니다. IAM 시스템에서는 특정 인스턴스의 운영 체제 액세스를 허용 또는 거부할 방법을 제공하지 않습니다.

- 지원 사용자 - 현재 제공되고 있는 Amazon EC2 리소스를 나열하는 것 외에는 어떤 Amazon EC2 작업도 수행할 필요가 없습니다. 따라서 Nikki는 Amazon EC2 'Describe' API 작업만 호출할 수 있는 권한을 부여하는 정책을 생성하고 지원 그룹에 연결합니다.

이러한 정책의 예제를 보려면 IAM 자격 증명 기반 정책의 예 및 Amazon EC2 사용 설명서의 AWS Identity and Access Management를 참조하세요.

사용자의 직무 변경

그러다가 개발자 중 한 명인 Paulo Santos가 직무를 바꾸어 관리자가 되었습니다. 관리자로서 Paulo는 개발자를 위한 지원 사례를 열 수 있도록 Support users 그룹의 일원이 됩니다. Mateo은 Paulo를 Developers 사용자 그룹에서 관리자 사용자 그룹으로 옮겼습니다. 이로 인해 Amazon EC2 인스턴스와 상호 작용하는 기능이 제한됩니다. 즉, 인스턴스를 실행하거나 시작할 수 없으며, 이전에 자신이 시작한 인스턴스일지라도 더 이상 기존 인스턴스를 중지하거나 종료할 수 없습니다. Example Corp 사용자가 시작한 인스턴스를 나열할 수만 있습니다.

Amazon S3에서의 IAM의 사용 사례

Example Corp와 같은 회사는 또한 기본적으로 IAM과 함께 Amazon S3를 사용합니다. John은 aws-s3-bucket이라는 회사용 Amazon S3 버킷을 생성했습니다.

추가 사용자와 사용자 그룹 생성

직원인 Zhang Wei와 Mary Major는 모두 회사의 버킷에 데이터를 생성할 수 있어야 합니다. 또한 개발자들이 작업 중인 공유 데이터를 읽고 쓸 수 있어야 합니다. 이를 위해 Mateo는 다음 그림과 같은 Amazon S3 키 접두사 체계에 따라 aws-s3-bucket의 데이터에 대한 논리적 구조를 정했습니다.

```
/aws-s3-bucket
    /home
        /zhang
        /major
    /share
        /developers
        /managers
```

Mateo는 `/aws-s3-bucket`을 각 직원별 홈 디렉터리, 그리고 개발자와 관리자의 그룹에서 함께 공유하는 영역으로 나누었습니다.

그런 다음 Mateo는 다음과 같이 사용자와 사용자 그룹에 대해 권한을 부여하는 정책의 조합을 생성했습니다.

- Zhang의 홈 디렉터리 액세스 - Mateo는 Wei에게 Amazon S3 키 접두사 로 객체에 대해 읽기, 쓰기 및 나열 권한을 부여하는 정책을 연결했습니다. `/aws-s3-bucket/home/zhang/`
- Major의 홈 디렉터리 액세스 - Mateo는 Mary에게 Amazon S3 키 접두사 로 객체에 대해 읽기, 쓰기 및 나열 권한을 부여하는 정책을 연결했습니다. `/aws-s3-bucket/home/major/`
- Developers 사용자 그룹에 대한 공유 디렉터리 액세스 - Mateo는 개발자에게 `/aws-s3-bucket/share/developers/` 키 접두사로 객체에 대해 읽기, 쓰기 및 나열 권한을 부여하는 정책을 사용자 그룹에 연결했습니다.
- Managers 사용자 그룹에 대한 공유 디렉터리 액세스 - Mateo는 관리자에게 `/aws-s3-bucket/share/managers/` 키 접두사로 객체에 대해 읽기, 쓰기 및 나열 권한을 부여하는 정책을 사용자 그룹에 연결했습니다.

> **Note**
>
> Amazon S3는 버킷 또는 객체를 만든 사용자에게 해당 버킷 또는 객체에 대해 자동으로 다른 작업을 수행할 권한을 부여하지 않습니다. 따라서 IAM 정책에서 명시적으로 사용자에게 사용자가 생성한 Amazon S3 리소스를 사용할 권한을 부여해야 합니다.

이러한 각 정책의 예를 보려면 Amazon Simple Storage Service 사용 설명서의 액세스 제어를 참조하세요. 런타임 시 정책이 어떻게 평가되는지 알아보려면 정책 평가 로직 섹션을 참조하세요.

사용자의 직무 변경

그러다가 개발자 중 한 명인 Zhang Wei가 직무를 바꾸어 관리자가 되었습니다. 따라서 더 이상 `share/developers` 디렉터리의 문서에 액세스할 필요가 없으므로 관리자인 Mateo는 Wei를 `Managers` 사용자 그룹에서 `Developers` 사용자 그룹으로 옮겼습니다. 이처럼 간단한 재할당만으로 `Managers` 사용자 그룹에 허가된 모든 권한이 자동으로 Wei에게 부여되고, 더 이상 `share/developers` 디렉터리의 데이터에서는 액세스하지 못하게 됩니다.

서드 파티 기업과 통합

기업은 종종 파트너 업체와 컨설턴트, 계약자들과 작업합니다. Example Corp는 Widget Company라는 파트너가 있으며, 이 Widget Company의 직원인 Shirley Rodriguez에게 Example Corp에서 사용하는 버킷에 데이터를 추가할 권한을 부여해야 합니다. Nikki는 WidgetCo라는 사용자 그룹과 `Shirley`라는 사용자를 생성하고 Shirley를 WidgetCo 사용자 그룹에 추가했습니다. Nikki는 또한 Shirley가 사용할 수 있는 aws-s3-bucket1이라는 특수 버킷을 만듭니다.

Nikki는 기존 정책을 업데이트하거나 새 정책을 추가하여 Widget Company 파트너에게 적절한 권한을 부여할 수 있습니다. 예를 들어 Nikki는 WidgetCo 사용자 그룹의 멤버에게는 쓰기 이외의 모든 작업을 사용할 권한을 거부하는 새 정책을 생성할 수 있습니다. 모든 사용자에게 광범위한 Amazon S3 작업에 대해 액세스 권한을 부여하는 정책이 있을 경우에만 이 정책이 필요합니다.

IAM 자습서

다음 튜토리얼에서는 AWS Identity and Access Management(IAM)에 대한 일반적인 작업을 처음부터 끝까지 수행하는 절차를 보여줍니다. 이러한 절차는 가상 회사 이름 및 사용자 이름 등을 사용하여 랩 유형의 환경에 맞게 고안되었습니다. 그 목적은 일반적인 지침을 제공하는 것입니다. 조직의 고유한 요구 사항에 맞는지 주의 깊은 검토 및 조정 없이 프로덕션 환경에 바로 사용할 수 있도록 고안된 것이 아닙니다.

자습서

- 튜토리얼: IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임
- IAM 튜토리얼: 첫 번째 고객 관리형 정책 만들기 및 연결
- IAM 튜토리얼: 태그를 기반으로 AWS 리소스에 액세스할 수 있는 권한 정의
- IAM 자습서: 사용자가 자신의 자격 증명 및 MFA 설정을 관리하도록 허용

튜토리얼: IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임

> **Important**
>
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자를 사용하는 대신, 인간 사용자가 자격 증명 공급자와의 페더레이션을 사용하여 임시 보안 인증으로 AWS에 액세스하도록 하는 것입니다.

이 자습서에서는 역할을 사용하여 Destination과 Originating이라는 서로 다른 AWS 계정의 리소스에 대한 액세스 권한을 위임하는 방법을 설명합니다. 한 계정의 리소스는 다른 계정의 사용자와 공유합니다. 이러한 방식으로 크로스 계정 액세스를 설정하면 각 계정에 개별 IAM 사용자를 생성할 필요가 없습니다. 또한 사용자는 다른 AWS 계정의 리소스에 액세스하기 위해 한 계정에서 로그아웃하고 다른 계정에 로그인할 필요가 없습니다. 역할을 구성한 후에는 AWS Management Console, AWS CLI 및 API에서 역할을 사용하는 방법에 대해서도 알아봅니다.

이 자습서에서 Destination 계정은 다양한 애플리케이션과 팀이 액세스하는 애플리케이션 데이터를 관리합니다. 각 계정에서 Amazon S3 버킷에 애플리케이션 정보를 저장합니다. Developers와 Analysts라는 두 가지 IAM 사용자 역할이 있는 Originating 계정에서 IAM 사용자를 관리합니다. Developers와 Analysts는 Originating 계정을 사용하여 여러 마이크로서비스가 공유하는 데이터를 생성합니다. 두 역할 모두 Originating 계정에서 작업하고 이 계정의 리소스에 액세스할 수 있는 권한이 있

습니다. 개발자는 종종 Destination 계정의 공유 데이터를 업데이트해야 합니다. 개발자는 이 데이터를 `shared-container`라는 Amazon S3 버킷에 저장합니다.

이 자습서의 마지막에서는 다음 항목을 갖게 됩니다.

- Destination 계정의 특정 역할을 수임할 수 있는 Originating 계정(신뢰할 수 있는 계정)의 사용자.
- 특정 Amazon S3 버킷에 액세스할 수 있는 Destination 계정(신뢰하는 계정)의 역할.
- Destination 계정의 `shared-container` 버킷.

개발자들은 AWS Management Console에서 이 역할을 사용하여 Destination 계정의 `shared-container` 버킷에 액세스할 수 있습니다. 또한 역할을 통해 제공되는 임시 자격 증명으로 API 호출을 인증함으로써 버킷에 액세스하는 것도 가능합니다. 하지만 분석가가 이 역할을 사용하려는 비슷한 시도는 실패합니다.

이 워크플로우는 세 가지 기본 단계로 이루어집니다.

Destination 계정에 역할 생성

먼저 AWS Management Console을 사용하여 Destination 계정(ID 번호 999999999999)과 Originating 계정(ID 번호 111111111111) 사이에 신뢰를 설정합니다. UpdateData라는 IAM 역할을 생성하여 시작합니다. 역할을 생성할 때 Originating 계정을 신뢰할 수 있는 엔터티로 정의하고 신뢰할 수 있는 사용자가 `shared-container` 버킷을 업데이트하는 것을 허용하는 권한 정책을 지정합니다.

역할에 대한 액세스 권한 부여

이 섹션에서는 분석가들이 `UpdateData` 역할에 액세스하는 것을 거부하도록 역할 정책을 수정합니다. 분석가들은 이 시나리오에서 PowerUser 액세스 권한을 갖기 때문에 역할을 사용하지 못하도록 명시적으로 거부해야 합니다.

역할을 전환하여 액세스 테스트

마지막으로, 개발자로서 `UpdateData` 역할을 사용하여 Destination 계정의 `shared-container` 버킷을 업데이트합니다. 이제 AWS 콘솔, AWS CLI 및 API를 통해 역할에 액세스할 수 있게 되었습니다.

고려 사항

IAM 역할을 사용하여 AWS 계정 전반의 리소스 액세스를 위임하기 전에 다음 사항을 고려해야 합니다.

- AWS 계정 루트 사용자로 로그인하면 역할을 바꿀 수 없습니다.
- IAM 역할 및 리소스 기반 정책은 단일 파티션 내에서만 계정 간에 액세스 권한을 위임합니다. 예를 들어 표준 aws 파티션의 미국 서부(캘리포니아 북부)에 계정이 있다고 가정합니다. aws-cn 파티션의 중국(베이징)에도 계정이 있습니다. 중국(베이징)의 계정에서 Amazon S3 리소스 기반 정책을 사용하여 표준 aws 계정의 사용자에 대한 액세스를 허용할 수 없습니다.
- AWS IAM Identity Center을 사용하면 SAML(Security Assertion Markup Language)을 사용한 외부 AWS 계정(사용자 AWS Organizations 외부의 계정)의 Single Sign-On(SSO)을 촉진할 수 있습니다. 자세한 내용은 Integrate external AWS 계정 into AWS IAM Identity Center for central access management with independent billing using SAML 2.0을 참조하세요.
- Amazon EC2 인스턴스 또는 AWS Lambda 함수와 같은 AWS 리소스에 역할을 연결할 수 있습니다. 세부 정보는 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.
- 애플리케이션이 다른 AWS 계정의 역할을 수임하도록 하려면 크로스 계정 역할 수임에 AWS SDK를 사용할 수 있습니다. 자세한 내용은 AWS SDK 및 도구 참조 안내서의 위임 및 액세스를 참조하세요.
- AWS Management Console을 사용한 역할 전환은 ExternalId가 필요하지 않은 계정에서만 작동합니다. 예를 들어, 계정에 대한 액세스 권한을 타사에 부여하고 권한 정책의 Condition 요소에 ExternalId가 필요하다고 가정합니다. 이 경우 타사는 AWS API 또는 명령줄 도구를 사용해야만 계정에 액세스할 수 있습니다. ExternalId에 대한 값을 제공해야 하기 때문에 타사는 콘솔을 사용할 수 없습니다. 이 시나리오에 대한 자세한 정보는 타사가 소유한 AWS 계정에 액세스 및 AWS 보안 블로그의 AWS Management Console에 대한 크로스 계정 액세스를 가능하게 하는 방법을 참조하세요.

사전 조건

이 자습서에서는 다음을 이미 완료했다고 가정합니다.

- 사용할 수 있는 2개의 개별 AWS 계정(Originating 계정을 대표하는 1개와 Destination 계정을 대표하는 1개).
- Originating 계정에서 다음과 같이 생성 및 구성된 사용자 및 역할.

직무	User	권한
개발자	David	두 사용자 모두 Originating 계정의 AWS Management Console에 로그인하고 사용할 수 있습니다.
분석가	Jane	

- Destination 계정에는 사용자를 생성할 필요가 없습니다.

- Destination 계정에서 생성된 Amazon S3 버킷. 이 자습서에서는 이를 `shared-container`(이)라고 부릅니다. 하지만 S3 버킷 이름은 전역에서 고유해야 하므로 다른 이름의 버킷을 사용해야 합니다.

Destination 계정에 역할 생성

한 AWS 계정의 사용자가 다른 AWS 계정의 리소스에 액세스하도록 허용할 수 있습니다. 이 자습서에서는 해당 리소스에 액세스할 수 있는 사용자와 해당 역할로 전환하는 사용자에게 부여할 권한을 정의하는 역할을 생성하여 액세스를 허용합니다.

자습서의 이 단계에서는 Destination 계정에서 역할을 생성하고 Originating 계정을 신뢰할 수 있는 엔터티로 지정합니다. 또한 역할 권한을 `shared-container` 버킷에 대한 읽기 및 쓰기 액세스 권한으로 제한합니다. 따라서 역할을 사용할 수 있는 권한을 받은 사용자는 누구나 `shared-container` 버킷에 대해 읽기나 쓰기가 가능합니다.

역할을 생성하기 전에 Originating AWS 계정의 계정 ID가 필요합니다. 각 AWS 계정은 할당된 고유 계정 ID 식별자를 갖습니다.

Originating AWS 계정 ID를 가져오는 방법

1. Originating 계정 관리자로 AWS Management Console에 로그인한 후 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. IAM 콘솔에서 상단 오른쪽 모서리에 있는 탐색 표시줄에서 사용자 이름을 선택합니다. 일반적인 형식은 ***username***@***account_ID_number_or_alias***입니다.

   이 시나리오에서는 Originating 계정에 계정 ID 111111111111을 사용할 수 있습니다. 그러나 테스트 환경에서 시나리오를 재구성하는 경우에는 유효한 계정 ID를 사용해야 합니다.

Originating 계정이 사용할 수 있는 역할을 Destination 계정에 생성하는 방법

1. Destination 계정 관리자로 에AWS Management Console 로그인한 후 IAM 콘솔을 엽니다.
2. 역할을 만들기 전에 먼저 해당 역할에 필요한 권한을 정의하는 관리형 정책을 준비합니다. 차후 단계에서 이 정책을 해당 역할에 연결합니다.

   `shared-container` 버킷에 대한 읽기 및 쓰기 액세스 권한을 설정하려고 합니다. AWS에 이미 몇 가지 Amazon S3 관리형 정책이 있지만, 단일 Amazon S3 버킷에 대한 읽기 및 쓰기 액세스가 가능한 정책은 없습니다. 대신에 사용자가 직접 정책을 생성할 수 있습니다.

   탐색 창에서 정책을 선택한 후 정책 생성을 선택합니다.

3. JSON 탭을 선택하고 다음 JSON 정책 문서에서 텍스트를 복사합니다. 이 텍스트를 JSON 텍스트 상자에 붙여 넣어 리소스 ARN(`arn:aws:s3:::shared-container`)을 실제 Amazon S3 버킷에 대한 ARN으로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
       ],
      "Resource": "arn:aws:s3:::shared-container"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::shared-container/*"
    }
  ]
}
```

`ListAllMyBuckets` 작업은 요청의 인증된 발신자가 소유한 모든 버킷을 나열할 수 있는 권한을 부여합니다. `ListBucket`은 사용자가 `shared-container` 버킷에 저장되어 있는 객체를 볼 수 있는 권한입니다. `GetObject`, `PutObject` 및 `DeleteObject`는 사용자가 `shared-container` 버킷에 저장되어 있는 콘텐츠를 각각 보거나, 업데이트하거나, 삭제할 수 있는 권한입니다.

> **Note**
>
> 언제든지 시각적 편집기 옵션과 JSON 편집기 옵션 간에 전환할 수 있습니다. 그러나 변경을 적용하거나 시각적 편집기에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화되도록 정책을 재구성할 수 있습니다. 자세한 내용은 정책 재구성 단원을 참조하십시오.

4. 검토 및 생성 페이지에서 정책 이름에 **`read-write-app-bucket`**을 입력합니다. 정책이 부여한 권한을 검토한 다음 정책 생성을 선택하여 작업을 저장합니다.

   새로운 정책이 관리형 정책 목록에 나타납니다.
5. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
6. AWS 계정 역할 유형을 선택합니다.
7. 계정 ID에 Originating 계정 ID를 입력합니다.

   이 자습서에서는 Originating 계정의 예제 ID **111111111111**을 사용합니다. 하지만 실제로는 유효한 계정 ID를 사용해야 합니다. **111111111111**과 같이 잘못된 계정 ID를 사용할 경우, IAM에서 새로운 역할을 생성할 수 없습니다.

   이 연습에서는 외부 ID를 요구하거나, 사용자가 역할을 위임하기 위해 멀티 팩터 인증(MFA)을 요구할 필요가 없습니다. 이러한 옵션을 선택하지 않은 상태로 두십시오. 자세한 내용은 IAM의 AWS 다중 인증 단원을 참조하십시오.
8. 다음: 권한을 선택하여 역할과 연결된 권한을 설정합니다.
9. 앞에서 생성한 정책 옆의 확인란을 선택합니다.

> **도움말**
>
> Filter(필터)에서 Customer managed(고객 관리형)을 선택하여 생성한 정책만 포함하도록 목록을 필터링합니다. 이렇게 하면 AWS에서 생성한 정책이 표시되지 않아서 필요한 정책을 쉽게 찾을 수 있습니다.

   그리고 다음을 선택합니다.
10. (선택 사항)태그를 키-값 페어로 연결하여 메타데이터를 역할에 추가합니다. IAM에서의 태그 사용에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요.
11. (선택 사항)설명에 새 역할에 대한 설명을 입력합니다.
12. 역할을 검토한 후 역할 만들기를 선택합니다.

역할 목록에 UpdateData 역할이 표시됩니다.

이제 역할의 Amazon 리소스 이름(ARN), 즉 역할에 대한 고유 식별자를 얻어야 합니다. Originating 계정의 Developer 역할을 수정할 때 권한을 부여하거나 거부하려면 Destination 계정의 역할 ARN을 지정해야 합니다.

UpdateData의 ARN을 가져오는 방법

1. IAM 콘솔의 탐색 창에서 역할을 선택합니다.
2. 역할 목록에서 UpdateData 역할을 선택합니다.
3. 세부 정보 창의 요약 섹션에서 역할 ARN 값을 복사합니다.

   Destination 계정 ID는 999999999999입니다. 따라서 역할 ARN은 arn:aws:iam::999999999999:role/UpdateData입니다. Destination 계정의 실제 AWS 계정 ID를 제공해야 합니다.

이제 Destination 계정과 Originating 계정 간에 신뢰가 설정되었습니다. Destination 계정에서 Originating 계정을 신뢰할 수 있는 보안 주체로 식별하는 역할을 생성하여 이 작업을 수행했습니다. 그 밖에도 UpdateData 역할 전환 사용자의 권한까지 정의했습니다.

다음으로 Developer 역할의 권한을 수정합니다.

역할에 대한 액세스 권한 부여

이 시점에는 Analysts와 Developers 모두 Originating 계정의 데이터를 관리할 수 있는 권한을 갖고 있습니다. 역할 전환에 필요한 권한을 추가하려면 다음의 몇 단계를 거쳐야 합니다.

UpdateData 역할로 전환할 수 있도록 Developers 역할을 수정하는 방법

1. Originating 계정에 관리자로 로그인한 다음 IAM 콘솔을 엽니다.
2. 역할을 선택한 후 Developers를 선택합니다.
3. 권한(Permissions) 탭을 선택하고 권한 추가(Add permissions), 인라인 정책 생성(Create inline policy)을 차례로 선택합니다.
4. JSON 탭을 선택합니다.

5. 아래 정책문을 추가하여 Destination 계정의 UpdateData 역할에 대한 AssumeRole 작업을 허용합니다. 이때 Resource 요소의 *DESTINATION-ACCOUNT-ID*를 Destination 계정의 실제 AWS 계정 ID로 변경해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::DESTINATION-ACCOUNT-ID:role/UpdateData"
  }
}
```

   Allow는 Destination 계정에서 UpdateData 역할에 대한 Developers 그룹의 액세스를 명시적으로 허용하는 값입니다. 개발자라면 누구나 이 역할에 액세스할 수 있습니다.

6. 정책 검토를 선택합니다.
7. 이름에 **allow-assume-S3-role-in-destination**과 같은 사용자 이름을 입력합니다.
8. 정책 생성을 선택합니다.

대부분 환경에서 다음과 같은 절차는 필요하지 않을 수 있습니다. 하지만 PowerUserAccess 권한을 사용하는 경우에는 역할 전환을 할 수 있는 그룹이 있을 수도 있습니다. 다음 절차는 Analysts 그룹이 역할을 수임할 수 없도록 Analysts 그룹에 "Deny" 권한을 추가하는 방법을 보여줍니다. 이 절차가 필요 없는 환경인 경우 추가하지 않는 것이 좋습니다. "Deny" 권한은 전체 권한 구조를 관리하고 이해하기가 더 복잡하게 만듭니다. "Deny" 권한은 더 좋은 방법이 없을 때만 사용하십시오.

UpdateData 역할 수임 권한을 거부하도록 Analysts 역할을 수정하는 방법

1. 역할을 선택한 다음 Analysts를 선택합니다.
2. 권한(Permissions) 탭을 선택하고 권한 추가(Add permissions), 인라인 정책 생성(Create inline policy)을 차례로 선택합니다.
3. JSON 탭을 선택합니다.
4. 다음 정책을 추가하여 AssumeRole 역할의 UpdateData 작업을 거부합니다. 이때 Resource 요소의 *DESTINATION-ACCOUNT-ID*를 Destination 계정의 실제 AWS 계정 ID로 변경해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
```

```
    "Effect": "Deny",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::DESTINATION-ACCOUNT-ID:role/UpdateData"
  }
}
```

Deny는 Destination 계정에서 UpdateData 역할에 대한 Analysts 그룹의 액세스를 명시적으로 거부하는 값입니다. 이 역할에 액세스하려는 모든 분석가는 액세스 거부 메시지를 받습니다.

5. 정책 검토를 선택합니다.
6. **deny-assume-S3-role-in-destination**과 같은 이름을 입력합니다.
7. 정책 생성을 선택합니다.

Developers 역할은 이제 Destination 계정에서 UpdateData 역할을 사용할 수 있는 권한이 생겼습니다. Analysts 역할은 UpdateData 역할을 사용하지 못합니다.

다음으로 개발자인 David가 Destination 계정의 shared-container 버킷에 액세스하는 방법을 알아봅니다. David는 AWS Management Console, AWS CLI 또는 AWS API에서 버킷에 액세스할 수 있습니다.

역할을 전환하여 액세스 테스트

이 자습서의 첫 두 단계를 완료하면 Destination 계정의 리소스에 대한 액세스 권한을 부여하는 역할이 생깁니다. 또한 Originating 계정에 역할 하나와 해당 역할을 사용할 수 있는 사용자도 생깁니다. 이 단계에서는 AWS Management Console, AWS CLI 및 AWS API에서 해당 역할로 전환을 테스트하는 방법을 설명합니다.

IAM 역할 작업 시 발생할 수 있는 공통적 문제에 대한 도움을 받으려면 IAM 역할 문제 해결 섹션을 참조하세요.

역할 전환(콘솔)

David가 AWS Management Console에서 Destination 계정의 데이터를 업데이트해야 할 경우 역할 전환을 사용하면 됩니다. David가 계정 ID나 별칭 및 역할 이름을 지정하면 David의 권한이 해당 역할에 허용되는 권한으로 즉시 전환됩니다. 그런 다음 David는 콘솔을 사용하여 shared-container 버킷으로 작업할 수 있지만 Destination의 다른 리소스로는 작업할 수 없습니다. David가 이 역할을 사용하는 동안에는 Originating 계정에서 파워 유저 권한도 사용할 수 없습니다. 한 번에 하나의 권한 집합만 적용할 수 있기 때문입니다.

IAM는 David가 Switch Role(역할 전환) 페이지를 시작하는 데 사용할 수 있는 두 가지 방법을 제공합니다.

- David가 관리자로부터 미리 정의된 Switch Role 구성을 가리키는 링크를 받습니다. 이 링크는 역할 생성 마법사의 마지막 페이지 또는 크로스 계정 역할의 Role Summary(역할 요약) 페이지에서 관리자에게 제공됩니다. 이 링크를 선택하면 David는 계정 ID(Account ID) 및 역할 이름(Role name) 필드에 이미 정보가 채워진 역할 전환(Switch Role) 페이지로 이동됩니다. David는 역할 전환(Switch Roles) 버튼을 선택하기만 하면 됩니다.
- 관리자가 이메일로 링크를 보내는 대신 계정 ID 번호 및 역할 이름 값을 보냅니다. 역할을 전환하려면 David가 값을 수동으로 입력해야 합니다. 다음 절차에 이 내용이 잘 설명되어 있습니다.

역할 위임

1. David가 Originating 역할의 일반 사용자로 AWS Management Console에 로그인합니다.
2. 관리자가 이메일로 보낸 링크를 선택합니다. 계정 ID나 별칭 및 역할 이름 정보가 이미 채워진 역할 전환(Switch Role) 페이지가 David에게 표시됩니다.

   - 또는 -

   David는 탐색 모음에서 자신의 이름(아이덴티티(Identity) 메뉴)을 선택한 후 역할 전환(Switch Roles)을 선택합니다.

   David가 이 방법으로 처음 Switch Role(역할 전환) 페이지 액세스를 시도하는 것이라면 첫 실행 Switch Role(역할 전환) 페이지가 표시됩니다. 이 페이지에는 역할 전환을 통해 사용자가 여러 AWS 계정의 리소스를 관리하는 방법에 대한 추가 정보가 제공됩니다. 이 절차의 나머지 부분을 완료하려면 David가 이 페이지에서 Switch Role(역할 전환)을 선택해야 합니다.
3. 다음으로, 해당 역할에 액세스하기 위해 David는 수동으로 Destination 계정 ID 번호(999999999999)와 역할 이름(`UpdateData`)을 입력해야 합니다.

   또한 David는 IAM에서 현재 활성 상태인 역할 및 관련 권한을 모니터링하려고 합니다. 이 정보를 추적하려면 표시 이름(Display Name) 텍스트 상자에 `Destination`을 입력하고 빨간색 옵션을 선택한 다음 역할 전환(Switch Role)을 선택합니다.
4. 이제 David는 Amazon S3 콘솔을 사용하여 Amazon S3 버킷으로 작업하거나 `UpdateData` 역할에 권한이 있는 다른 모든 리소스로 작업할 수 있습니다.
5. 완료되면 David는 원래 권한으로 돌아갈 수 있습니다. 이를 위해 탐색 모음에서 Destination 역할 표시 이름을 선택한 다음 Back to David @ 111111111111을 선택합니다.

6. 다음에 David가 역할을 전환하려고 탐색 모음에서 자격 증명 메뉴를 선택하면 지난 번에 사용한 Destination 항목이 표시되는 것을 볼 수 있습니다. 계정 ID와 역할 이름을 다시 입력할 필요 없이 해당 항목을 선택하기만 하면 즉시 역할이 전환됩니다.

역할 전환(AWS CLI)

David가 명령줄에서 Destination 환경으로 작업해야 할 경우 AWS CLI을 사용하면 됩니다. David는 `aws sts assume-role` 명령을 실행하고 역할 ARN을 전달하여 해당 역할에 대한 임시 보안 자격 증명을 얻습니다. 그런 다음 후속 AWS CLI 명령이 해당 역할의 권한을 사용하여 작동하도록 환경 변수에서 해당 자격 증명을 구성합니다. 한 번에 한 가지 권한 세트만 적용될 수 있기 때문에 David가 해당 역할을 사용하는 동안에는 Originating 계정의 파워 유저 권한을 사용할 수 없습니다.

모든 액세스 키와 토큰은 예시일 뿐이며 표시된 대로 사용할 수 없습니다. 라이브 환경의 적절한 값으로 바꾸세요.

역할 위임

1. David가 명령 프롬프트 창을 열고 다음 명령을 실행하여 AWS CLI 클라이언트가 작동하는지 확인합니다.

   ```
   aws help
   ```

   > **Note**
   >
   > David의 기본 환경에는 `David` 명령으로 만든 기본 프로필의 `aws configure` 사용자 자격 증명이 사용됩니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface 구성을 참조하세요.

2. David가 Destination 계정의 `UpdateData` 역할로 전환하기 위해 다음 명령을 실행하여 역할 전환 프로세스를 시작합니다. David는 해당 역할을 만든 관리자에게서 역할 ARN을 받았습니다. 이 명령을 실행하려면 세션 이름도 제공해야 합니다. 원하는 아무 텍스트나 선택할 수 있습니다.

   ```
   aws sts assume-role --role-arn "arn:aws:iam::999999999999:role/UpdateData" --role-session-name "David-ProdUpdate"
   ```

   다음 내용이 출력됩니다.

   ```
   {
   ```

```
    "Credentials": {
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "SessionToken": "AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPLEeYjs1M2FUIgIJx9tQqNMBEXAMPLE
CvSRyh0FW7jEXAMPLEW+vE/7s1HRpXviG7b+qYf4nD00EXAMPLEmj4wxS04L/
uZEXAMPLECihzFB5lTYLto9dyBgSDy
EXAMPLE9/
g7QRUhZp4bqbEXAMPLENwGPyOj59pFA4lNKCIkVgkREXAMPLEjlzxQ7y52gekeVEXAMPLEDiB9ST3UuysgsKdEXAMPLE1TVastU1A0SKFEXAMPLEiywCC/Cs8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP
+4eZScEXAMPLEsnf87e
NhyDHq6ikBQ==",
        "Expiration": "2014-12-11T23:08:07Z",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
    }
}
```

3. 출력의 Credentials 섹션에 David에게 필요한 세 가지 항목이 표시됩니다.

   - `AccessKeyId`
   - `SecretAccessKey`
   - `SessionToken`

   David는 후속 호출 시 이러한 파라미터를 사용하도록 AWS CLI 환경을 구성해야 합니다. 자격 증명을 구성하는 다양한 방법에 대한 자세한 정보는 AWS Command Line Interface 구성을 참조하십시오. `aws configure` 명령은 세션 토큰 캡처를 지원하지 않기 때문에 사용할 수 없습니다. 하지만 구성 파일에 정보를 수동으로 입력할 수 있습니다. 이는 비교적 만료 시간이 짧은 임시 자격 증명이기 때문에 현재 명령줄 세션의 환경에 추가하는 것이 가장 쉽습니다.

4. 세 값을 환경에 추가하기 위해 David는 이전 단계의 출력을 잘라내어 다음 명령에 붙여 넣습니다. 세션 토큰 출력의 줄 바꿈 문제를 해결하기 위해 간단한 텍스트 편집기에서 텍스트 출력을 잘라내어 붙여 넣을 수 있습니다. 여기서는 명확성을 위해 줄을 바꾸어 표시했지만 긴 문자열 하나로 추가해야 합니다.

   다음 예시는 Windows 환경에 표시된 명령을 나타내며, 여기서 "set"은 환경 변수를 생성하라는 명령입니다. Linux 또는 macOS 컴퓨터에서는 "export" 명령을 대신 사용할 수 있습니다. 예시의 나머지 부분은 세 환경에서 모두 유효합니다.

   Windows Powershell용 도구를 사용하는 방법에 대한 자세한 내용은 IAM 역할로 전환(Tools for Windows PowerShell)을 참조하세요.

```
set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
set AWS_SESSION_TOKEN=AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPLEeYjs1M2FUIgIJx9tQqNMBEXAMPLECvS
Ryh0FW7jEXAMPLEW+vE/7s1HRpXviG7b+qYf4nD00EXAMPLEmj4wxS04L/
uZEXAMPLECihzFB5lTYLto9dyBgSDyEXA
MPLEKEY9/
g7QRUhZp4bqbEXAMPLENwGPyOj59pFA4lNKCIkVgkREXAMPLEjlzxQ7y52gekeVEXAMPLEDiB9ST3UusKd
EXAMPLE1TVastU1A0SKFEXAMPLEiywCC/Cs8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP
+4eZScEXAMPLENhykxiHen
DHq6ikBQ==
```

이 시점에서 모든 후속 명령은 해당 자격 증명으로 식별되는 역할의 권한에 따라 실행됩니다. David의 경우 `UpdateData` 역할입니다.

> **Important**
>
> AWS CLI에서 유지되는 파일에 자주 사용되는 구성 설정과 보안 인증을 저장할 수 있습니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 기존 구성 및 보안 인증 파일 사용을 참조하세요.

5. 명령을 실행하여 Destination 계정의 리소스에 액세스합니다. 이 예시에서 David는 단순히 다음 명령을 사용하여 S3 버킷의 콘텐츠를 나열합니다.

```
aws s3 ls s3://shared-container
```

Amazon S3 버킷 이름은 범용 고유 이름이기 때문에 해당 버킷을 소유하는 계정 ID를 지정할 필요가 없습니다. 다른 AWS 서비스의 리소스에 액세스하려면 해당 서비스의 AWS CLI 설명서에서 해당 리소스를 참조하는 데 필요한 명령과 구문을 참조하세요.

AssumeRole(AWS API)사용

David가 코드에서 Destination 계정을 업데이트해야 하는 경우 `AssumeRole`을 호출하여 `UpdateData` 역할을 수임합니다. 이 호출은 Destination 계정에서 David가 `shared-container` 버킷에 액세스하기 위해 사용할 수 있는 임시 보안 인증 정보를 반환합니다. David는 해당 자격 증명을 사용하여 `shared-container` 버킷을 업데이트하는 API 호출을 실행할 수 있습니다. 그러나 David는 Originating 계정의 파워 유저 권한이 있더라도 Destination 계정의 다른 리소스에 액세스하는 API 호출을 실행할 수 없습니다.

역할 위임

1. David가 애플리케이션의 일부로 `AssumeRole`을 호출합니다. `UpdateData` ARN인 `arn:aws:iam::999999999999:role/UpdateData`을 지정해야 합니다.

   `AssumeRole` 호출의 응답에는 `AccessKeyId` 및 `SecretAccessKey`가 있는 임시 자격 증명이 포함됩니다. 또한 자격 증명이 만료되고 새 자격 증명을 요청해야 하는 시점을 나타내는 `Expiration` 시간이 포함됩니다. AWS SDK로 역할 함께 묶기를 설정하면 많은 보안 인증 정보 공급자가 만료 전에 보안 인증 정보를 자동으로 새로 고칩니다.
2. David가 임시 자격 증명을 사용하여 `s3:PutObject` 버킷을 업데이트하는 `shared-container` 호출을 실행합니다. 이때 자격 증명을 `AuthParams` 파라미터로 API 호출에 전달합니다. 임시 역할 보안 인증 정보에는 `shared-container` 버킷에 대한 읽기 및 쓰기 권한만 있기 때문에 Destination 계정의 다른 모든 작업은 거부됩니다.

코드 샘플(Python 사용)은 IAM 역할로 전환(AWS API) 단원을 참조하십시오.

추가 리소스

다음 리소스는 이 자습서의 주제에 대해 자세히 알아보는 데 도움이 됩니다.

- IAM 사용자에 대한 자세한 내용은 IAM 자격 증명(사용자, 사용자 그룹 및 역할) 섹션을 참조하세요.
- Amazon S3 버킷에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Create a Bucket(버킷 생성)을 참조하세요.
- 해당 신뢰 영역(신뢰할 수 있는 조직 또는 계정) 외의 계정 내 보안 주체가 역할을 수임하는 권한이 있는지 자세히 알고 싶다면, IAM Access Analyzer란 무엇일까요?를 참조하세요.

요약

크로스 계정 API 액세스 자습서를 완료했습니다. 다른 계정과 신뢰 관계를 설정하기 위한 역할을 만들고 신뢰할 수 있는 주체가 수행할 수 있는 작업을 정의했습니다. 그런 다음 해당 역할에 액세스할 수 있는 IAM 사용자를 제어하도록 역할 정책을 수정했습니다. 그 결과 Originating 계정의 개발자가 임시 보안 인증 정보를 사용하여 Destination 계정에서 `shared-container` 버킷을 업데이트할 수 있습니다.

IAM 튜토리얼: 첫 번째 고객 관리형 정책 만들기 및 연결

이 튜토리얼에서는 AWS Management Console을 사용하여 고객 관리형 정책을 만든 다음 이 정책을 AWS 계정의 IAM 사용자에게 연결합니다. 여기서 생성하는 정책은 IAM 테스트 사용자가 읽기 전용 권한으로 AWS Management Console에 직접 로그인할 수 있도록 허용합니다.

이 워크플로우는 세 가지 기본 단계로 이루어집니다.

1단계: 정책 생성

기본적으로 IAM 사용자에게는 아무런 권한이 없습니다. 관리자가 허용하지 않는 한, 이들 사용자가 AWS 관리 콘솔에 액세스하거나 그 안에서 데이터를 관리할 수 없습니다. 이 단계에서는 연결된 사용자가 콘솔에 로그인할 수 있도록 허용하는 고객 관리형 정책을 생성합니다.

2단계: 정책 연결

정책을 사용자에 연결할 경우 이 사용자는 해당 정책과 관련된 액세스 권한을 모두 상속받습니다. 이 단계에서는 새 정책을 테스트 사용자에 연결합니다.

3단계: 사용자 액세스 테스트

정책을 연결하고 나면 해당 사용자로 로그인하여 정책을 테스트할 수 있습니다.

사전 조건

이 자습서의 단계를 수행하려면 다음이 준비되어 있어야 합니다.

- 관리 권한을 가진 IAM 사용자로 로그인할 수 있는 AWS 계정.
- 다음과 같이 할당된 권한 또는 그룹 멤버십이 없는 테스트 IAM 사용자.

사용자 이름	그룹	권한
PolicyUser	<없음>	<없음>

1단계: 정책 생성

이 단계에서는 연결된 사용자가 IAM 데이터에 대한 읽기 전용 권한으로 AWS Management Console에 로그인할 수 있도록 허용하는 고객 관리형 정책을 생성합니다.

테스트 사용자에 대한 정책을 생성하려면

1. 관리자 권한이 있는 사용자로 https://console.aws.amazon.com/iam/의 IAM 콘솔에 로그인합니다.
2. 탐색 창에서 정책을 선택합니다.
3. 콘텐츠 창에서 정책 생성을 선택합니다.
4. JSON 옵션을 선택하고 다음 JSON 정책 문서에서 텍스트를 복사합니다. 이 텍스트를 JSON 텍스트 상자에 붙여 넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [ {
        "Effect": "Allow",
        "Action": [
            "iam:GenerateCredentialReport",
            "iam:Get*",
            "iam:List*"
        ],
        "Resource": "*"
    } ]
}
```

5. 정책 검증 동안 생성된 모든 보안 경고, 오류 또는 일반 경고를 해결하고 다음을 선택합니다.

> **Note**
>
> 언제든지 시각적 편집기 옵션과 JSON 편집기 옵션을 서로 전환할 수 있습니다. 그러나 변경을 수행하거나 시각적 편집기 탭에서 정책 검토를 선택한 경우 IAM은 시각적 편집기에 최적화되도록 정책을 재구성할 수 있습니다. 자세한 내용은 정책 재구성 단원을 참조하십시오.

6. 검토 및 생성 페이지에서 정책 이름에 **`UsersReadOnlyAccessToIAMConsole`**을 입력합니다. 정책이 부여한 권한을 검토한 다음 정책 생성을 선택하여 작업을 저장합니다.

   새로운 정책이 관리형 정책 목록에 나타나며 연결 준비가 완료됩니다.

2단계: 정책 연결

다음에는 방금 생성한 테스트 IAM 사용자에 정책을 연결합니다.

테스트 사용자에 정책을 연결하려면

1. IAM 콘솔의 탐색 창에서 정책(Policies)을 선택합니다.
2. 정책 목록의 맨 위에 있는 검색 상자에서 해당 정책이 표시될 때까지 **`UsersReadOnlyAccesstoIAMConsole`** 입력을 시작합니다. 그런 다음 목록에서 UsersReadOnlyAccessToIAMConsole 옆에 있는 라디오 버튼을 선택합니다.
3. 작업(Actions) 버튼을 선택한 후 연결(Attach)을 선택합니다.
4. IAM 엔터티에서 사용자를 필터링할 옵션을 선택합니다.
5. 검색 상자에서 해당 사용자가 목록에 표시될 때까지 **`PolicyUser`** 입력을 시작합니다. 그런 다음 목록에서 해당 사용자 옆의 확인란을 선택합니다.
6. 정책 연결을 선택합니다.

이제 IAM 테스트 사용자에 정책을 연결했습니다. 즉, 이 사용자가 IAM 콘솔에 읽기 전용으로 액세스할 수 있습니다.

3단계: 사용자 액세스 테스트

이 자습서에서는 사용자가 어떤 경험을 하게 되는지 볼 수 있도록 테스트 사용자로 로그인하여 액세스를 테스트할 것을 권장합니다.

테스트 사용자로 로그인하여 액세스 권한을 테스트하려면

1. IAM 콘솔(https://console.aws.amazon.com/iam/)에 `PolicyUser` 테스트 사용자로 로그인합니다.
2. 콘솔에서 각 페이지를 탐색하고 새 사용자 또는 그룹을 생성해 봅니다. `PolicyUser`는 데이터를 표시할 수는 있지만 IAM 데이터를 생성하거나 수정할 수는 없습니다.

관련 리소스

관련 내용은 다음 리소스를 참조하십시오.

- 관리형 정책과 인라인 정책
- AWS Management Console에 대한 IAM 사용자 액세스 제어

요약

이제 고객 관리형 정책을 생성 및 연결하는 데 필요한 모든 단계를 성공적으로 완료했습니다. 테스트 계정으로 IAM 콘솔에 로그인하여 사용자의 환경을 확인할 수 있습니다.

IAM 튜토리얼: 태그를 기반으로 AWS 리소스에 액세스할 수 있는 권한 정의

ABAC(속성 기반 액세스 제어)는 속성에 근거하여 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 엔터티(사용자 또는 역할)와 AWS 리소스에 태그를 연결할 수 있습니다. 태그 조건 키를 사용하여 해당 태그를 기반으로 보안 주체에 권한을 부여하는 정책을 정의할 수 있습니다. 태그를 사용하여 AWS 리소스에 대한 액세스를 제어하면 AWS 정책에 대한 변경 사항이 줄어들면서 팀과 리소스가 성장할 수 있습니다. ABAC 정책은 각 개별 리소스를 나열해야 하는 기존 AWS 정책보다 유연합니다. ABAC에 대한 자세한 내용 및 기존 정책과 비교할 때의 이점은 ABAC 권한 부여를 통한 속성 기반 권한 정의 단원을 참조하십시오.

> **Note**
>
> 각 세션 태그에는 단일 값을 전달해야 합니다. AWS Security Token Service에서는 다중 값 세션 태그를 지원하지 않습니다.

주제

- IAM 튜토리얼: ABAC에 SAML 세션 태그 사용

자습서 개요

이 튜토리얼에서는 보안 주체 태그가 있는 IAM 역할이 일치하는 태그가 있는 리소스에 액세스할 수 있도록 허용하는 정책을 생성하고 테스트하는 방법을 보여줍니다. 보안 주체가 AWS에 요청하면 보안 주체와 리소스 태그가 일치하는지 여부에 따라 권한이 부여됩니다. 이 전략을 통해 개인이 자신의 작업에 필요한 AWS 리소스만 보거나 편집할 수 있습니다.

시나리오

Example Corporation이라는 대기업의 수석 개발자이자 숙련된 IAM 관리자가 있다고 가정해 보겠습니다. 이 관리자는 IAM 사용자, 역할 및 정책을 생성하고 관리하는 데 익숙합니다. 개발 엔지니어와 품질 보증 팀 멤버가 필요한 리소스에 액세스할 수 있도록 해야 하며 기업의 성장에 따라 확장 가능한 전략도 준비해야 합니다.

AWS 리소스 태그와 IAM 역할 보안 주체 태그를 사용하여 이를 지원하는 서비스에 대한 ABAC 전략을 구현하도록 선택합니다(AWS Secrets Manager로 시작). 태그를 기반으로 권한 부여를 지원하는 서비스에 대한 자세한 내용은 AWS IAM으로 작업하는 서비스 단원을 참조하십시오. 각 서비스의 작업 및 리소스와 함께 정책에서 사용할 수 있는 태그 지정 조건 키에 대한 자세한 내용은 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요. 세션 태그를 AWS에 전달하도록 SAML 기반 또는 웹 자격 증명 공급자를 구성할 수 있습니다. 직원이 AWS에 연동되면 해당 속성이 AWS의 결과 보안 주체에 적용됩니다. 그런 다음 ABAC를 사용하여 이러한 속성에 따라 권한을 허용하거나 거부할 수 있습니다. SAML 페더레이션 자격 증명과 함께 세션 태그를 사용하는 것이 이 자습서와 어떻게 다른지 알아보려면 IAM 튜토리얼: ABAC에 SAML 세션 태그 사용 단원을 참조하십시오.

엔지니어링 및 품질 보증 팀 멤버는 Pegasus 또는 Unicorn 프로젝트에 참여하고 있습니다. 다음 세 글자로 된 프로젝트 및 팀 태그 값을 선택합니다.

- Pegasus 프로젝트에 대해 `access-project` = `peg`
- Unicorn 프로젝트에 대해 `access-project` = `uni`
- 엔지니어링 팀에 대해 `access-team` = `eng`
- 품질 보증 팀에 대해 `access-team` = `qas`

또한 사용자 지정 AWS 결제 보고서를 활성화할 때 `cost-center` 비용 할당 태그가 필요하도록 선택할 수 있습니다. 자세한 내용은 AWS Billing and Cost Management 사용 설명서의 비용 할당 태그 사용을 참조하세요.

주요 의사 결정 요약

- 직원은 IAM 사용자 자격 증명으로 로그인한 다음 팀 및 프로젝트의 IAM 역할을 맡습니다. 회사에 자체 자격 증명 시스템이 있는 경우 직원이 IAM 사용자 없이 역할을 맡을 수 있도록 연동을 설정할 수 있습니다. 자세한 내용은 IAM 튜토리얼: ABAC에 SAML 세션 태그 사용 단원을 참조하십시오.
- 동일한 정책이 모든 역할에 연결됩니다. 작업은 태그에 따라 허용되거나 거부됩니다.
- 직원은 자신의 역할에 적용되는 리소스에 동일한 태그를 연결하는 경우에만 새 리소스를 생성할 수 있습니다. 이렇게 하면 직원이 리소스를 생성한 후 해당 리소스를 볼 수 있습니다. 관리자는 더 이상 새 리소스의 ARN으로 정책을 업데이트할 필요가 없습니다.
- 직원은 프로젝트와 관계없이 자신의 팀이 소유한 리소스를 읽을 수 있습니다.
- 직원은 자신의 팀과 프로젝트가 소유한 리소스를 업데이트하고 삭제할 수 있습니다.
- IAM 관리자는 새 프로젝트에 대한 새 역할을 추가할 수 있습니다. 해당 역할에 대한 액세스를 허용하도록 새 IAM 사용자를 생성하고 태그를 지정할 수 있습니다. 관리자는 새 프로젝트 또는 팀 멤버를 지원하기 위해 정책을 편집할 필요가 없습니다.

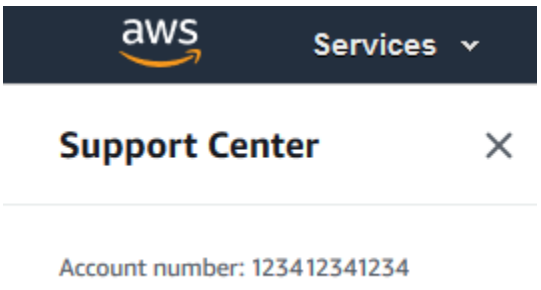
이 자습서에서는 각 리소스에 태그를 지정하고 프로젝트 역할에 태그를 지정하고 역할에 정책을 추가하여 앞에서 설명한 동작을 허용합니다. 결과 정책은 동일한 프로젝트 및 팀 태그로 태그가 지정된 리소스에 대한 역할 `Create`, `Read`, `Update` 및 `Delete` 액세스를 허용합니다. 또한 이 정책은 동일한 팀으로 태그가 지정된 리소스에 대해 프로젝트 간 `Read` 액세스를 허용합니다.

사전 조건

이 자습서의 단계를 수행하려면 다음이 준비되어 있어야 합니다.

- 관리 권한을 가진 사용자로 로그인할 수 있는 AWS 계정.
- 3단계에서 역할을 생성하는 데 사용한 12자리 계정 ID입니다.

  AWS Management Console을 사용하여 AWS 계정 ID 번호를 찾으려면 오른쪽 상단에 있는 탐색 모음에서 지원을 선택한 후 지원 센터를 선택합니다. 계정 번호(ID)가 왼쪽 탐색 창에 나타납니다.



- AWS Management Console에서 IAM 사용자, 역할 및 정책을 생성 및 편집해 본 경험. 그러나 IAM 관리 프로세스를 기억해야 하는 경우를 위해 이 튜토리얼에서는 단계별 지침을 볼 수 있는 링크를 제공합니다.

1단계: 테스트 사용자 생성

테스트를 위해 동일한 태그를 사용하여 역할을 맡을 권한이 있는 IAM 사용자 4명을 생성합니다. 이렇게 하면 팀에 사용자를 더 쉽게 추가할 수 있습니다. 사용자에게 태그를 지정하면 올바른 역할을 맡을 수 있는 액세스 권한이 자동으로 부여됩니다. 사용자가 하나의 프로젝트와 팀에서만 작업하는 경우 역할의 신뢰 정책에 사용자를 추가할 필요가 없습니다.

1. 다음과 같이 `access-assume-role`이라는 고객 관리형 정책을 생성합니다. JSON 정책 생성에 대한 자세한 내용은 IAM 정책 생성 단원을 참조하십시오.

   ABAC 정책: 사용자 및 역할 태그가 일치하는 경우에만 모든 ABAC 역할 수임

   다음 정책은 사용자가 `access-` 이름 접두사가 있는 계정의 모든 역할을 맡을 수 있도록 허용합니다. 역할에는 사용자와 동일한 프로젝트, 팀 및 비용 센터 태그가 지정되어 있어야 합니다.

   이 정책을 사용하려면 기울임꼴 자리 표시자 텍스트를 계정 정보로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TutorialAssumeRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::account-ID-without-hyphens:role/access-*",
            "Condition": {
                "StringEquals": {
                    "iam:ResourceTag/access-project": "${aws:PrincipalTag/access-
project}",
                    "iam:ResourceTag/access-team": "${aws:PrincipalTag/access-
team}",
                    "iam:ResourceTag/cost-center": "${aws:PrincipalTag/cost-
center}"
                }
            }
        }
    ]
```

```
}
```

이 자습서를 많은 사용자로 확장하기 위해 정책을 그룹에 연결하고 각 사용자를 그룹에 추가할 수 있습니다. 자세한 내용은 IAM 사용자 그룹 생성 및 IAM 사용자 그룹의 사용자 편집 단원을 참조하세요.

2. 다음 IAM 사용자를 생성하고, `access-assume-role` 권한 정책을 연결합니다. AWS Management Console에 대한 사용자 액세스 제공을 선택하고 다음 태그를 추가해야 합니다. 새 사용자 생성 및 태그 지정에 대한 자세한 내용은 IAM 사용자 생성(콘솔) 단원을 참조하십시오.

사용자 이름	사용자 태그 키	사용자 태그 값
access-Arnav-peg-eng	access-project	peg
	access-team	eng
	cost-center	987654
access-Mary-peg-qas	access-project	peg
	access-team	qas
	cost-center	987654
access-Saanvi-uni-eng	access-project	uni
	access-team	eng
	cost-center	123456
access-Carlos-uni-qas	access-project	uni
	access-team	qas
	cost-center	123456

2단계: ABAC 정책 생성

다음과 같이 **access-same-project-team**이라는 정책을 생성합니다. 이후 단계에서 이 정책을 역할에 추가합니다. JSON 정책 생성에 대한 자세한 내용은 IAM 정책 생성 단원을 참조하십시오.

이 자습서에 적용할 수 있는 추가 정책은 다음 페이지를 참조하십시오.

- IAM 보안 주체에 대한 액세스 제어
- Amazon EC2: 프로그래밍 방식으로 콘솔에서 사용자가 태그를 지정한 EC2 인스턴스를 시작 또는 중지할 수 있도록 허용
- EC2: 일치하는 보안 주체 및 리소스 태그를 기반으로 인스턴스 시작 또는 중지
- EC2: 태그를 기반으로 인스턴스 시작 또는 중지
- IAM: 특정 태그가 있는 역할 수임

ABAC 정책: 보안 주체 및 리소스 태그가 일치하는 경우에만 Secrets Manager 리소스 액세스

다음 정책은 보안 주체와 동일한 키-값 페어로 리소스에 태그가 지정된 경우에만 보안 주체가 리소스를 생성, 읽기, 편집 및 삭제할 수 있도록 허용합니다. 보안 주체가 리소스를 생성할 때 보안 주체의 태그와 일치하는 값을 가진 `access-project`, `access-team` 및 `cost-center` 태그를 추가해야 합니다. 이 정책에서는 선택 사항인 `Name` 또는 `OwnedBy` 태그를 추가할 수도 있습니다.

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Sid": "AllActionsSecretsManagerSameProjectSameTeam",
         "Effect": "Allow",
         "Action": "secretsmanager:*",
         "Resource": "*",
         "Condition": {
             "StringEquals": {
                 "aws:ResourceTag/access-project": "${aws:PrincipalTag/access-
project}",
                 "aws:ResourceTag/access-team": "${aws:PrincipalTag/access-team}",
                 "aws:ResourceTag/cost-center": "${aws:PrincipalTag/cost-center}"
             },
             "ForAllValues:StringEquals": {
                 "aws:TagKeys": [
                     "access-project",
                     "access-team",
```

```
                    "cost-center",
                    "Name",
                    "OwnedBy"
                ]
            },
            "StringEqualsIfExists": {
                "aws:RequestTag/access-project": "${aws:PrincipalTag/access-project}",
                "aws:RequestTag/access-team": "${aws:PrincipalTag/access-team}",
                "aws:RequestTag/cost-center": "${aws:PrincipalTag/cost-center}"
            }
        }
    },
    {
        "Sid": "AllResourcesSecretsManagerNoTags",
        "Effect": "Allow",
        "Action": [
            "secretsmanager:GetRandomPassword",
            "secretsmanager:ListSecrets"
        ],
        "Resource": "*"
    },
    {
        "Sid": "ReadSecretsManagerSameTeam",
        "Effect": "Allow",
        "Action": [
            "secretsmanager:Describe*",
            "secretsmanager:Get*",
            "secretsmanager:List*"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/access-team": "${aws:PrincipalTag/access-team}"
            }
        }
    },
    {
        "Sid": "DenyUntagSecretsManagerReservedTags",
        "Effect": "Deny",
        "Action": "secretsmanager:UntagResource",
        "Resource": "*",
        "Condition": {
            "ForAnyValue:StringLike": {
                "aws:TagKeys": "access-*"
```

```
                }
            }
        },
        {
            "Sid": "DenyPermissionsManagement",
            "Effect": "Deny",
            "Action": "secretsmanager:*Policy",
            "Resource": "*"
        }
    ]
}
```

이 정책이 하는 일은 무엇입니까?

- `AllActionsSecretsManagerSameProjectSameTeam` 문은 리소스 태그가 보안 주체 태그와 일치하는 경우에만 모든 관련 리소스에 대해 이 서비스의 모든 작업을 허용합니다. 정책에 `"Action": "secretsmanager:*"`를 추가하면 Secrets Manager가 커질수록 정책도 커집니다. Secrets Manager에서 새 API 작업을 추가하는 경우 문에 해당 작업을 추가할 필요가 없습니다. 이 문은 세 가지 조건 블록을 사용하여 ABAC를 구현합니다. 세 블록 모두 true를 반환하는 경우에만 요청이 허용됩니다.
  - 지정된 태그 키가 리소스에 있고 해당 값이 보안 주체의 태그와 일치하는 경우 이 문의 첫 번째 조건 블록은 true를 반환합니다. 이 블록은 일치하지 않는 태그 또는 리소스 태그 지정을 지원하지 않는 작업에 대해 false를 반환합니다. 이 블록에서 허용되지 않는 작업에 대한 자세한 내용은 AWS Secrets Manager에 대한 작업, 리소스 및 조건 키를 참조하십시오 . 이 페이지에서는 비밀 리소스 유형에 대해 수행된 작업이 `secretsmanager:ResourceTag/tag-key` 조건 키를 지원한다는 것을 보여 줍니다. 일부 Secrets Manager 작업에서는 `GetRandomPassword` 및 `ListSecrets`를 포함하여 해당 해당 리소스 유형을 지원하지 않습니다. 이러한 작업을 허용하려면 추가 문을 생성해야 합니다.
  - 두 번째 조건 블록은 요청에 전달된 모든 태그 키가 지정된 목록에 포함된 경우 true를 반환합니다. 이 작업은 `StringEquals` 조건 연산자와 함께 `ForAllValues`를 사용하여 수행됩니다. 키 또는 키 세트의 일부가 전달되지 않으면 조건이 true를 반환합니다. 이 경우 요청에서 태그 전달을 허용하지 않는 `Get*` 작업을 사용할 수 있습니다. 요청자가 목록에 없는 태그 키를 포함하는 경우 조건은 false를 반환합니다. 요청에 전달되는 모든 태그 키가 이 목록의 멤버와 일치해야 합니다. 자세한 내용은 다중 값 컨텍스트 키 단원을 참조하십시오.
  - 세 번째 조건 블록은 요청이 태그 전달을 지원하고, 세 개의 태그가 모두 존재하고, 보안 주체 태그 값과 일치하는 경우 true를 반환합니다. 요청이 태그 전달을 지원하지 않는 경우에도 이 블록은 true를 반환합니다. 그 이유는 조건 연산자의 `...IfExists` 때문입니다. 지원하는 작업 중에 전달된 태그가 없거나 태그 키 및 값이 일치하지 않으면 블록이 false를 반환합니다.

- `AllResourcesSecretsManagerNoTags` 문은 첫 번째 문에서 허용되지 않는 `GetRandomPassword` 및 `ListSecrets` 작업을 허용합니다.
- `ReadSecretsManagerSameTeam` 문은 보안 주체가 리소스와 동일한 액세스 팀 태그로 태그가 지정된 경우 읽기 전용 작업을 허용합니다. 이는 프로젝트 또는 비용 센터 태그와 관계없이 허용됩니다.
- `DenyUntagSecretsManagerReservedTags` 문은 Secrets Manager에서 “access-”로 시작하는 키가 있는 태그를 제거하라는 요청을 거부합니다. 이러한 태그는 리소스에 대한 액세스를 제어하는 데 사용되므로 태그를 제거하면 권한이 제거될 수 있습니다.
- `DenyPermissionsManagement` 문은 Secrets Manager 리소스 기반 정책을 생성, 편집 또는 삭제할 수 있는 권한을 거부합니다. 이러한 정책을 사용하여 비밀의 권한을 변경할 수 있습니다.

> **Important**
>
> 이 정책은 전략을 사용하여 서비스에 대한 모든 작업을 허용하지만 권한 변경 작업을 명시적으로 거부합니다. 작업을 거부하면 보안 주체가 해당 작업을 수행할 수 있도록 허용하는 다른 정책을 재정의합니다. 이로 인해 의도하지 않은 결과가 발생할 수 있습니다. 명시적 거부는 해당 작업을 허용해야 하는 상황이 없는 경우에만 사용하는 것이 가장 좋습니다. 그렇지 않은 경우 개별 작업 목록을 허용하고 원치 않는 작업이 기본적으로 거부됩니다.

3단계: 역할 생성

다음 IAM 역할을 생성하고 이전 단계에서 생성한 **`access-same-project-team`** 정책을 연결합니다. IAM 역할 생성에 대한 자세한 내용은 IAM 사용자에게 권한을 위임할 역할 생성 섹션을 참조하세요. IAM 사용자 및 역할 대신 연동을 사용하도록 선택한 경우 IAM 튜토리얼: ABAC에 SAML 세션 태그 사용 섹션을 참조하세요.

직무	역할 이름	역할 태그	역할 설명
프로젝트 Pegasus 엔지니어링	access-peg-engineering	access-project = peg access-team = eng cost-center = 987654	엔지니어는 모든 엔지니어링 리소스를 읽고 Pegasus 엔지니어링 리소스를 생성 및 관리할 수 있습니다.

직무	역할 이름	역할 태그	역할 설명
프로젝트 Pegasus 품질 보증	access-peg-quality-assurance	access-project = peg access-team = qas cost-center = 987654	QA 팀은 모든 QA 리소스를 읽고 모든 Pegasus QA 리소스를 생성 및 관리할 수 있습니다.
프로젝트 Unicorn 엔지니어링	access-uni-engineering	access-project= uni access-team = eng cost-center = 123456	엔지니어는 모든 엔지니어링 리소스를 읽고 Unicorn 엔지니어링 리소스를 생성 및 관리할 수 있습니다.
프로젝트 Unicorn 품질 보증	access-uni-quality-assurance	access-project = uni access-team = qas cost-center = 123456	QA 팀은 모든 QA 리소스를 읽고 모든 Unicorn QA 리소스를 생성 및 관리할 수 있습니다.

4단계: 비밀 생성 테스트

역할에 연결된 권한 정책을 통해 직원이 비밀을 생성할 수 있습니다. 이 작업은 비밀에 프로젝트, 팀 및 비용 센터 태그가 지정된 경우에만 허용됩니다. Secrets Manager에서 사용자로 로그인하고, 올바른 역할을 맡고, 활동을 테스트하여 권한이 예상대로 작동하는지 확인합니다.

필수 태그를 사용하거나 사용하지 않고 비밀 생성을 테스트하려면

1. 기본 브라우저 창에서 관리자 사용자로 로그인한 상태로 유지되므로 IAM에서 사용자, 역할 및 정책을 검토할 수 있습니다. 테스트를 위해 브라우저 익명 창 또는 별도의 브라우저를 사용하십시오. 이제 `access-Arnav-peg-eng` IAM 사용자로 로그인하고 https://console.aws.amazon.com/secretsmanager/에서 Secrets Manager 콘솔을 엽니다.
2. `access-uni-engineering` 역할로의 전환을 시도합니다.

access-Arnav-peg-eng 사용자와 access-uni-engineering 역할의 access-project 및 cost-center 태그 값이 일치하지 않기 때문에 이 작업은 실패합니다.

AWS Management Console에서의 역할 전환에 대한 자세한 내용은 역할로 전환(콘솔)의 내용을 참조하세요.

3. access-peg-engineering 역할로 전환합니다.
4. 다음 정보를 사용하여 새 비밀을 저장합니다. 비밀을 저장하는 방법에 대한 자세한 내용을 알아보려면 AWS Secrets Manager 사용 설명서의 Creating a Basic Secret(기본 보안 암호 생성)을 참조하세요.

> **Important**
>
> Secrets Manager에 Secrets Manager에 사용되는 추가 AWS 서비스에 대한 권한이 없다는 알림이 표시됩니다. 예를 들어, Amazon RDS 데이터베이스에 대한 자격 증명을 생성하려면 RDS 인스턴스, RDS 클러스터 및 Amazon Redshift 클러스터를 설명할 수 있는 권한이 있어야 합니다. 이 자습서에서 이러한 특정 AWS 서비스를 사용하지 않으므로 이러한 알림은 무시해도 좋습니다.

1. 비밀 유형 선택 섹션에서 다른 유형의 비밀을 선택합니다. 두 텍스트 상자에 test-access-key 및 test-access-secret를 입력합니다.
2. 비밀 이름 필드에 test-access-peg-eng를 입력합니다.
3. 다음 표에서 다양한 태그 조합을 추가하고 예상되는 동작을 확인합니다.
4. Store(저장)를 선택하여 비밀을 생성합니다. 스토리지에 장애가 발생하면 이전 Secrets Manager 콘솔 페이지로 돌아가서 아래 표에 있는 다음 태그 세트를 사용합니다. 마지막 태그 세트가 허용되고 비밀이 성공적으로 생성됩니다.

다음 표에는 test-access-peg-eng 역할의 ABAC 태그 조합이 나와 있습니다.

access-project 태그 값	**access-team** 태그 값	**cost-center** 태그 값	추가 태그	예상되는 동작
(none)	(none)	(none)	(none)	태그 값이 역할의 access-project 값과 일치하지 않아 거부됩니다.peg
uni	eng	987654	(none)	태그 값이 역할의 access-project 값과 일치하지 않아 거부됩니다.peg
peg	qas	987654	(none)	태그 값이 역할의 access-team 값과 일치하지 않아 거부됩니다.eng
peg	eng	123456	(none)	태그 값이 역할의 cost-center 값과 일치하지 않아 거부됩니다.987654
peg	eng	987654	Owner = Jane	세 개의 필수 태그가 모두 있고 해당 값이 역할 값과 일치하지만 추가 태그 owner가 정책에서 허용되지 않아 거부됩니다.
peg	eng	987654	Name = Jane	세 개의 필수 태그가 모두 있고 해당 값이 역할 값과 일치하기 때문에 허용됩니다. 선택적으로 Name 태그를 포함할 수도 있습니다.

5. 로그아웃하고 다음 역할 및 태그 값 각각에 대해 이 절차의 처음 세 단계를 반복합니다. 이 절차의 네 번째 단계에서는 선택한 누락된 태그, 선택적 태그, 허용되지 않는 태그 및 잘못된 태그 값 세트를 테스트합니다. 그런 다음 필수 태그를 사용하여 다음 태그와 이름으로 비밀을 생성합니다.

사용자 이름	역할 이름	비밀 이름	비밀 태그
access-Mary-peg-qas	access-peg-quality-assurance	test-access-peg-qas	access-project = peg

사용자 이름	역할 이름	비밀 이름	비밀 태그
			access-team = qas cost-center = 987654
access-Saanvi-uni-eng	access-uni-engineering	test-access-uni-eng	access-project = uni access-team = eng cost-center = 123456
access-Carlos-uni-qas	access-uni-quality-assurance	test-access-uni-qas	access-project = uni access-team = qas cost-center = 123456

5단계: 비밀 보기 테스트

각 역할에 연결된 정책을 통해 직원은 프로젝트와 관계없이 팀 이름으로 태그가 지정된 모든 비밀을 볼 수 있습니다. Secrets Manager에서 역할을 테스트하여 권한이 예상대로 작동하는지 확인합니다.

필수 태그를 사용하거나 사용하지 않고 비밀 보기를 테스트하려면

1. 다음 IAM 사용자 중 한 명으로 로그인합니다.
   - access-Arnav-peg-eng
   - access-Mary-peg-qas
   - access-Saanvi-uni-eng
   - access-Carlos-uni-qas
2. 일치하는 역할로 전환합니다.
   - access-peg-engineering
   - access-peg-quality-assurance
   - access-uni-engineering

- access-uni-quality-assurance

AWS Management Console에서의 역할 전환에 대한 자세한 내용은 역할로 전환(콘솔) 단원을 참조하십시오.

3. 왼쪽의 탐색 창에서 메뉴 아이콘을 선택하여 메뉴를 확장한 다음 비밀을 선택합니다.
4. 현재 역할과 관계없이 표에 네 가지 비밀이 모두 표시됩니다. 이는 access-same-project-team이라는 정책이 모든 리소스에 대해 secretsmanager:ListSecrets 작업을 허용하기 때문입니다.
5. 비밀 중 하나의 이름을 선택합니다.
6. 비밀에 대한 세부 정보 페이지에서 역할의 태그에 따라 페이지 콘텐츠를 볼 수 있는지 여부가 결정됩니다. 역할의 이름을 비밀의 이름과 비교합니다. 동일한 팀 이름을 공유하는 경우 access-team 태그가 일치합니다. 일치하지 않으면 액세스가 거부됩니다.

   다음 표는 각 역할의 ABAC 비밀 보기 동작을 보여줍니다.

역할 이름	비밀 이름	예상되는 동작
access-peg-engineering	test-access-peg-eng	허용됨
	test-access-peg-qas	거부됨
	test-access-uni-eng	허용됨
	test-access-uni-qas	거부됨
access-peg-quality-assurance	test-access-peg-eng	거부됨
	test-access-peg-qas	허용됨
	test-access-uni-eng	거부됨
	test-access-uni-qas	허용됨
access-uni-engineering	test-access-peg-eng	허용됨
	test-access-peg-qas	거부됨
	test-access-uni-eng	허용됨

역할 이름	비밀 이름	예상되는 동작
	test-access-uni-qas	거부됨
access-uni-quality-assurance	test-access-peg-eng	거부됨
	test-access-peg-qas	허용됨
	test-access-uni-eng	거부됨
	test-access-uni-qas	허용됨

7. 페이지 상단의 이동 경로에서 비밀을 선택하여 비밀 목록으로 돌아갑니다. 다른 역할을 사용하여 이 절차의 단계를 반복하여 각 암호를 볼 수 있는지 여부를 테스트합니다.

6단계: 테스트 확장성

RBAC(역할 기반 액세스 제어)를 통해 ABAC(속성 기반 액세스 제어)를 사용하는 중요한 이유는 확장성입니다. 회사에서 새 프로젝트, 팀 또는 인력을 AWS에 추가할 때 ABAC 기반 정책을 업데이트할 필요가 없습니다. 예를 들어, Example Company가 코드명이 Centaur인 새로운 프로젝트에 자금을 조달하고 있다고 가정합니다. Saanvi Sarkar라는 엔지니어는 Unicorn 프로젝트에서 계속 작업하면서 Centaur의 수석 엔지니어도 겸할 예정입니다. Saanvi는 Peg 프로젝트의 작업도 검토할 것입니다. 또한 Nikhil Jayashankar를 포함하여 Centaur 프로젝트에서만 작업하기 위해 몇 명의 엔지니어가 새로 고용되었습니다.

새 프로젝트를 AWS에 추가하려면

1. IAM 관리자로 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할(Roles)을 선택한 후 `access-cen-engineering`이라는 IAM 역할을 추가합니다. 역할에 **`access-same-project-team`** 권한 정책을 연결하고 다음 역할 태그 추가:
   - `access-project` = `cen`
   - `access-team` = `eng`
   - `cost-center` = `101010`
3. 왼쪽에 있는 탐색 창에서 Users(사용자)를 선택합니다.
4. `access-Nikhil-cen-eng`라는 새 사용자를 추가하고, `access-assume-role`이라는 정책을 연결하고, 다음 사용자 태그를 추가합니다.

- access-project = cen
- access-team = eng
- cost-center = 101010

5. 4단계: 비밀 생성 테스트 및 5단계: 비밀 보기 테스트의 절차를 사용합니다. 다른 브라우저 창에서 Nikhil이 Centaur 엔지니어링 비밀만 만들 수 있는지와 모든 엔지니어링 비밀을 볼 수 있는지를 테스트합니다.
6. 관리자로 로그인한 기본 브라우저 창에서 사용자 access-Saanvi-uni-eng를 선택합니다.
7. 권한 탭에서 access-assume-role 권한 정책을 제거합니다.
8. access-assume-specific-roles라는 다음 인라인 정책을 추가합니다. 인라인 정책을 사용자에게 추가하는 방법에 대한 자세한 내용은 사용자 또는 역할의 인라인 정책을 포함하려면(콘솔) 단원을 참조하십시오.

   ABAC 정책: 특정 역할만 수임

   이 정책을 통해 Saanvi는 Pegasus 또는 Centaur 프로젝트의 엔지니어링 역할을 맡을 수 있습니다. IAM에서는 다중 값 태그를 지원하지 않으므로 이 사용자 지정 정책을 생성해야 합니다. Saanvi의 사용자에게 access-project = peg 및 access-project = cen 태그를 지정할 수 없습니다. 또한 AWS 권한 부여 모델은 두 값과 모두 일치할 수 없습니다. 자세한 내용은 IAM 및 AWS STS의 태그 지정 규칙 단원을 참조하십시오. 대신 Saanvi가 맡을 수 있는 두 역할을 수동으로 지정해야 합니다.

   이 정책을 사용하려면 기울임꼴 자리 표시자 텍스트를 계정 정보로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TutorialAssumeSpecificRoles",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": [
                "arn:aws:iam::account-ID-without-hyphens:role/access-peg-
engineering",
                "arn:aws:iam::account-ID-without-hyphens:role/access-cen-
engineering"
            ]
        }
    ]
```

```
}
```

9. 4단계: 비밀 생성 테스트 및 5단계: 비밀 보기 테스트의 절차를 사용합니다. 다른 브라우저 창에서 Saanvi가 두 역할을 모두 맡을 수 있는지 확인합니다. 역할의 태그에 따라 프로젝트, 팀 및 비용 센터에 대해서만 비밀을 생성할 수 있는지 확인합니다. 또한 Saanvi가 방금 생성한 비밀을 포함하여 엔지니어링 팀이 소유한 비밀에 대한 세부 정보를 볼 수 있는지 확인합니다.

7단계: 비밀 업데이트 및 삭제 테스트

역할에 연결된 `access-same-project-team` 정책을 통해 직원은 프로젝트, 팀 및 비용 센터로 태그가 지정된 모든 비밀을 업데이트하고 삭제할 수 있습니다. Secrets Manager에서 역할을 테스트하여 권한이 예상대로 작동하는지 확인합니다.

필수 태그를 사용하거나 사용하지 않고 비밀 업데이트 및 삭제를 테스트하려면

1. 다음 IAM 사용자 중 한 명으로 로그인합니다.

   - `access-Arnav-peg-eng`
   - `access-Mary-peg-qas`
   - `access-Saanvi-uni-eng`
   - `access-Carlos-uni-qas`
   - `access-Nikhil-cen-eng`

2. 일치하는 역할로 전환합니다.

   - `access-peg-engineering`
   - `access-peg-quality-assurance`
   - `access-uni-engineering`
   - `access-peg-quality-assurance`
   - `access-cen-engineering`

   AWS Management Console에서의 역할 전환에 대한 자세한 내용은 역할로 전환(콘솔) 단원을 참조하십시오.

3. 각 역할에 대해 비밀 설명을 업데이트하고 다음 비밀을 삭제해봅니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 보안 암호 수정과 Deleting and Restoring a Secret(보안 암호 삭제 및 복원)을 참조하세요.

다음 표는 각 역할의 ABAC 비밀 업데이트 및 삭제 동작을 보여줍니다.

역할 이름	비밀 이름	예상되는 동작
access-peg-engineering	test-access-peg-eng	허용됨
	test-access-uni-eng	거부됨
	test-access-uni-qas	거부됨
access-peg-quality-assurance	test-access-peg-qas	허용됨
	test-access-uni-eng	거부됨
access-uni-engineering	test-access-uni-eng	허용됨
	test-access-uni-qas	거부됨
access-peg-quality-assurance	test-access-uni-qas	허용됨

요약

이제 속성 기반 액세스 제어(ABAC)에 태그를 사용하는 데 필요한 모든 단계를 성공적으로 완료했습니다. 태그 지정 전략을 정의하는 방법을 배웠습니다. 보안 주체와 리소스에 해당 전략을 적용했습니다. Secrets Manager에 대한 전략을 적용하는 정책을 생성하고 적용했습니다. 또한 새 프로젝트 및 팀 멤버를 추가할 때 ABAC가 쉽게 확장된다는 사실을 알게 되었습니다. 따라서 테스트 역할을 사용하여 IAM 콘솔에 로그인하고 AWS에서 ABAC에 대한 태그를 사용하는 방법을 경험할 수 있습니다.

> **Note**
>
> 특정 조건에서만 작업을 허용하는 정책을 추가했습니다. 더 광범위한 권한을 가진 사용자 또는 역할에 다른 정책을 적용하는 경우, 작업에서 태그 지정이 필요하도록 제한을 받지 않을 수 있습니다. 예를 들어 `AdministratorAccess` AWS 관리형 정책을 사용하여 사용자에게 전체 관리 권한을 부여하는 경우, 이러한 정책은 해당 액세스를 제한하지 않습니다. 여러 정책이 적용될 때 권한이 결정되는 방법에 대한 자세한 내용은 계정 내에서 요청 허용 여부 결정 단원을 참조하십시오.

관련 리소스

관련 내용은 다음 리소스를 참조하십시오.

- ABAC 권한 부여를 통한 속성 기반 권한 정의
- AWS 글로벌 조건 컨텍스트 키
- IAM 사용자 생성(콘솔)
- IAM 사용자에게 권한을 위임할 역할 생성
- AWS Identity and Access Management 리소스용 태그
- 태그를 사용한 AWS 리소스 액세스 제어
- 역할로 전환(콘솔)
- IAM 튜토리얼: ABAC에 SAML 세션 태그 사용

계정의 태그를 모니터링하는 방법을 알아보려면 서버리스 워크플로 및 Amazon CloudWatch Events를 사용하여 AWS에서 리소스 태그 변경 모니터링을 참조하세요.

IAM 튜토리얼: ABAC에 SAML 세션 태그 사용

ABAC(속성 기반 액세스 제어)는 속성에 근거하여 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 엔터티(사용자 또는 역할)와 AWS 리소스에 태그를 연결할 수 있습니다. 엔터티를 사용하여 AWS에 요청하면 해당 엔터티가 보안 주체가 되고 해당 보안 주체에 태그가 포함됩니다.

역할을 맡거나 사용자를 연동할 때 세션 태그를 전달할 수도 있습니다. 그런 다음 태그 조건 키를 사용하여 해당 태그를 기반으로 보안 주체에 권한을 부여하는 정책을 정의할 수 있습니다. 태그를 사용하여 AWS 리소스에 대한 액세스를 제어하면 AWS 정책에 대한 변경 사항이 줄어들면서 팀과 리소스가 성장할 수 있습니다. ABAC 정책은 각 개별 리소스를 나열해야 하는 기존 AWS 정책보다 유연합니다. ABAC에 대한 자세한 내용 및 기존 정책과 비교할 때의 이점은 ABAC 권한 부여를 통한 속성 기반 권한 정의 단원을 참조하십시오.

회사에서 SAML 기반 자격 증명 공급자(IdP)를 사용하여 회사 사용자 자격 증명을 관리하는 경우 AWS에서 SAML 속성을 사용하여 세밀한 액세스 제어를 수행할 수 있습니다. 속성에는 비용 센터 식별자, 사용자 이메일 주소, 부서 분류 및 프로젝트 할당이 포함될 수 있습니다. 이러한 속성을 세션 태그로 전달하면 이러한 세션 태그를 기반으로 AWS에 대한 액세스를 제어할 수 있습니다.

세션 보안 주체에 SAML 속성을 전달하여 ABAC 자습서를 완료하려면 이 주제에 포함된 변경 사항을 사용하여 IAM 튜토리얼: 태그를 기반으로 AWS 리소스에 액세스할 수 있는 권한 정의의 작업을 완료합니다.

사전 조건

ABAC에 대해 SAML 세션 태그를 사용하는 단계를 수행하려면 다음 사항이 이미 있어야 합니다.

- 특정 속성을 가진 테스트 사용자를 생성할 수 있는 SAML 기반 IdP에 대한 액세스
- 관리 권한이 있는 사용자로 로그인하는 기능.
- AWS Management Console에서 IAM 사용자, 역할 및 정책을 생성 및 편집해 본 경험. 그러나 IAM 관리 프로세스를 기억해야 하는 경우를 위해 ABAC 튜토리얼에서는 단계별 지침을 볼 수 있는 링크를 제공합니다.
- IAM에서 SAML 기반 IdP를 설정해 본 경험. 자세한 내용과 자세한 IAM 설명서에 대한 링크를 보려면 AssumeRoleWithSAML을 사용하여 세션 태그 전달 섹션을 참조하세요.

1단계: 테스트 사용자 생성

1단계: 테스트 사용자 생성의 지침을 건너뜁니다. 자격 증명은 공급자에 정의되어 있으므로 직원에 대한 IAM 사용자를 추가할 필요는 없습니다.

2단계: ABAC 정책 생성

2단계: ABAC 정책 생성의 지침에 따라 IAM에서 지정된 관리형 정책을 생성합니다.

3단계: SAML 역할 생성 및 구성

SAML용 ABAC 자습서를 사용하는 경우 역할을 생성하고, SAML IdP를 구성하고, AWS Management Console 액세스를 활성화하기 위한 추가 단계를 수행해야 합니다. 자세한 내용은 3단계: 역할 생성 단원을 참조하십시오.

3A단계: SAML 역할 생성

SAML 자격 증명 공급자 및 1단계에서 생성한 `test-session-tags` 사용자를 신뢰하는 단일 역할을 생성합니다. ABAC 자습서에서는 역할 태그가 서로 다른 별도의 역할을 사용합니다. SAML IdP에서 세션 태그를 전달하기 때문에 역할은 하나만 필요합니다. SAML 기반 역할을 생성하는 방법은 SAML 2.0 페더레이션을 위한 역할 생성(콘솔) 단원을 참조하십시오.

역할 이름을 access-session-tags로 지정합니다. 역할에 access-same-project-team 권한 정책을 연결합니다. 다음 정책을 사용하도록 역할 신뢰 정책을 편집합니다. 역할의 신뢰 관계를 편집하는 방법에 대한 자세한 지침은 역할 트러스트 정책 업데이트 단원을 참조하십시오.

다음 역할 신뢰 정책은 SAML 자격 증명 공급자 및 test-session-tags 사용자가 역할을 맡을 수 있도록 허용합니다. 역할을 맡을 때는 세 개의 지정된 세션 태그를 전달해야 합니다. 이 sts:TagSession 작업은 세션 태그 전달을 허용하는 데 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSamlIdentityAssumeRole",
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRoleWithSAML",
                "sts:TagSession"
            ],
            "Principal": {"Federated":"arn:aws:iam::123456789012:saml-
provider/ExampleCorpProvider"},
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/cost-center": "*",
                    "aws:RequestTag/access-project": "*",
                    "aws:RequestTag/access-team": [
                        "eng",
                        "qas"
                    ]
                },
                "StringEquals": {"SAML:aud": "https://signin.aws.amazon.com/saml"}
            }
        }
    ]
}
```

이 AllowSamlIdentityAssumeRole 문을 사용하면 엔지니어링 및 품질 보증 팀의 멤버가 Example Corporation IdP에서 AWS로 연동될 때 이 역할을 맡을 수 있습니다. ExampleCorpProvider SAML 공급자는 IAM에 정의되어 있습니다. 관리자가 세 개의 필수 세션 태그를 전달하도록 SAML 어설션을 이미 설정했습니다. 어설션에서 추가 태그를 전달할 수 있지만 이 세 가지 태그는 반드시 있어야 합니다. 자격 증명의 속성은 cost-center 및 access-project 태그에 대한 값을 가질 수 있습니다. 그

러나 자격 증명이 엔지니어링 또는 품질 보증 팀에 속한다는 것을 나타내려면 `access-team` 속성 값이 `eng` 또는 `qas`와 일치해야 합니다.

3B단계: SAML IdP 구성

`cost-center`, `access-project` 및 `access-team` 속성을 세션 태그로 전달하도록 SAML IdP를 구성합니다. 자세한 내용은 AssumeRoleWithSAML을 사용하여 세션 태그 전달 단원을 참조하십시오.

이러한 속성을 세션 태그로 전달하려면 SAML 어설션에 다음 요소를 포함합니다.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:cost-center">
  <AttributeValue>987654</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:access-project">
  <AttributeValue>peg</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:access-team">
  <AttributeValue>eng</AttributeValue>
</Attribute>
```

3C 단계: 콘솔 액세스 활성화

페더레이션 SAML 사용자에 대해 콘솔 액세스를 활성화합니다. 자세한 내용은 SAML 2.0 페더레이션 사용자가 AWS Management Console에 액세스할 수 있게 하기 단원을 참조하십시오.

4단계: 비밀 생성 테스트

`access-session-tags` 역할을 사용하여 AWS Management Console에 연동합니다. 자세한 내용은 SAML 2.0 페더레이션 사용자가 AWS Management Console에 액세스할 수 있게 하기 단원을 참조하십시오. 그런 다음 4단계: 비밀 생성 테스트의 지침에 따라 비밀을 생성합니다. 다양한 SAML 자격 증명을 속성과 함께 사용하여 ABAC 자습서에 표시된 태그와 일치시킵니다. 자세한 내용은 4단계: 비밀 생성 테스트 단원을 참조하십시오.

5단계: 비밀 보기 테스트

5단계: 비밀 보기 테스트 단원의 지침에 따라 이전 단계에서 생성한 비밀을 확인합니다. 다양한 SAML 자격 증명을 속성과 함께 사용하여 ABAC 자습서에 표시된 태그와 일치시킵니다.

6단계: 테스트 확장성

6단계: 테스트 확장성 단원의 지침에 따라 확장성을 테스트합니다. 다음 속성을 사용하여 SAML 기반 IdP에 새 자격 증명을 추가하면 됩니다.

- `cost-center = 101010`
- `access-project = cen`
- `access-team = eng`

7단계: 비밀 업데이트 및 삭제 테스트

7단계: 비밀 업데이트 및 삭제 테스트 단원의 지침에 따라 비밀을 업데이트 및 삭제합니다. 다양한 SAML 자격 증명을 속성과 함께 사용하여 ABAC 자습서에 표시된 태그와 일치시킵니다.

> **Important**
>
> 요금이 청구되지 않도록 생성한 모든 비밀을 삭제합니다. Secrets Manager의 요금에 대한 자세한 내용은 AWS Secrets Manager 요금을 참조하세요.

요약

이제 권한 관리를 위해 SAML 세션 태그 및 리소스 태그를 사용하는 데 필요한 모든 단계를 성공적으로 완료했습니다.

> **Note**
>
> 특정 조건에서만 작업을 허용하는 정책을 추가했습니다. 더 광범위한 권한을 가진 사용자 또는 역할에 다른 정책을 적용하는 경우, 작업에서 태그 지정이 필요하도록 제한을 받지 않을 수 있습니다. 예를 들어 `AdministratorAccess` AWS 관리형 정책을 사용하여 사용자에게 전체 관리 권한을 부여하는 경우, 이러한 정책은 해당 액세스를 제한하지 않습니다. 여러 정책이 적용될 때 권한이 결정되는 방법에 대한 자세한 내용은 계정 내에서 요청 허용 여부 결정 단원을 참조하십시오.

IAM 자습서: 사용자가 자신의 자격 증명 및 MFA 설정을 관리하도록 허용

사용자가 보안 인증 페이지에서 자신의 다중 인증(MFA) 디바이스와 보안 인증을 스스로 관리하도록 허가할 수 있습니다. AWS Management Console을 사용해 보안 인증(액세스 키, 암호, 서명 인증서 및 SSH 퍼블릭 키)을 구성하거나 필요하지 않은 보안 인증을 삭제 또는 비활성화하거나 사용자에 대해

MFA 디바이스를 활성화할 수 있습니다. 이 작업은 소수의 사용자에게는 유용하지만 사용자 수가 증가하면 곧 작업에 너무 많은 시간이 소요될 수 있습니다. 이 자습서에서는 관리자에게 부담을 주지 않으면서 이러한 모범 사례를 활성화하는 방법을 보여 줍니다.

이 자습서에서는 사용자가 MFA를 사용하여 로그인하는 경우에만 AWS 서비스에 액세스할 수 있도록 허용하는 방법을 보여 줍니다. MFA 디바이스에 로그인하지 않으면 사용자가 다른 서비스에 액세스할 수 없습니다.

이 워크플로우는 세 가지 기본 단계로 이루어집니다.

1단계: MFA 로그인을 강제할 정책 생성

소량의 IAM 작업을 제외하고 모든 작업을 금지하는 고객 관리형 정책을 생성합니다. 이러한 예외는 사용자가 보안 인증 페이지에서 자신의 보안 인증을 변경하고 MFA 디바이스를 관리할 수 있도록 허용합니다. 해당 페이지에 액세스하는 방법에 대한 자세한 내용은 IAM 사용자가 자신의 암호를 변경하는 방법(콘솔) 단원을 참조하세요.

2단계: 테스트 사용자 그룹에 정책 연결

멤버가 MFA로 로그인한 경우 해당 멤버에게 모든 Amazon EC2 작업의 전체 액세스 권한을 부여한 사용자 그룹을 생성합니다. 이러한 사용자 그룹을 생성하려면 `AmazonEC2FullAccess`라는 AWS 관리형 정책과 1단계에서 생성한 고객 관리형 정책을 모두 연결합니다.

3단계: 사용자 액세스 테스트

테스트 사용자로 로그인하여 사용자가 MFA 디바이스를 생성할 때까지 Amazon EC2에 대한 액세스가 차단되는지 확인합니다. 그런 다음 사용자는 해당 디바이스를 사용하여 로그인할 수 있습니다.

사전 조건

이 자습서의 단계를 수행하려면 다음이 준비되어 있어야 합니다.

- 관리 권한을 가진 IAM 사용자로 로그인할 수 있는 AWS 계정.
- 1단계에서 정책에 입력한 계정 ID 번호.

  계정 ID 번호를 찾으려면 페이지 상단의 탐색 표시줄에서 지원을 선택한 후 지원 센터를 선택합니다. 이 페이지의 지원 메뉴에서 계정 ID를 찾을 수 있습니다.
- 가상(소프트웨어 기반) MFA 디바이스, FIDO 보안 키 또는 하드웨어 기반 MFA 디바이스.
- 다음과 같은 사용자 그룹의 멤버인 테스트 IAM 사용자:

사용자 이름	사용자 이름 지침	사용자 그룹 이름	사용자를 멤버로 추가	사용자 그룹 지침
MFAUser	콘솔 액세스 활성화 – 선택사항 옵션만 선택하고 암호를 지정합니다.	EC2MFA	MFAUser	이 사용자 그룹에 정책을 연결하거나 권한을 부여하지 마세요.

1단계: MFA 로그인을 강제할 정책 생성

IAM 사용자가 자신의 자격 증명과 MFA 디바이스를 관리하는 데 필요한 권한을 제외한 모든 권한을 거부하는 IAM 고객 관리형 정책을 생성하는 것부터 시작합니다.

1. 관리자 자격 증명을 가진 사용자로 AWS 관리 콘솔에 로그인합니다. IAM 모범 사례를 준수하려면 AWS 계정 루트 사용자 자격 증명을 사용하여 로그인하지 마세요.

   > **Important**
   >
   > IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자를 사용하는 대신, 인간 사용자가 자격 증명 공급자와의 페더레이션을 사용하여 임시 보안 인증으로 AWS에 액세스하도록 하는 것입니다.

2. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

3. 탐색 창에서 정책을 선택한 후 정책 생성을 선택합니다.

4. JSON 탭을 선택하고 다음 JSON 정책 문서에서 텍스트를 복사합니다. AWS: MFA 인증 IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용

5. 이 텍스트를 JSON 텍스트 상자에 붙여넣습니다. 정책 검증 동안 생성된 모든 보안 경고, 오류 또는 일반 경고를 해결하고 다음을 선택합니다.

   > **Note**
   >
   > 언제든지 시각적 편집기 옵션과 JSON 옵션을 서로 전환할 수 있습니다. 그러나 위의 정책에는 시각적 편집기에서 사용할 수 없는 `NotAction` 요소가 포함되어 있습니다. 시각적 편집기 탭에 이 정책에 대한 알림 메시지가 표시됩니다. 이 정책으로 작업을 계속하려면 JSON으로 돌아가세요.

> 이 정책 예시에서는 사용자가 처음으로 AWS Management Console에 로그인하는 동안 암호 재설정을 허용하지 않습니다. 새 사용자가 로그인하고 암호를 재설정할 때까지 새 사용자에게 권한을 부여하지 않는 것이 좋습니다.

6. 검토 및 생성 페이지에서 정책 이름에 **Force_MFA**을 입력합니다. 정책 설명에 **This policy allows users to manage their own passwords and MFA devices but nothing else unless they authenticate with MFA.**를 입력합니다. 태그 영역에서 원한다면 고객 관리형 정책에 대한 태그 키-값 페어를 추가할 수 있습니다. 정책이 부여한 권한을 검토한 다음 정책 생성을 선택하여 작업을 저장합니다.

   새로운 정책이 관리형 정책 목록에 나타나며 연결 준비가 완료됩니다.

2단계: 테스트 사용자 그룹에 정책 연결

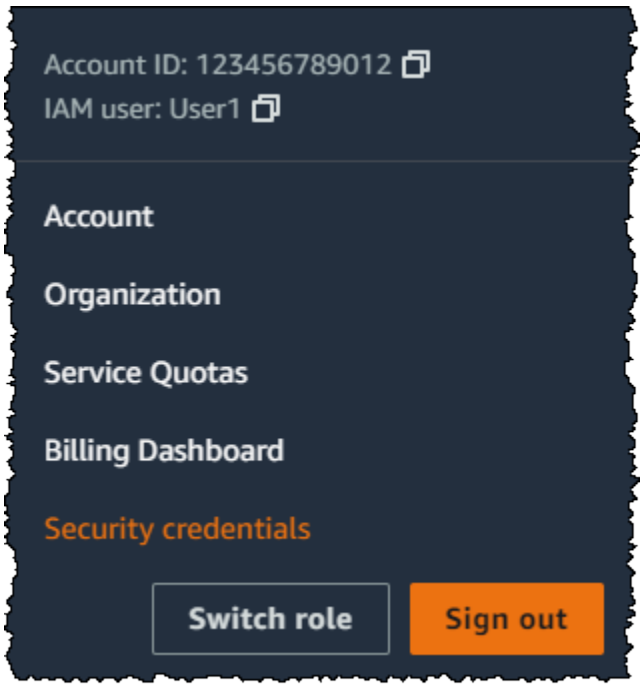
그런 다음 MFA 보호 권한을 부여하는 데 사용할 테스트 IAM 사용자 그룹에 두 개의 정책을 연결합니다.

1. 탐색 창에서 사용자 그룹(User groups)을 선택합니다.
2. 검색 상자에 **EC2MFA**를 입력한 다음 목록에서 그룹 이름(확인란 아님)을 선택합니다.
3. 권한 탭을 선택하고 권한 추가, 정책 연결을 차례로 선택합니다.
4. EC2MFA 그룹에 권한 정책 연결(Attach permission policies to EC2MFA group) 페이지의 검색 상자에 **EC2Full**을 입력합니다. 그런 다음 목록에서 AmazonEC2FullAccess 옆의 확인란을 선택합니다. 변경 내용을 저장하지 마세요.
5. 검색 상자에 **Force**를 입력한 다음, 목록에서 Force_MFA 옆에 있는 확인란을 선택합니다.
6. 정책 연결을 선택합니다.

3단계: 사용자 액세스 테스트

자습서의 이 부분에서는 테스트 사용자로 로그인하여 정책이 의도한 대로 작동하는지 검증합니다.

1. 이전 섹션에서 할당한 암호를 사용해 **MFAUser**로 AWS 계정에 로그인합니다. URL은 `https://<alias or account ID number>.signin.aws.amazon.com/console`을 사용합니다.
2. EC2를 선택해 Amazon EC2 콘솔을 열고 사용자에게 어떤 권한도 없는지 확인합니다.
3. 오른쪽 상단의 탐색 모음에서 `MFAUser` 사용자 이름을 선택한 다음 Security credentials(보안 자격 증명)를 선택합니다.



4. 이제 MFA 디바이스를 추가합니다. Multi-Factor Authentication (MFA) 섹션에서 MFA 디바이스 할당을 선택합니다.

> **Note**
>
> `iam:DeleteVirtualMFADevice` 수행 권한이 없다는 오류가 표시될 수 있습니다. 이는 이전에 다른 누군가가 가상 MFA 디바이스를 사용자에게 할당하기 시작했다가 프로세스를 취소한 경우 발생할 수 있습니다. 계속 진행하려면 사용자 또는 다른 관리자가 사용자의 기존 할당되지 않은 가상 MFA 디바이스를 삭제해야 합니다. 자세한 내용은 iam:DeleteVirtualMFADevice를 수행할 권한이 없음 단원을 참조하세요.

5. 이 자습서의 경우 휴대폰의 Google Authenticator 앱과 같은 가상(소프트웨어 기반) MFA 디바이스를 사용합니다. Authenticator app(인증 앱)을 선택하고 Next(다음)를 클릭합니다.

   IAM은 QR 코드 그래픽을 포함하여 가상 MFA 디바이스의 구성 정보를 생성 및 표시합니다. 그래픽은 QR 코드를 지원하지 않는 디바이스 상에서 수동 입력할 수 있는 보안 구성 키를 표시한 것입니다.

6. 가상 MFA 앱을 엽니다. (가상 MFA 디바이스의 호스팅에 사용되는 앱 목록은 가상 MFA 애플리케이션을 참조하십시오) 가상 MFA 앱이 다수의 계정(다수의 가상 MFA 디바이스)을 지원하는 경우 옵션을 선택하여 새로운 계정(새로운 가상 MFA 디바이스)을 생성합니다.

7. MFA 앱의 QR 코드 지원 여부를 결정한 후 다음 중 한 가지를 실행합니다.

- 마법사에서 Show QR code(QR 코드 표시)를 선택합니다. 그런 다음 앱을 사용하여 QR 코드를 스캔합니다. 예를 들어 카메라 모양의 아이콘을 선택하거나 코드 스캔(Scan code)과 비슷한 옵션을 선택한 다음, 디바이스의 카메라를 사용하여 코드를 스캔합니다.
- Set up device(디바이스 설정) 마법사에서 Show secret key(보안 키 표시)를 선택한 다음 MFA 앱에 보안 키를 입력합니다.

모든 작업을 마치면 가상 MFA 디바이스가 일회용 암호 생성을 시작합니다.

8. Set up device(디바이스 설정) 마법사의 Enter the code from your authenticator app.(Authenticator 앱에서 코드 입력) 상자에 현재 가상 MFA 디바이스에 표시된 일회용 암호를 입력합니다. Register MFA(Register MFA)를 선택합니다.

> **Important**
>
> 코드를 생성한 후 즉시 요청을 제출하세요. 코드를 생성하고 너무 오래 시간이 지난 후 요청을 제출하면 MFA 디바이스가 사용자와 연결은 되지만 MFA 디바이스가 동기화되지 않습니다. 이는 시간 기반 일회용 암호(TOTP)가 잠시 후에 만료되기 때문입니다. 이 경우, 디바이스를 재동기화할 수 있습니다.

이제 AWS에서 가상 MFA 디바이스를 사용할 준비를 마쳤습니다.

9. 콘솔에서 로그아웃한 다음, **`MFAUser`**로 다시 로그인합니다. 이번에는 AWS가 휴대전화로 받은 MFA 코드를 입력하도록 요청합니다. 코드를 받아 상자에 입력한 후 전송을 선택합니다.
10. EC2를 선택하여 Amazon EC2 콘솔을 다시 엽니다. 이번에는 모든 정보를 볼 수 있으며 원하는 작업은 모두 수행할 수 있습니다. 이 사용자로 다른 콘솔로 이동하면 액세스 거부 메시지가 표시됩니다. 그 이유는 이 튜토리얼의 정책에서 Amazon EC2에 대해서만 액세스 권한을 부여하기 때문입니다.

관련 리소스

추가 정보는 다음 주제를 참조하세요.

- IAM의 AWS 다중 인증
- MFA 지원 로그인

IAM 자격 증명(사용자, 사용자 그룹 및 역할)

> **Tip**
>
> AWS에 로그인하는 데 문제가 있나요? 올바른 로그인 페이지에 있는지 확인합니다.
>
> - AWS 계정 루트 사용자(계정 소유자)로 로그인하려면 AWS 계정을 생성할 때 설정한 보안 인증을 사용합니다.
> - IAM 사용자로 로그인하려면 계정 관리자가 AWS 로그인을 위해 제공한 보안 인증을 사용합니다.
> - IAM IDentity Center 사용자로 로그인하려면 IAM IDentity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.
>
>   IAM Identity Center 사용자로 로그인하는 데 도움이 필요한 경우 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하십시오.
>
> 로그인 자습서는 AWS 로그인 사용 설명서의 AWS 로그인 방법을 참조하세요.

> **Note**
>
> 지원을 요청해야 하는 경우 이 페이지의 Feedback(피드백) 링크를 사용하지 않도록 하세요. 입력한 피드백은 AWS Support 팀이 아닌 AWS 설명서 팀에서 받습니다. 대신 이 페이지 상단의 Contact Us(문의하기) 링크를 선택해 주세요. 여기에서 필요한 지원을 받는 데 도움이 되는 리소스 링크를 찾을 수 있습니다.

AWS 계정 루트 사용자 또는 계정의 관리 사용자가 IAM 자격 증명을 생성할 수 있습니다. IAM 자격 증명을 통해 AWS 계정에 액세스할 수 있습니다. IAM 사용자 그룹은 하나의 단위로 관리되는 IAM 사용자 집합입니다. IAM 자격 증명은 인간 사용자 또는 프로그래밍 워크로드를 대표하며, 인증된 후 AWS에서 작업을 수행할 수 있는 권한을 부여받습니다. 각 IAM 자격 증명은 하나 이상의 정책과 연결될 수 있습니다. 정책은 사용자, 역할 또는 사용자 그룹 멤버가 수행할 수 있는 작업, 작업의 대상 AWS 리소스 및 작업 수행 조건을 결정합니다.

AWS 계정 루트 사용자

AWS 계정을 생성할 때는 해당 계정의 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 단일 로그인 ID로 시작합니다. 이 ID는 AWS 계정루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다.

> **Important**
>
> 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록을 보려면 루트 사용자 보안 인증이 필요한 작업 섹션을 참조하세요.

IAM 사용자

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 AWS 계정내 ID입니다. 모범 사례로서, 가능하면 암호 및 액세스 키와 같은 장기 보안 인증 정보가 있는 IAM 사용자를 생성하는 대신 임시 보안 인증을 사용하는 것이 좋습니다. 액세스 키를 생성하기 전에 장기 액세스 키의 대안을 검토합니다. 액세스 키가 필요한 특정 사용 사례가 있는 경우 필요할 때 액세스 키를 업데이트하는 것이 좋습니다. 자세한 내용은 장기 보안 인증이 필요한 사용 사례에 필요한 경우 액세스 키 업데이트 단원을 참조하십시오. AWS 계정에 IAM 사용자를 추가하려면 AWS 계정에서 IAM 사용자 생성의 내용을 참조하세요.

> **Note**
>
> 보안 모범 사례로 IAM 사용자를 생성하는 대신 ID 페더레이션을 통해 리소스에 대한 액세스를 제공하는 것이 좋습니다. IAM 사용자가 필요한 특정 상황에 대한 자세한 내용은 IAM 사용자(역할이 아님)를 생성해야 하는 경우를 참조하세요.

IAM 사용자 그룹

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 귀하는 그룹을 사용하여 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어 IAMPublishers라는 그룹을 만들어 게시 워크로드에 일반적으로 필요한 유형의 권한을 해당 그룹에 부여합니다.

IAM 역할

IAM 역할은 특정 권한을 가지고 있는 AWS 계정 계정 내 ID입니다. 이 역할은 IAM 사용자와 유사하지만 특정 개인과 연결되지 않습니다. 역할 전환하여 AWS Management Console에서 IAM 역할을 임시로 수임할 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 역할 수임 방법 섹션을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 사용됩니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 서드 파티 자격 증명 공급자의 역할 만들기를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스를 사용하면 (역할을 프록시로 사용하는 대신) 리소스에 정책을 직접 연결할 수 있습니다. 크로스 계정 액세스를 위한 역할 및 리소스 기반 정책 간 차이점을 살펴보려면 IAM의 크로스 계정 리소스 액세스 섹션을 참조하세요.
- 교차 서비스 액세스 - 일부 AWS 서비스는 다른 AWS 서비스의 기능을 사용합니다. 예를 들어 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는직접적으로 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

  - 서비스 연결 역할 – 서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수는 없습니다.
- Amazon EC2에서 실행 중인 애플리케이션 – IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 보안 인증을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS역할을 할당하고 해당 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

IAM의 임시 자격 증명

가장 좋은 방법은 인간 사용자와 워크로드 모두에 임시 보안 인증을 사용하는 것입니다. 임시 자격 증명은 기본적으로 IAM 역할에 사용되지만 다른 용도로도 사용됩니다. 일반 IAM 사용자보다 제한된 권한을 갖는 임시 자격 증명을 요청할 수 있습니다. 이렇게 하면 제한된 보안 인증 정보로는 허용되지 않는 작업을 뜻하지 않게 수행하는 것을 방지할 수 있습니다. 임시 자격 증명의 장점은 설정한 기간이 지나면 자동으로 만료된다는 것입니다. 자격 증명의 유효 기간을 통제할 수 있습니다.

언제 IAM Identity Center 사용자를 사용하나요?

모든 인간 사용자가 IAM Identity Center를 사용하여 AWS 리소스에 액세스하는 것이 좋습니다. IAM Identity Center는 IAM 사용자로서 AWS 리소스에 액세스하는 것보다 훨씬 향상된 기능을 제공합니다. IAM Identity Center는 다음을 제공합니다.

- ID 및 할당의 중앙 세트
- 전체 AWS 조직의 계정에 대한 액세스
- 기존 ID 제공업체에 대한 연결
- 임시 보안 인증
- 멀티 팩터 인증(MFA)
- 최종 사용자를 위한 셀프 서비스 MFA 구성
- MFA 사용의 행정 집행
- 모든 AWS 계정 권한에 대한 Single Sign-On

자세한 내용은 AWS IAM Identity Center 사용 설명서의 IAM Identity Center란 무엇인가요?를 참조하세요.

IAM 사용자(역할이 아님)를 생성해야 하는 경우

페더레이션 사용자가 지원하지 않는 사용 사례에는 IAM 사용자만 사용하는 것이 좋습니다. 일부 사용 사례는 다음과 같습니다.

- IAM 역할을 사용할 수 없는 워크로드 - AWS에 액세스해야 하는 위치에서 워크로드를 실행할 수 있습니다. 일부 상황에서는 예를 들어, WordPress 플러그인에 대한 임시 보안 인증을 제공하기 위해 IAM 역할을 사용할 수 없습니다. 이러한 상황에서는 해당 워크로드에 IAM 사용자 장기 액세스 키를 사용하여 AWS에 대해 인증합니다.
- 타사 AWS 클라이언트 – IAM Identity Center를 사용한 액세스를 지원하지 않는 도구를 사용하는 경우(예: AWS에서 호스팅되지 않은 타사 AWS 클라이언트 또는 공급업체) IAM 사용자 장기 액세스 키를 사용합니다.
- AWS CodeCommit 액세스 – CodeCommit을 사용하여 코드를 저장하는 경우 CodeCommit에 대한 SSH 키 또는 서비스별 보안 인증이 있는 IAM 사용자를 사용하여 리포지토리를 인증할 수 있습니다. 일반 인증에 IAM Identity Center 사용자를 사용하는 것 외에 이렇게 하는 것이 좋습니다. IAM Identity Center 사용자는 AWS 계정 또는 클라우드 애플리케이션에 대한 액세스 권한이 필요한 인력의 사용자입니다. IAM 사용자를 구성하지 않고 CodeCommit 리포지토리에 대한 액세스 권한을 사용자에게 부여하기 위해 git-remote-codecommit 유틸리티를 구성할 수 있습니다. IAM 및 CodeCommit에 대한 자세한 내용은 CodeCommit용 IAM 자격 증명: Git 자격 증명, SSH 키 및 AWS 액세스 키 단원을 참조하세요. git-remote-codecommit 유틸리티 구성에 대한 자세한 내용은 AWS CodeCommit 사용 설명서의 보안 인증을 교체하여 AWS CodeCommit 리포지토리에 연결을 참조하세요.
- Amazon Keyspaces(Apache Cassandra용) 액세스 – IAM Identity Center 사용자를 사용할 수 없는 상황(예: Cassandra 호환성 테스트 목적)의 경우 서비스별 보안 인증이 있는 IAM 사용자를 사용하여 Amazon Keyspaces로 인증할 수 있습니다. IAM Identity Center 사용자는 AWS 계정 또는 클라우드 애플리케이션에 대한 액세스 권한이 필요한 인력의 사용자입니다. 임시 보안 인증을 사용하여 Amazon Keyspaces 연결할 수도 있습니다. 자세한 내용은 Amazon Keyspaces(Apache Cassandra용) 개발자 안내서의 임시 보안 인증과 IAM 역할 및 SigV4 플러그인을 사용하여 Amazon Keyspaces에 연결을 참조하세요.
- 긴급 액세스 – ID 제공업체에 액세스할 수 없고 AWS 계정에 조치를 취해야 하는 상황인 경우입니다. 복원 계획에 긴급 액세스 IAM 사용자 설정을 포함할 수 있습니다. 다중 인증(MFA)을 사용하여 긴급 사용자 보안 인증을 엄격하게 제어하고 보호하는 것이 좋습니다.

IAM 역할(사용자가 아님)를 만들어야 하는 경우

다음 상황에서 IAM 역할을 생성합니다.

Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에서 실행되는 애플리케이션을 생성 중이고 해당 애플리케이션이 AWS에 요청하는 경우

IAM 사용자를 만들어 해당 사용자의 자격 증명을 애플리케이션에 전달하거나 자격 증명을 애플리케이션에 포함하지 마세요. 대신 EC2 인스턴스에 연결하는 IAM 역할을 생성하여 인스턴스에서 실행되는 애플리케이션에 임시 보안 자격 증명을 부여합니다. 애플리케이션이 AWS에서 이러한 자격 증명을 사용하면 역할에 연결된 정책에서 허용하는 모든 작업을 수행할 수 있습니다. 자세한 내용은 섹션을 참조하세요IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여

휴대폰에서 실행되는 앱을 만들고 그 앱이 AWS로 요청을 보내는 경우

IAM 사용자를 만들어 앱을 통해 해당 사용자의 액세스 키를 배포하지 마세요. 대신 Login with Amazon, Amazon Cognito, Facebook 또는 Google과 같은 자격 증명 공급자를 사용하여 사용자를 인증한 다음 사용자를 IAM 역할에 매핑합니다. 앱은 역할을 사용함으로써 역할에 연결된 정책에 의해 지정된 권한을 갖는 임시 보안 자격 증명을 얻을 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- Amazon Cognito 사용 설명서
- OIDC 페더레이션

회사의 사용자가 기업 네트워크에서 인증을 받았는데 다시 로그인하지 않고도 AWS를 사용할 수 있기를 원합니다. 즉, 사용자가 AWS로 페더레이션되도록 허용하고 싶습니다.

IAM 사용자는 만들지 마세요. 엔터프라이즈 자격 증명 시스템과 AWS 사이의 연동 관계를 구성하세요. 두 가지 방법으로 수행할 수 있습니다.

- 회사의 자격 증명 시스템이 SAML 2.0과 호환된다면 회사의 자격 증명 시스템과 AWS 간에 신뢰를 구축할 수 있습니다. 자세한 내용은 SAML 2.0 연동 단원을 참조하십시오.
- 사용자의 엔터프라이즈 자격 증명을 임시 AWS 보안 자격 증명을 제공하는 IAM 역할로 변환하는 사용자 지정 프록시 서버를 만들고 사용합니다. 자세한 내용은 사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기 단원을 참조하십시오.

AWS 계정 루트 사용자 보안 인증 및 IAM 사용자 보안 인증 비교

루트 사용자는 계정 소유자이며 AWS 계정 생성 시 생성됩니다. IAM 사용자를 포함해 다른 유형의 사용자와 AWS IAM Identity Center 사용자는 루트 사용자 또는 계정 관리자가 생성합니다. 모든 AWS 사용자는 보안 인증 정보를 가지고 있습니다.

루트 사용자 보안 인증

계정 소유자의 보안 인증은 계정 내 모든 리소스에 대한 전체 액세스 권한을 허용합니다. IAM 정책을 사용하여 리소스에 대한 루트 사용자 액세스 권한을 명시적으로 거부할 수는 없습니다. 멤버 계정의 루트 사용자 권한을 제한하려면 AWS Organizations 서비스 제어 정책(SCP)을 사용해야 합니다. 따라서 IAM Identity Center에서 일상적인 AWS 작업에 사용할 관리 사용자를 생성하는 것이 좋습니다. 그런 다음 루트 사용자 보안 인증을 보호하고 이 보안 인증을 사용하여 루트 사용자로 로그인해야 하는 몇 가지 계정 및 서비스 관리 작업만 수행합니다. 이러한 작업 목록은 루트 사용자 보안 인증이 필요한 작업 섹션을 참조하세요. IAM Identity Center에서 일상적인 사용을 위해 관리자를 설정하는 방법을 알아보려면 IAM Identity Center 사용 설명서의 Getting started를 참조하세요.

IAM 보안 인증

IAM 사용자는 AWS에서 생성하는 엔터티로, AWS 리소스와 상호 작용하기 위해 IAM 사용자를 사용하는 사람 또는 서비스를 말합니다. 이러한 사용자는 특정 사용자 지정 권한을 가진 AWS 계정 내 자격 증명에 해당합니다. 예를 들어 IAM 사용자를 생성하고 IAM Identity Center에서 디렉터리를 생성할 권한을 부여할 수 있습니다. IAM 사용자는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로 또는 AWS Management Console을 사용하여 AWS에 액세스하는 데 사용할 수 있는 장기 보안 인증을 갖고 있습니다. IAM 사용자가 AWS Management Console에 로그인하는 방법에 대하 단계별 지침은 AWS 로그인 사용 설명서의 Sign in to the AWS Management Console as an IAM user를 참조하세요.

일반적으로 IAM 사용자는 사용자 이름 및 암호와 같은 장기 보안 인증을 가지고 있으므로 생성하지 않는 것이 좋습니다. 그 대신 AWS에 액세스할 때 인간 사용자에게 임시 보안 인증을 사용하도록 요구해야 합니다. 임시 보안 인증을 제공하는 IAM 역할을 수임하여 인간 사용자가 AWS 계정에 페더레이션 액세스를 제공하도록 ID 제공업체를 사용할 수 있습니다. 중앙 액세스 관리를 위해 IAM Identity Center를 사용하여 계정에 대한 액세스 권한과 해당 계정 내 권한을 관리하는 것이 좋습니다. IAM Identity Center를 사용하여 사용자 자격 증명을 관리하거나 외부 자격 증명 공급자의 IAM Identity Center에서 사용자 자격 증명에 대한 액세스 권한을 관리할 수 있습니다. 자세한 내용은 IAM Identity Center 사용 설명서의 What is IAM Identity Center를 참조하세요.

AWS 계정 루트 사용자

Amazon Web Services(AWS) 계정을 처음 생성하는 경우에는 계정의 모든 AWS 서비스 및 리소스에 대한 전체 액세스 권한을 지닌 단일 로그인 자격 증명으로 시작합니다. 이 자격 증명은 AWS 계정 루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다.

> **Important**
>
> 일상적인 작업에 루트 사용자를 사용하지 말고 AWS 계정에 대한 루트 사용자 모범 사례를 따르는 것이 좋습니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록을 보려면 루트 사용자 보안 인증이 필요한 작업 섹션을 참조하세요.

루트 사용자 문제에 대한 도움말은 루트 사용자 관련 문제 해결의 내용을 참조하세요.

다음 주제에서는 루트 사용자와 관련된 관리 작업에 대한 자세한 정보를 제공합니다.

Tasks

- AWS 계정 루트 사용자에 대한 다중 인증
- AWS 계정 루트 사용자의 암호 변경
- 잊거나 분실한 루트 사용자 암호 재설정
- 루트 사용자용 액세스 키 생성
- 루트 사용자용 액세스 키 삭제
- 루트 사용자 보안 인증이 필요한 작업
- 관련 정보

AWS 계정 루트 사용자에 대한 다중 인증

다중 인증(MFA)은 보안을 강화하는 간단하고 효과적인 메커니즘입니다. 첫 번째 요소인 암호는 기억해야 하는 비밀이며 지식 요소라고도 합니다. 다른 요소로는 보유 요소(보안 키 등 귀하가 보유하는 것) 또는 고유 요소(생체인식 스캔 등 귀하에 대한 것)가 있습니다. 보안 강화를 위해 다중 인증(MFA)을 구성하여 AWS 리소스를 보호하는 것이 좋습니다.

AWS 계정 루트 사용자와 IAM 사용자에 대해 MFA를 활성화할 수 있습니다. 루트 사용자에 대해 활성화한 MFA는 루트 사용자 자격 증명에만 영향을 줍니다. IAM 사용자에 대해 MFA를 활성화하는 방법에 대한 자세한 내용은 IAM의 AWS 다중 인증 단원을 참조하세요.

루트 사용자에 대해 MFA를 활성화하기 전에 계정 설정과 연락처 정보를 검토하고 업데이트하여 이메일 및 전화번호에 대한 액세스 권한이 있는지 확인합니다. MFA 디바이스가 분실 또는 도난되었거나 작동하지 않는 경우에도 해당 이메일과 전화번호를 사용하여 자격 증명을 확인하여 루트 사용자로 로그인할 수 있습니다. 다른 인증 요소를 사용하여 로그인하는 방법은 IAM에서 MFA로 보호되는 ID 복구 섹션을 참조하세요. 이 기능을 비활성화하려면 AWS Support에 문의하세요.

AWS는 루트 사용자를 위해 다음과 같은 MFA 유형을 지원합니다.

- 패스키 및 보안 키
- 가상 인증 애플리케이션
- 하드웨어 TOTP 토큰

패스키 및 보안 키

AWS Identity and Access Management는 MFA용 패스키 및 보안 키를 지원합니다. FIDO 표준에 기반한 패스키는 퍼블릭 키 암호화 기법을 사용하여 암호보다 안전한 강력한 피싱 방지 인증을 제공합니다. AWS는 디바이스 바운드 패스키(보안 키)와 동기화된 패스키라는 두 가지 유형의 패스키를 지원합니다.

- 보안 키: YubiKey처럼 2차 인증 요소로 사용되는 물리적 디바이스입니다. 하나의 보안 키가 여러 루트 사용자 계정과 IAM 사용자를 지원할 수 있습니다.
- 동기화된 패스키: Google, Apple, Microsoft 계정 같은 공급자와 1Password, Dashlane, Bitwarden 같은 서드 파티 서비스의 자격 증명 관리자를 2차 인증 요소로 사용합니다.

Apple MacBook의 Touch ID와 같은 내장된 생체 인식 인증자를 사용하여 자격 증명 관리자의 잠금을 해제하고 AWS에 로그인할 수 있습니다. 패스키는 지문, 얼굴 또는 디바이스 PIN을 사용하여 선택한 공급자와 함께 생성됩니다. 디바이스 간에 패스키를 동기화하여 AWS 로그인을 용이하게 하고 사용성과 복구 가능성을 높일 수 있습니다.

IAM은 Windows Hello에 대한 로컬 패스키 등록을 지원하지 않습니다. 패스키를 생성하고 사용하려면 Windows 사용자는 모바일 디바이스와 같은 한 디바이스의 패스키나 하드웨어 보안 키를 사용하여 노트북 등의 다른 디바이스에 로그인하는 크로스 디바이스 인증을 사용해야 합니다. FIDO Alliance는 FIDO 사양과 호환되는 모든 FIDO 인증 제품 목록을 유지 관리합니다. 패스키 및 보안 키 활성화에 대한 자세한 내용은 루트 사용자용 패스키 또는 보안 키 활성화(콘솔) 섹션을 참조하세요.

가상 인증 애플리케이션

가상 인증 애플리케이션은 전화 또는 기타 디바이스에서 실행되고 물리적 디바이스를 에뮬레이트합니다. 가상 인증 앱은 시간 기반 일회용 암호(TOTP) 알고리즘을 구현하고 단일 디바이스에서 여러 토큰을 지원합니다. 사용자는 로그인 중에 안내에 따라 디바이스의 유효 코드를 입력해야 합니다. 사용자에게 할당된 각 토큰은 고유해야 합니다. 사용자는 다른 사용자의 토큰의 코드를 입력하여 인증할 수 없습니다.

하드웨어 구매 승인을 기다리는 동안 또는 하드웨어 도착을 기다리는 동안 가상 MFA 디바이스를 사용하는 것이 좋습니다. 가상 MFA 디바이스로 사용할 수 있는 몇 가지 지원되는 앱의 목록은 다중 인증(MFA) 섹션을 참조하세요. AWS를 사용하여 가상 MFA 디바이스를 설정하기 위한 지침은 루트 사용자용 가상 MFA 디바이스 활성화(콘솔) 섹션을 참조하세요.

하드웨어 TOTP 토큰

하드웨어 디바이스가 시간 기반 일회용 암호(TOTP) 알고리즘에 따라 6자리 숫자 코드를 생성합니다. 사용자는 로그인할 때 두 번째 웹페이지에서 디바이스의 유효 코드를 입력해야 합니다. 사용자에게 할당된 각 MFA 디바이스는 고유해야 합니다. 사용자는 다른 사용자의 디바이스의 코드를 입력하여 인증받을 수 없습니다. 지원되는 하드웨어 MFA 디바이스에 대한 자세한 내용은 다중 인증(MFA) 섹션을 참조하세요. AWS를 사용하여 하드웨어 TOTP 토큰을 설정하는 지침은 루트 사용자에 대해 하드웨어 TOTP 토큰 활성화(콘솔) 섹션을 참조하세요.

물리적 MFA 디바이스를 사용하려는 경우 하드웨어 TOTP 디바이스 대신 FIDO 보안 키를 사용하는 것이 좋습니다. FIDO 보안 키는 배터리 요구 사항이 없고 피싱 방지가 가능하다는 이점이 있으며, 단일 디바이스에서 여러 루트 및 IAM 사용자를 지원하여 보안을 강화합니다.

주제

- 루트 사용자용 패스키 또는 보안 키 활성화(콘솔)
- 루트 사용자용 가상 MFA 디바이스 활성화(콘솔)
- 루트 사용자에 대해 하드웨어 TOTP 토큰 활성화(콘솔)

루트 사용자용 패스키 또는 보안 키 활성화(콘솔)

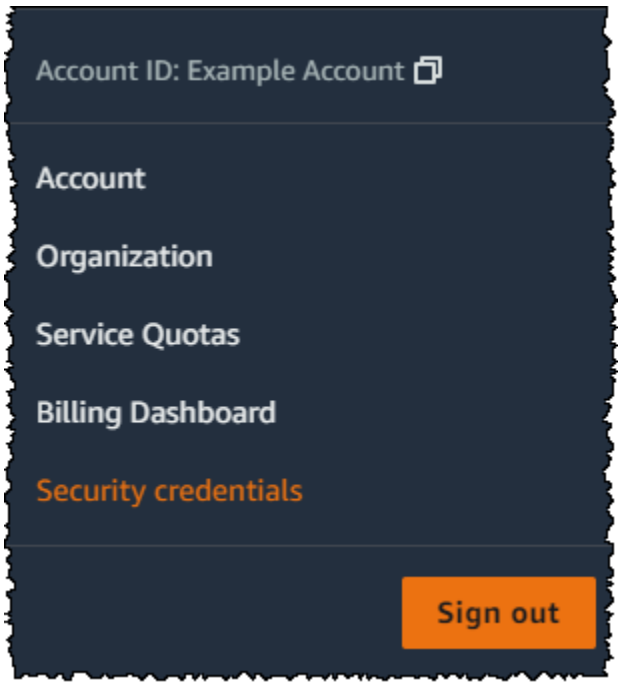
루트 사용자 패스키 구성 및 활성화는 AWS Management Console에서만 가능하고 AWS CLI 또는 AWS API에서는 불가능합니다.

루트 사용자 패스키 또는 보안 키를 활성화하려면(콘솔)

1. 루트 사용자를 선택하고 AWS 계정 계정 이메일 주소를 입력하여 IAM 콘솔에 계정 소유자로 로그인합니다. 다음 페이지에서 암호를 입력합니다.

   > **Note**
   >
   > 루트 사용자는 IAM 사용자로 로그인 페이지에 로그인할 수 없습니다. IAM 사용자로 로그인 페이지가 보이면 페이지 하단에 있는 루트 사용자 이메일을 사용하여 로그인을 선택합니다. 루트 사용자로 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 루트 사용자로 AWS Management Console 로그인을 참조하세요.

2. 탐색 표시줄 오른쪽에서 계정 이름을 선택하고 Security credentials(보안 자격 증명)를 선택합니다. 필요한 경우 Continue to Security Credentials(보안 자격 증명으로 계속)를 선택합니다.



3. 루트 사용자 내 보안 자격 증명 페이지의 다중 인증(MFA)에서 MFA 디바이스 할당을 선택합니다.
4. MFA 디바이스 이름 페이지에서 디바이스 이름을 입력하고 패스키 또는 보안 키를 선택한 후 다음을 선택합니다.
5. 디바이스 설정에서 패스키를 설정합니다. 얼굴이나 지문 같은 생체 인식 데이터 또는 디바이스 PIN을 사용하거나 컴퓨터의 USB 포트에 FIDO 보안 키를 삽입한 다음 탭하여 패스키를 생성합니다.
6. 브라우저의 지침에 따라 패스키 공급자를 선택하거나 여러 디바이스에서 사용할 패스키를 저장할 위치를 선택합니다.

7. 계속을 선택합니다.

이제 AWS에서 사용할 패스키를 등록했습니다. 다음에 루트 사용자 자격 증명을 사용하여 로그인할 때 패스키로 인증하여 로그인 절차를 완료해야 합니다.

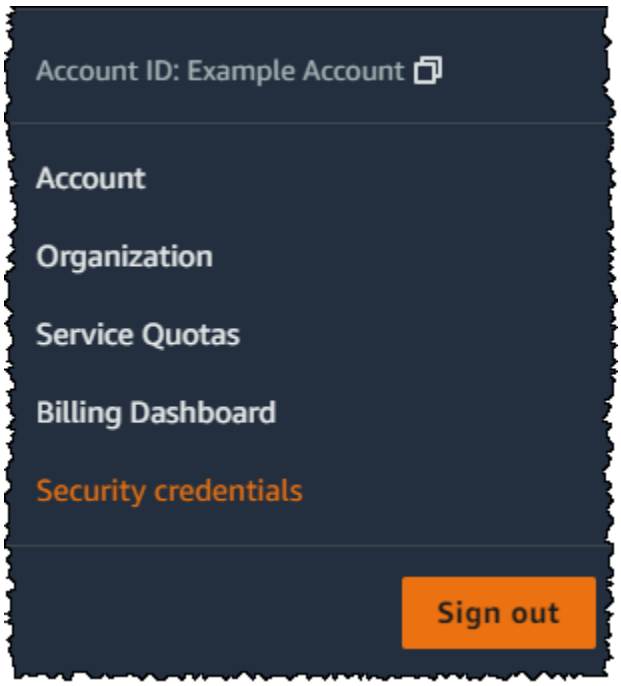
FIDO 보안 키 문제를 해결하는 데 도움이 필요한 경우 FIDO 보안 키 문제 해결 섹션을 참조하세요.

루트 사용자용 가상 MFA 디바이스 활성화(콘솔)

AWS Management Console을 사용하여 루트 사용자의 가상 MFA 디바이스를 구성 및 활성화할 수 있습니다. AWS 계정에 대해 MFA 디바이스를 활성화하려면 루트 사용자 보안 인증으로 AWS에 로그인해야 합니다.

루트 사용자에 사용하기 위해 가상 MFA 디바이스를 구성 및 활성화하려면(콘솔)

1. AWS Management Console에 로그인합니다.
2. 탐색 표시줄 오른쪽에서 계정 이름을 선택하고 Security credentials(보안 자격 증명)를 선택합니다. 필요한 경우 Continue to Security Credentials(보안 자격 증명으로 계속)를 선택합니다.



3. Multi-Factor Authentication (MFA)(다중 인증(MFA)) 섹션에서 Assign MFA device(MFA 디바이스 할당)를 선택합니다.
4. 마법사에서 디바이스 이름을 입력하고 인증 앱을 선택한 후 다음을 선택합니다.

IAM은 QR 코드 그래픽을 포함하여 가상 MFA 디바이스의 구성 정보를 생성 및 표시합니다. 그래픽은 QR 코드를 지원하지 않는 디바이스 상에서 수동 입력할 수 있는 보안 구성 키를 표시한 것입니다.

5. 디바이스에서 가상 MFA 앱을 엽니다.

   가상 MFA 앱이 다수의 가상 MFA 디바이스 또는 계정을 지원하는 경우 새로운 가상 MFA 디바이스 또는 계정을 생성하는 옵션을 선택합니다.

6. 앱을 구성하는 가장 쉬운 방법은 앱을 사용하여 QR 코드를 스캔하는 것입니다. 코드를 스캔하지 못하는 경우 구성 정보를 직접 입력할 수 있습니다. IAM에서 생성된 QR 코드와 보안 구성 키는 AWS 계정과 연동되기 때문에 다른 계정에서는 사용할 수 없습니다. 하지만 사용하던 MFA 디바이스에 대한 액세스 권한을 잃은 경우 재사용을 통해 계정에 대한 새로운 MFA 디바이스를 구성할 수 있습니다.

   - QR 코드를 사용하여 가상 MFA 디바이스를 구성하려면, 마법사에서 Show QR code(QT 코드 표시)를 선택합니다. 그리고 코드 스캔에 대한 앱 지침을 따릅니다. 예를 들어 카메라 모양의 아이콘을 선택하거나, 계정 바코드 스캔(Scan account barcode)과 같은 명령을 선택한 다음, 디바이스의 카메라를 사용하여 QR 코드를 스캔할 수 있습니다.
   - Set up device(디바이스 설정) 마법사에서 Show secret key(보안 키 표시)를 선택한 다음 MFA 앱에 보안 키를 입력합니다.

   > **Important**
   >
   > QR 코드 또는 보안 구성 키를 안전하게 백업하거나, 혹은 계정의 여러 MFA 디바이스를 활성화하세요. 현재 지원되는 MFA 유형을 조합하여 최대 8개의 MFA 디바이스를 AWS 계정 루트 사용자 및 IAM 사용자에게 등록할 수 있습니다. 예를 들어 가상 MFA 디바이스가 호스팅되어 있는 스마트폰을 분실하는 경우 가상 MFA 디바이스를 사용할 수 없습니다. 이 경우 사용자에게 연결된 추가 MFA 디바이스가 없거나 루트 사용자 MFA 디바이스 복구로도 계정에 로그인할 수 없는 경우 계정에 로그인할 수 없으며 고객 서비스에 문의하여 계정에 대한 MFA 보호를 제거해야 합니다.

   그 디바이스는 6자리 번호를 생성합니다.

7. 마법사의 MFA code 1(MFA 코드 1) 상자에 현재 가상 MFA 디바이스에 표시된 일회용 암호를 입력합니다. 디바이스가 새로운 일회용 암호를 생성할 때까지 최대 30초 기다립니다. 그런 다음 두

번째 일회용 암호를 MFA code 2(MFA 코드 2) 상자에 입력합니다. Add MFA(MFA 추가)를 선택합니다.

> **Important**
>
> 코드를 생성한 후 즉시 요청을 제출하세요. 코드를 생성한 후 너무 오래 기다렸다 요청을 제출할 경우 MFA 디바이스가 사용자와 연결은 되지만 MFA 디바이스가 동기화되지 않습니다. 이는 시간 기반 일회용 암호(TOTP)가 잠시 후에 만료되기 때문입니다. 이 경우, 디바이스를 재동기화할 수 있습니다.

이제 AWS에서 디바이스를 사용할 준비가 끝났습니다. AWS Management Console의 MFA 사용 방법에 대한 자세한 내용은 MFA 지원 로그인 섹션을 참조하세요.

루트 사용자에 대해 하드웨어 TOTP 토큰 활성화(콘솔)

AWS Management Console에서만 루트 사용자에 대한 물리적 MFA 디바이스를 구성하고 활성화할 수 있으며, AWS CLI 또는 AWS API에서는 활성화할 수 없습니다.

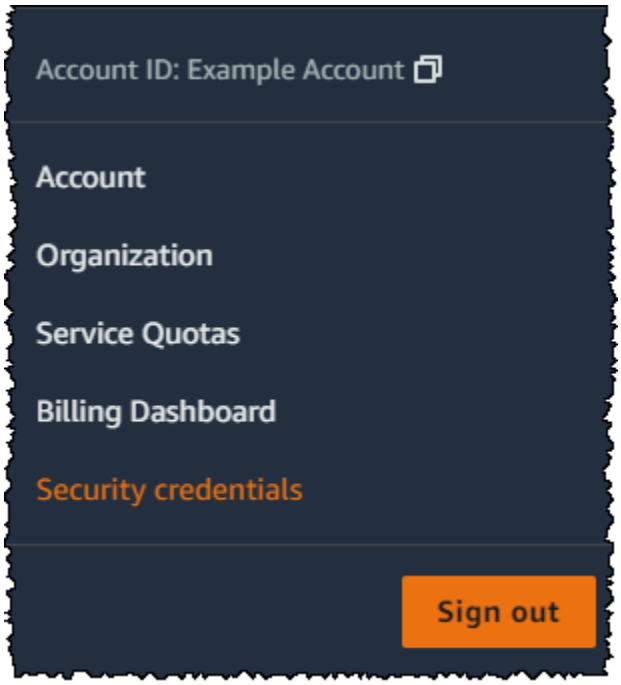
> **Note**
>
> MFA를 사용하여 로그인 및 인증 디바이스 문제 해결과 같은 다른 텍스트가 나타날 수 있습니다. 그러나 동일한 기능이 제공됩니다. 어느 경우든 대체 인증 요소를 사용하여 계정 이메일 주소 및 전화번호를 확인할 수 없는 경우 AWS Support에 문의하여 MFA 설정을 비활성화합니다.

루트 사용자에 대해 MFA 디바이스를 활성화하려면(콘솔)

1. 루트 사용자를 선택하고 AWS 계정 계정 이메일 주소를 입력하여 IAM 콘솔에 계정 소유자로 로그인합니다. 다음 페이지에서 암호를 입력합니다.

   > **Note**
   >
   > 루트 사용자는 IAM 사용자로 로그인 페이지에 로그인할 수 없습니다. IAM 사용자로 로그인 페이지가 보이면 페이지 하단에 있는 루트 사용자 이메일을 사용하여 로그인을 선택합니다. 루트 사용자로 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 루트 사용자로 AWS Management Console 로그인을 참조하세요.

2. 탐색 모음의 오른쪽에서 계정 이름을 선택한 다음 Security credentials(보안 자격 증명)를 선택합니다. 필요한 경우 Continue to Security Credentials(보안 자격 증명으로 계속)를 선택합니다.



3. Multi-factor authentication (MFA)(멀티 팩터 인증(MFA)) 섹션을 확장합니다.
4. Assign MFA device(MFA 디바이스 할당)를 선택합니다.
5. 마법사에서 Device name(디바이스 이름)을 입력하고 Hardware TOTP token(하드웨어 TOTP 토큰), Next(다음)를 차례로 선택합니다.
6. Serial number(일련 번호) 상자에 MFA 디바이스 뒷면에 있는 일련 번호를 입력합니다.
7. MFA code 1(MFA 코드 1) 상자에 MFA 디바이스에 표시된 6자리 번호를 입력합니다. 디바이스 전면의 버튼을 눌러야 번호가 표시되는 경우도 있습니다.

8. 디바이스가 코드를 새로 고칠 때까지 30초 동안 기다린 다음 MFA code 2(MFA 코드 2) 상자에 다음 6자리 번호를 입력합니다. 다시 디바이스 전면의 버튼을 눌러야 두 번째 번호가 표시되는 경우도 있습니다.
9. Add MFA(MFA 추가)를 선택합니다. 이제 MFA 디바이스가 AWS 계정과 연결되었습니다.

> **Important**
>
> 인증 코드를 생성한 후 바로 요청을 제출하세요. 코드를 생성한 후 너무 오래 기다렸다 요청을 제출할 경우 MFA 디바이스가 사용자와 연결은 되지만 MFA 디바이스가 동기화되지 않습니다. 이는 시간 기반 일회용 암호(TOTP)가 잠시 후에 만료되기 때문입니다. 이 경우, 디바이스를 재동기화할 수 있습니다.

다음에 루트 사용자 자격 증명을 사용하여 로그인할 때도 MFA 디바이스의 코드를 입력해야 합니다.

AWS 계정 루트 사용자의 암호 변경

보안 인증 정보 또는 계정 페이지에서 이메일 주소 및 암호를 변경할 수 있습니다. 또한 AWS 로그인 페이지에서 Forgot password?(암호 찾기)를 선택하여 암호를 재설정할 수 있습니다.

루트 사용자 암호를 변경하려면 IAM 사용자가 아니라 AWS 계정 루트 사용자로 로그인해야 합니다. 잊어버린 루트 사용자 암호를 재설정하는 방법에 대한 자세한 내용은 잊거나 분실한 루트 사용자 암호 재설정 섹션을 참조하세요.

암호를 보호하려면 다음과 같은 모범 사례를 활용하는 것이 중요합니다.

- 암호를 주기적으로 변경합니다.
- 암호는 비공개로 유지합니다. 암호를 아는 사람이 해당 계정에 액세스할 수도 있기 때문입니다.
- AWS의 암호를 다른 사이트에서 사용하는 것과 다르게 지정하세요.
- 짐작하기 쉬운 암호를 사용하지 마세요. 여기에는 `secret`, `password`, `amazon` 또는 `123456` 같은 암호가 포함됩니다. 또한 사전에 나오는 단어, 사용자 이름, 이메일 주소 또는 알아내기 쉬운 그 밖의 개인 정보도 피합니다.

AWS Management Console

루트 사용자의 암호를 변경하려면

> **최소 권한**
>
> 다음 단계를 수행하려면 적어도 다음과 같은 IAM 권한이 있어야 합니다.

- 추가 AWS Identity and Access Management(IAM) 권한이 필요하지 않은 AWS 계정 루트 사용자로 로그인해야 합니다. IAM 사용자 또는 역할로는 이 단계를 수행할 수 없습니다.

1. AWS 계정의 이메일 주소와 암호를 사용하여 AWS Management Console에 AWS 계정 루트 사용자로 로그인합니다.
2. 콘솔의 오른쪽 상단 모서리 부분에서 계정 이름이나 번호를 선택한 후 계정을 선택합니다.
3. 계정 페이지에서 계정 설정 옆의 편집을 선택합니다. 보안을 위해 재인증하라는 메시지가 표시됩니다.

> **Note**
>
> 편집 옵션이 나타나지 않으면 계정의 루트 사용자로 로그인하지 않은 것일 수 있습니다. IAM 사용자 또는 역할로 로그인한 상태에서는 계정 설정을 수정할 수 없습니다.

4. 계정 설정 업데이트 페이지의 암호에서 편집을 선택합니다.
5. 암호 업데이트 페이지에서 현재 암호, 새 암호, 새 암호 확인 필드를 입력합니다.

> **Important**
>
> 강력한 암호를 선택해야 합니다. IAM 사용자에 대한 계정 암호 정책을 설정할 수는 있지만 루트 사용자에게는 이 정책이 적용되지 않습니다.

AWS는 암호가 다음 조건을 충족하도록 요구합니다.

- 최소 8자, 최대 128자여야 합니다.
- 대문자, 소문자, 숫자, 기호(! @ # $ % ^ & * () <> [] {} | _+-=) 중 적어도 세 가지 문자 유형을 혼합하여 포함해야 합니다.
- AWS 계정 이름 또는 이메일 주소와 동일하지 않아야 합니다.

6. Save changes(변경 사항 저장)를 선택합니다.

AWS CLI or AWS SDK

이 작업은 AWS SDK 중 하나의 API 작업에서 또는 AWS CLI에서 지원되지 않습니다. 이 작업은 AWS Management Console을 사용해야만 수행할 수 있습니다.

잊거나 분실한 루트 사용자 암호 재설정

AWS 계정을 처음 생성할 때 이메일 주소와 암호를 입력했습니다. 그 주소와 암호가 바로 AWS 계정 루트 사용자 자격 증명입니다. 루트 사용자 암호를 잊어버린 경우, AWS Management Console에서 암호를 재설정할 수 있습니다.

> **Important**
>
> AWS에 로그인하는 데 문제가 있나요? 사용자 유형에 맞는 올바른 AWS 로그인 페이지에 있는지 확인합니다. AWS 계정 루트 사용자(계정 소유자)인 경우 AWS 계정을 생성할 때 설정한 보안 인증을 사용하여 AWS에 로그인할 수 있습니다. IAM 사용자인 경우 계정 관리자가 AWS에 로그인하는 데 사용할 수 있는 자격 증명을 제공할 수 있습니다. 지원을 요청해야 하는 경우 이 페이지의 피드백 링크를 사용하지 마세요. 이 양식은 AWS Support가 아닌 AWS 설명서 팀에서 접수합니다. 대신 Contact Us(문의처) 페이지에서 Still unable to log into your AWS account(여전히 계정에 로그인할 수 없음)을 선택한 다음 사용 가능한 지원 옵션 중 하나를 선택합니다.

루트 사용자 암호를 재설정하려면:

1. AWS 계정 이메일 주소를 사용하여 AWS Management Console에 루트 사용자로 로그인한 후 다음(Next)을 선택합니다.

   > **Note**
   >
   > AWS Management Console에 IAM 사용자 자격 증명을 사용하여 로그인한 경우 로그아웃해야 루트 사용자 암호를 재설정할 수 있습니다. 해당 계정의 IAM 사용자 로그인 페이지가 표시되면, 페이지 하단에 있는 루트 계정 자격 증명을 사용한 로그인(Sign-in using root account credentials)을 선택합니다. 필요한 경우 계정 이메일 주소를 입력하고 다음을 선택하여 Root user sign in(루트 사용자 로그인) 페이지에 액세스합니다.

2. 비밀번호가 생각나지 않는 경우를 선택합니다.

   > **Note**
   >
   > IAM 사용자인 경우 이 옵션을 사용할 수 없습니다. 비밀번호가 생각나지 않는 경우 옵션은 루트 사용자 계정에만 제공됩니다. IAM 사용자는 관리자에게 잊어버린 암호를 재설정해달라고 요청해야 합니다. 자세한 내용은 I forgot my IAM user password for my AWS

account를 참조하세요. AWS 액세스 포털을 통해 로그인하는 경우 Resetting your IAM Identity Center user password를 참조하세요.

3. 계정과 연결된 이메일 주소를 입력합니다. 그런 다음 CAPTCHA 텍스트를 입력하고 계속을 선택합니다.
4. AWS 계정과 연결된 이메일에 Amazon Web Services에서 보낸 메시지가 있는지 확인합니다. `@verify.signin.aws`로 끝나는 주소에서 보낸 이메일입니다. 이메일 지침을 따릅니다. 계정으로 이메일이 오지 않았으면 스팸 폴더를 점검합니다. 이메일에 더 이상 액세스할 수 없는 경우 AWS 로그인 사용 설명서의 AWS 계정 이메일에 액세스할 수 없음을 참조하세요.

루트 사용자용 액세스 키 생성

Warning

루트 사용자에 대한 액세스 키 페어는 생성하지 않는 것이 좋습니다. 일부 작업에만 루트 사용자가 필요하고 이러한 작업은 대개 자주 수행하지 않으므로 AWS Management Console에 로그인하여 루트 사용자 작업을 수행하는 것이 좋습니다. 액세스 키를 생성하기 전에 장기 액세스 키의 대안을 검토합니다.

권장하지는 않지만 루트 사용자에 대한 액세스 키를 생성할 수는 있습니다. AWS Command Line Interface(AWS CLI)에서 명령을 실행하거나 루트 사용자 보안 인증을 사용하여 AWS SDK 중 하나에서 API 작업을 사용하면 됩니다. 액세스 키를 만들 때 액세스 키 ID와 보안 액세스 키를 한 세트로 생성합니다. 액세스 키 생성 중에 AWS는 액세스 키의 보안 액세스 키 부분을 확인하고 다운로드할 기회를 한 번 부여합니다. 보안 액세스 키를 다운로드하지 않았거나 분실한 경우 액세스 키를 삭제한 다음 새로 생성할 수 있습니다. 콘솔, AWS CLI 또는 AWS API를 사용해 루트 사용자 액세스 키를 생성할 수 있습니다.

새로 생성한 액세스 키는 활성 상태입니다. 즉, CLI 및 API 호출에 대해 액세스 키를 사용할 수 있습니다. 루트 사용자에게 최대 두 개의 액세스 키를 할당할 수 있습니다.

사용하지 않는 액세스 키는 비활성화해야 합니다. 액세스 키가 비활성화되면 API 호출에 사용할 수 없습니다. 비활성 키는 여전히 제한에 포함됩니다. 언제든지 액세스 키를 생성하거나 삭제할 수 있습니다. 그러나 액세스 키를 삭제하면 영구 삭제되어 되돌릴 수 없습니다.

AWS Management Console

AWS 계정 루트 사용자에 대한 액세스 키를 생성하려면

> **최소 권한**
>
> 다음 단계를 수행하려면 적어도 다음과 같은 IAM 권한이 있어야 합니다.
>
> - 추가 AWS Identity and Access Management(IAM) 권한이 필요하지 않은 AWS 계정 루트 사용자로 로그인해야 합니다. IAM 사용자 또는 역할로는 이 단계를 수행할 수 없습니다.

1. AWS 계정의 이메일 주소와 암호를 사용하여 AWS Management Console 시작하기에 AWS 계정 루트 사용자로 로그인합니다.
2. 콘솔의 오른쪽 상단 모서리 부분에서 계정 이름이나 번호를 선택한 후 보안 인증을 선택합니다.
3. 액세스 키 섹션에서 액세스 키 생성을 선택합니다. 이 옵션을 사용할 수 없는 경우 이미 최대 개수의 액세스 키가 있는 것입니다. 새 키를 생성하려면 먼저 기존 액세스 키 중 하나를 삭제해야 합니다. 자세한 내용은 IAM 객체 할당량을 참조하세요.
4. 루트 사용자 액세스 키의 대안 페이지에서 보안 권장 사항을 검토합니다. 계속하려면 확인란을 선택하고 액세스 키 생성을 선택합니다.
5. 액세스 키 검색 페이지에 액세스 키 ID가 표시됩니다.
6. 비밀 액세스 키에서 표시를 선택하고 브라우저 창에서 액세스 키 ID 및 비밀 키를 복사해 다른 안전한 곳에 붙여넣습니다. 또는 .csv 파일 다운로드를 선택하면 액세스 키 ID와 비밀 키가 포함된 `rootkey.csv` 파일을 다운로드할 수 있습니다. 파일을 안전한 곳에 저장합니다.
7. 완료를 선택합니다. 액세스 키가 더 이상 필요하지 않으면 삭제하는 것이 좋습니다. 아니면 다른 사람이 악용하지 못하도록 최소한 비활성화하는 것이 좋습니다.

AWS CLI & SDKs

루트 사용자에 대한 액세스 키를 생성하려면

> **Note**
>
> 루트 사용자로 다음 명령 또는 API 작업을 실행하려면 활성 액세스 키 페어가 이미 하나 있어야 합니다. 액세스 키가 없는 경우 AWS Management Console을 사용하여 첫 번째 액세

스 키를 생성합니다. 그런 다음, AWS CLI에서 첫 번째 액세스 키의 보안 인증을 사용하여 두 번째 액세스 키를 생성하거나 액세스 키를 삭제할 수 있습니다.

- AWS CLI: aws iam create-access-key

  Example

```
$ aws iam create-access-key
{
    "AccessKey": {
        "UserName": "MyUserName",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "Status": "Active",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "CreateDate": "2021-04-08T19:30:16+00:00"
    }
}
```

- AWS API: IAM API 참조의 CreateAccessKey.

루트 사용자용 액세스 키 삭제

AWS Management Console, AWS CLI 또는 AWS API를 사용하여 루트 사용자 액세스 키를 삭제할 수 있습니다.

AWS Management Console

루트 사용자에 대한 액세스 키를 삭제하려면

> **최소 권한**
>
> 다음 단계를 수행하려면 적어도 다음과 같은 IAM 권한이 있어야 합니다.
>
> - 추가 AWS Identity and Access Management(IAM) 권한이 필요하지 않은 AWS 계정 루트 사용자로 로그인해야 합니다. IAM 사용자 또는 역할로는 이 단계를 수행할 수 없습니다.

1. AWS 계정의 이메일 주소와 암호를 사용하여 AWS Management Console 시작하기에 AWS 계정 루트 사용자로 로그인합니다.

2. 콘솔의 오른쪽 상단 모서리 부분에서 계정 이름이나 번호를 선택한 후 보안 인증을 선택합니다.
3. 액세스 키 섹션에서 삭제하려는 액세스 키를 선택하고 작업에서 삭제를 선택합니다.

> **Note**
>
> 또는 액세스 키를 영구적으로 삭제하는 대신 비활성화할 수도 있습니다. 이렇게 하면 키 ID나 비밀 키를 변경하지 않고도 나중에 액세스 키를 다시 사용할 수 있습니다. 키가 비활성 상태인 동안 AWS API에 대한 요청에 이 키를 사용하려고 하면 액세스 거부됨 오류로 인해 실패합니다.

4. <access key ID> 삭제 대화 상자에서 비활성화를 선택하고 삭제할 액세스 키 ID를 입력한 다음, 삭제를 선택합니다.

AWS CLI & SDKs

루트 사용자에 대한 액세스 키를 삭제하려면

> **최소 권한**
>
> 다음 단계를 수행하려면 적어도 다음과 같은 IAM 권한이 있어야 합니다.
>
> - 추가 AWS Identity and Access Management(IAM) 권한이 필요하지 않은 AWS 계정 루트 사용자로 로그인해야 합니다. IAM 사용자 또는 역할로는 이 단계를 수행할 수 없습니다.

- AWS CLI: aws iam delete-access-key

  Example

  ```
  $ aws iam delete-access-key \
      --access-key-id AKIAIOSFODNN7EXAMPLE
  ```

  이 명령은 성공 시 출력을 생성하지 않습니다.

- AWS API: DeleteAccessKey

루트 사용자 보안 인증이 필요한 작업

> **Important**
>
> AWS에 로그인하는 데 문제가 있나요? 사용자 유형에 맞는 올바른 AWS 로그인 페이지에 있는지 확인합니다. AWS 계정 루트 사용자(계정 소유자)인 경우 AWS 계정을 생성할 때 설정한 보안 인증을 사용하여 AWS에 로그인할 수 있습니다. IAM 사용자인 경우 계정 관리자가 AWS에 로그인하는 데 사용할 수 있는 자격 증명을 제공할 수 있습니다. 지원을 요청해야 하는 경우 이 페이지의 피드백 링크를 사용하지 마세요. 이 양식은 AWS Support가 아닌 AWS 설명서 팀에서 접수합니다. 대신 Contact Us(문의처) 페이지에서 Still unable to log into your AWS account(여전히 계정에 로그인할 수 없음)을 선택한 다음 사용 가능한 지원 옵션 중 하나를 선택합니다.

일상 작업을 수행하고 AWS 리소스에 액세스하도록 AWS IAM Identity Center에서 관리 사용자를 구성하는 것이 좋습니다. 하지만 아래 나열된 작업은 계정의 루트 사용자로 로그인한 경우에만 수행할 수 있습니다.

루트 사용자 문제에 대한 도움말은 루트 사용자 관련 문제 해결의 내용을 참조하세요.

계정 관리 작업

- 계정 설정을 변경합니다. 여기에는 계정 이름, 이메일 주소, 루트 사용자 암호 및 루트 사용자 액세스 키가 포함됩니다. 연락처 정보, 결제 통화 기본 설정 및 AWS 리전 등의 기타 계정 설정에는 루트 사용자 보안 인증이 필요하지 않습니다.
- IAM 사용자 권한을 복원합니다. IAM 관리자가 실수로 권한을 취소하면 루트 사용자로 로그인하여 정책을 편집하고 해당 권한을 복원할 수 있습니다.
- AWS 계정을 닫습니다.

  자세한 정보는 다음 주제를 참조하세요.

  - AWS 계정에 대한 내 소유권을 다른 법인에 할당하려면 어떻게 해야 하나요? .
  - 내 AWS 계정을 어떻게 해지하나요?.
  - 독립 실행형 AWS 계정을 닫습니다.

결제 작업

- 과금 정보 및 비용 관리 콘솔에 대한 IAM 액세스를 활성화합니다.

- 일부 결제 작업은 루트 사용자만 수행할 수 있습니다. 자세한 내용은 AWS Billing 사용 설명서에서 AWS 계정 관리를 참조하세요.
- 특정 세금 계산서를 조회합니다. aws-portal:ViewBilling 권한이 있는 IAM 사용자는 AWS 유럽(AWS Inc. 또는 Amazon Internet Services Private Limited(AISPL) 제외)에서 VAT 인보이스를 보고 다운로드할 수 있습니다.

AWS GovCloud (US) 작업

- AWS GovCloud (US)에 대한 가입.
- AWS Support에서 AWS GovCloud (US) 계정 루트 사용자 액세스 키를 요청합니다.

Amazon EC2 작업

- 예약 인스턴스 마켓플레이스에 판매자로 등록합니다.

AWS KMS 작업

- AWS Key Management Service 키를 관리할 수 없게 된 경우 관리자는 AWS Support에 연락하여 키를 복구할 수 있습니다. 그러나 AWS Support는 티켓 OTP 확인을 통한 권한 부여를 위해 루트 사용자의 기본 전화번호로 응답합니다.

Amazon Mechanical Turk Task

- 귀하의 AWS 계정을 MTurk 요청자 계정에 연결합니다.

Amazon Simple Storage Service Tasks

- 멀티 팩터 인증(MFA)을 활성화하도록 Amazon S3 버킷을 구성합니다.
- 모든 보안 주체를 거부하는 Amazon S3 버킷 정책을 편집하거나 삭제합니다.

Amazon Simple Queue Service Task

- 모든 보안 주체를 거부하는 Amazon SQS 리소스 정책을 편집하거나 삭제합니다.

관련 정보

다음 문서에서는 루트 사용자 작업에 대한 추가 정보를 제공합니다.

- 내 AWS 계정 및 해당 리소스를 보호하기 위한 모범 사례로 무엇이 있나요?
- 내 루트 사용자가 사용되었음을 알리는 EventBridge 이벤트 규칙을 생성하려면 어떻게 해야 하나요?
- AWS 계정 루트 사용자 활동 모니터링 및 알림
- IAM 루트 사용자 활동 모니터링

IAM 사용자

> **Important**
>
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자를 사용하는 대신, 인간 사용자가 자격 증명 공급자와의 페더레이션을 사용하여 임시 보안 인증으로 AWS에 액세스하도록 하는 것입니다.

AWS Identity and Access Management(IAM) 사용자는 AWS에서 생성하는 엔터티입니다. IAM 사용자는 AWS와의 상호 작용에 IAM 사용자를 사용하는 인간 사용자 또는 서비스를 나타냅니다. AWS에서 사용자는 이름과 보안 인증으로 구성됩니다.

관리자 권한이 있는 IAM 사용자는 AWS 계정 루트 사용자과 같지 않습니다. 루트 사용자에 대한 자세한 내용은 AWS 계정 루트 사용자 섹션을 참조하세요.

AWS가 IAM 사용자를 식별하는 방법

사용자를 생성하면 IAM은 해당 사용자를 식별하는 다음과 같은 방법을 생성합니다.

- IAM 사용자 생성 시 지정한 이름으로서 `Richard` 또는 `Anaya`와 같은 IAM 사용자가 "쉽게 알 수 있는 이름"입니다. 이 이름들은 AWS Management Console에서 볼 수 있습니다.
- IAM 사용자의 Amazon 리소스 이름(ARN)입니다. 모든 AWS 전반에 IAM 사용자를 특별하게 식별할 필요가 있는 경우 ARN을 사용합니다. 예를 들어, ARN을 사용하여 IAM 사용자를 Amazon S3 버킷의 IAM 정책에서 `Principal`로 지정할 수 있습니다. IAM 사용자의 ARN은 다음과 같을 수 있습니다.

  ```
  arn:aws:iam::account-ID-without-hyphens:user/Richard
  ```

- IAM 사용자의 고유 식별자입니다. 이 ID는 IAM 사용자를 생성하기 위해 API, Tools for Windows PowerShell 또는 AWS CLI를 사용할 때만 반환됩니다. 콘솔에서는 이 ID를 볼 수 없습니다.

이 식별자에 대한 자세한 정보는 IAM 식별자 섹션을 참조하세요.

IAM 사용자 및 보안 인증

AWS는 IAM 사용자 보안 인증에 따라 다양한 방법으로 액세스할 수 있습니다.

- 콘솔 암호: IAM 사용자가 입력해 AWS Management Console과 같은 상호 작용 세션으로 로그인할 수 있는 암호. IAM 사용자의 암호(콘솔 액세스)를 비활성화하면 사용자가 해당 로그인 보안 인증을 사용하여 AWS Management Console에 로그인하지 못합니다. 그렇더라도 사용자의 권한이 변경되거나 위임된 역할을 사용하여 콘솔에 액세스하지 못하게 되지는 않습니다.
- 액세스 키: 프로그래밍 방식으로 AWS를 호출하는 데 사용됩니다. 그러나 IAM 사용자를 위한 액세스 키를 생성하기 전에 고려해야 할 더 안전한 대안이 있습니다. 자세한 내용은 AWS 일반 참조의 장기 액세스 키에 대한 고려 사항 및 대안을 참조하세요. IAM 사용자에게 활성 액세스 키가 있다면 해당 키는 계속 작동하고 AWS CLI, Tools for Windows PowerShell 또는 AWS API 또는 AWS Console Mobile Application을 통해 액세스를 허용합니다.
- CodeCommit에 사용할 SSH 키: CodeCommit를 사용한 인증에 사용할 수 있는 OpenSSH 형식의 SSH 퍼블릭 키
- 서버 인증서: 일부 AWS 서비스를 사용한 인증에 사용할 수 있는 SSL/TLS 인증서. 서버 인증서를 프로비저닝, 관리 및 배포할 때 AWS Certificate Manager(ACM)를 사용하는 것이 좋습니다. ACM에서 지원하지 않는 리전에서 HTTPS 연결을 지원해야 하는 경우에만 IAM을 사용합니다. ACM을 지원하는 리전을 알아보려면 AWS 일반 참조의 AWS Certificate Manager 엔드포인트 및 할당량을 참조하세요.

IAM 사용자에게 적절한 자격 증명을 선택할 수 있습니다. AWS Management Console을 사용하여 IAM 사용자를 생성할 때 최소한 콘솔 암호 또는 액세스 키를 포함하도록 선택해야 합니다. 기본적으로 AWS CLI 또는 AWS API를 사용하여 새로 생성된 IAM 사용자는 어떤 종류의 자격 증명도 보유하지 않습니다. 사용자의 사용 사례를 기반으로 IAM 사용자의 보안 인증 유형을 생성해야 합니다.

다음 옵션을 이용해 암호, 액세스 키 및 다중 인증(MFA) 디바이스를 관리합니다.

- IAM 사용자 암호 관리. AWS Management Console에 대한 액세스를 허용하는 암호를 생성 및 변경합니다. 암호 정책을 최소 암호 복잡성을 적용하도록 설정 사용자에게 자신의 암호를 변경할 수 있도록 허용

- IAM 사용자의 액세스 키 관리. 계정의 리소스에 대한 프로그래밍 방식의 액세스를 위해 액세스 키를 생성하고 업데이트합니다.
- IAM 사용자에 대해 다중 인증(MFA)을 활성화합니다. 가장 좋은 방법은 계정에 속한 모든 IAM 사용자에게 다중 인증(MFA)을 요구하는 것입니다. MFA를 사용할 경우 사용자는 두 가지 형식의 식별이 가능합니다. 먼저, 사용자는 자격 증명(암호 또는 액세스 키)의 일부분인 보안 인증을 제공합니다. 또한 하드웨어 디바이스 또는 스마트폰이나 태블릿의 애플리케이션에서 생성된 임시 숫자 코드를 제공합니다.
- 미사용 암호 및 액세스 키 찾기. 계정 또는 계정 내 IAM 사용자의 암호 또는 액세스 키를 보유하는 사람은 누구든지 AWS 리소스에 액세스할 수 있습니다. 보안 모범 사례는 사용자에게 암호와 액세스 키가 필요하지 않을 때 그것들을 제거하는 것입니다.
- 계정의 자격 증명 보고서 다운로드. 계정의 모든 IAM 사용자와 해당 사용자의 암호, 액세스 키, MFA 디바이스 등 다양한 자격 증명의 상태를 나열하는 자격 증명 보고서를 생성하고 다운로드할 수 있습니다. 암호와 액세스 키의 경우 자격 증명 보고서를 통해 암호 또는 액세스 키가 언제 마지막으로 사용되었는지 알 수 있습니다.

IAM 사용자 및 권한

기본적으로 새로운 IAM 사용자는 작업을 수행할 어떠한 권한도 없습니다. 사용자는 AWS 작업을 수행하거나 AWS 리소스에 액세스할 수 있는 권한이 없습니다. 개별 IAM 사용자를 두면 각 사용자에게 개별적으로 권한을 할당할 수 있다는 장점이 있습니다. 사용자 몇 명에게 관리 권한을 할당하면 이들이 AWS 리소스를 관리하고 다른 IAM 사용자까지 생성하고 관리할 수 있습니다. 그러나 대부분의 경우 사용자의 업무에 필요한 작업(AWS 작업) 및 리소스로만 사용자의 권한을 제한합니다.

Diego라는 사용자가 있다고 가정해 보겠습니다. IAM 사용자 `Diego`를 생성할 때 해당 사용자에 대한 암호를 생성하고 권한을 연결하여 특정 Amazon EC2 인스턴스를 시작하고 Amazon RDS 데이터베이스의 테이블에서 (GET) 정보를 읽을 수 있도록 할 수 있습니다. 사용자를 생성하여 초기 자격 증명과 권한을 부여하는 절차는 AWS 계정에서 IAM 사용자 생성 섹션을 참조하세요. 기존 사용자에 대한 권한을 변경하는 절차는 IAM 사용자의 권한 변경 섹션을 참조하세요. 사용자의 암호나 액세스 키를 변경하는 절차는 AWS에서 사용자 암호 관리 및 IAM 사용자의 액세스 키 관리 섹션을 참조하세요.

IAM 사용자에게 권한 경계를 추가할 수 있습니다. 권한 경계는 AWS 관리형 정책을 사용하여 자격 증명 기반 정책이 IAM 사용자 또는 역할에 부여할 수 있는 최대 권한을 제한할 수 있는 고급 기능입니다. 정책 유형 및 활용에 대한 자세한 정보는 IAM의 정책 및 권한 섹션을 참조하세요.

IAM 사용자 및 계정

각 IAM 사용자는 오직 한 개의 AWS 계정과만 연결됩니다. IAM 사용자는 AWS 계정 내에서 정의되기 때문에 AWS에서 파일에 결제 방법을 저장해 두지 않아도 됩니다. 계정에 속한 IAM 사용자가 수행하는 모든 AWS 활동은 해당 계정으로 청구됩니다.

AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 단원을 참조하십시오.

서비스 계정인 IAM 사용자

IAM 사용자는 연결된 자격 증명 및 권한을 지닌 IAM의 리소스입니다. IAM 사용자는 자격 증명을 사용하여 AWS에 요청하는 사용자 또는 애플리케이션을 나타낼 수 있습니다. 이를 일반적으로 서비스 계정이라 합니다. 애플리케이션에서 IAM 사용자의 장기 자격 증명을 사용하기로 선택한 경우 액세스 키를 애플리케이션 코드에 직접 포함하지 마세요. AWS SDK 및 AWS Command Line Interface를 사용하면 코드에서 유지할 필요가 없도록 알려진 위치에 액세스 키를 추가할 수 있습니다. 자세한 내용은 AWS 일반 참조의 적절하게 IAM 사용자 액세스 키 관리를 참조하세요. 또는 모범 사례로서 장기 액세스 키 대신 임시 보안 자격 증명(IAM 역할)을 사용할 수 있습니다.

IAM 사용자가 AWS에 로그인하는 방법

IAM 사용자로 AWS Management Console에 로그인하려면 사용자 이름과 암호 이외에 계정 ID 또는 계정 별칭을 제공해야 합니다. 관리자가 콘솔에서 IAM 사용자를 생성한 경우, 계정 ID 또는 계정 별칭이 포함된 계정 로그인 페이지의 URL, 사용자 이름 등을 비롯한 로그인 자격 증명을 전송했을 것입니다.

```
https://My_AWS_Account_ID.signin.aws.amazon.com/console/
```

> **도움말**
>
> 웹 브라우저에서 계정 로그인 페이지를 위한 북마크를 만들려면 북마크 입력란에 계정의 로그인 URL을 직접 입력해야 합니다. 리디렉션은 로그인 URL을 가릴 수 있으므로 웹 브라우저 북마크 기능을 사용하지 마세요.

다음의 일반 로그인 엔드포인트에서 로그인하고 계정 ID 또는 계정 별칭을 직접 입력할 수도 있습니다.

```
https://console.aws.amazon.com/
```

사용자 편의를 위해 AWS 로그인 페이지는 브라우저 쿠키를 사용하여 IAM 사용자 이름 및 계정 정보를 기억합니다. 다음에 사용자가 AWS Management Console의 아무 페이지로든 이동하면 콘솔이 쿠키를 사용하여 사용자를 사용자 로그인 페이지로 리디렉션합니다.

IAM 사용자 자격 증명에 연결된 정책에서 관리자가 지정하는 AWS 리소스에만 액세스할 수 있습니다. 콘솔에서 작업하려면 AWS 리소스 나열 및 생성 등 콘솔이 수행하는 작업을 수행할 권한이 있어야 합니다. 자세한 내용은 AWS리소스에 대한 액세스 관리 및 IAM 자격 증명 기반 정책의 예 섹션을 참조하세요.

> **Note**
>
> 조직에 기존의 자격 증명 시스템이 있는 경우, Single Sign-On(SSO) 옵션을 만드는 것이 좋습니다. SSO는 IAM 사용자 자격 증명이 없어도 계정의 AWS Management Console에 액세스할 수 있는 권한을 사용자에게 제공합니다. 또한 SSO를 사용하면 사용자가 조직의 사이트와 AWS에 따로 로그인하지 않아도 됩니다. 자세한 내용은 사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기 섹션을 참조하세요.

CloudTrail에 로그인 세부 정보 로깅

CloudTrail에서 로그인 이벤트를 로그에 기록하도록 설정한 경우 CloudTrail에서 이벤트를 기록할 위치를 선택하는 방법을 알고 있어야 합니다.

- 사용자가 콘솔에 직접 로그인할 경우 선택한 서비스 콘솔이 리전을 지원하는지 여부를 기준으로 글로벌 또는 리전 로그인 종단점으로 리디렉션됩니다. 예를 들어 메인 콘솔 홈 페이지는 리전을 지원합니다. 따라서 다음 URL에 로그인할 경우

  ```
  https://alias.signin.aws.amazon.com/console
  ```

  `https://us-east-2.signin.aws.amazon.com`과 같은 리전 로그인 엔드포인트로 리디렉션되어 사용자의 리전 로그에 리전 CloudTrail 로그 항목이 기록됩니다.

  반면 Amazon S3 콘솔은 리전을 지원하지 않습니다. 따라서 다음 URL에 로그인할 경우

  ```
  https://alias.signin.aws.amazon.com/console/s3
  ```

  AWS가 `https://signin.aws.amazon.com`의 글로벌 로그인 엔드포인트로 사용자를 리디렉션하여 글로벌 CloudTrail 로그 항목이 기록됩니다.

- 다음과 같은 URL 구문을 사용하여 리전이 활성화된 메인 콘솔 홈 페이지에 로그인하면 특정 리전 로그인 종단점을 수동으로 요청할 수 있습니다.

```
https://alias.signin.aws.amazon.com/console?region=ap-southeast-1
```

  AWS가 사용자를 `ap-southeast-1` 리전 로그인 엔드포인트로 리디렉션하고 리전 CloudTrail 로그 이벤트가 발생합니다.

CloudTrail 및 IAM에 대한 자세한 내용은 CloudTrail로 IAM 이벤트 로깅을 참조하세요.

계정을 통해 사용자가 작업하기 위해 프로그래밍 방식의 액세스가 필요한 경우 각 사용자의 액세스 키 페어(액세스 키 ID와 보안 액세스 키)를 생성할 수 있습니다. 그러나 사용자를 위한 액세스 키를 생성하기 전에 고려해야 할 더 안전한 대안이 있습니다. 자세한 내용은 AWS 일반 참조의 장기 액세스 키에 대한 고려 사항 및 대안을 참조하세요.

MFA 지원 로그인

멀티 팩터 인증(MFA) 디바이스를 사용하여 구성된 사용자는 MFA 디바이스를 사용하여 AWS Management Console에 로그인해야 합니다. 사용자가 자신의 로그인 보안 인증 정보를 입력하면 AWS는 해당 사용자의 계정에서 해당 사용자에게 MFA가 필요한지를 확인합니다. 다음 주제에서는 MFA가 필요할 때 사용자가 로그인을 완료하는 방법에 대한 정보를 제공합니다.

주제

- 다중 MFA 디바이스 활성화
- FIDO 보안 키
- 가상 MFA 디바이스
- 하드웨어 TOTP 토큰

다중 MFA 디바이스 활성화

사용자가 AWS 계정 루트 사용자 또는 해당 계정에 대해 활성화된 여러 MFA 디바이스가 있는 IAM 사용자로 AWS Management Console에 로그인하는 경우 하나의 MFA 디바이스만 사용하여 로그인하면 됩니다. 사용자가 사용자 암호로 인증한 후 인증을 완료하는 데 사용할 MFA 디바이스 유형을 선택합니다. 그런 다음 사용자에게 선택한 디바이스 유형으로 인증하라는 메시지가 표시됩니다.

FIDO 보안 키

MFA가 필요한 사용자에게는 두 번째 로그인 페이지가 나타납니다. 사용자가 FIDO 보안 키를 터치해야 합니다.

> **Note**
>
> Google Chrome 사용자는 amazon.com으로 아이덴티티 확인(Verify your identity with amazon.com)을 요청하는 팝업에서 사용 가능한 옵션을 선택해서는 안 됩니다. 보안 키를 탭하기만 하면 됩니다.

다른 MFA 디바이스와 달리 FIDO 보안 키는 항상 동기화되어 있습니다. FIDO 보안 키를 분실했거나 도난당한 경우 관리자가 비활성화할 수 있습니다. 자세한 내용은 MFA 디바이스 비활성화(콘솔) 단원을 참조하십시오.

WebAuthn을 지원하는 브라우저와 AWS에서 지원하는 FIDO 준수 디바이스에 대한 자세한 내용을 알아보려면 패스키 및 보안 키 사용이 지원되는 구성 섹션을 참조하세요.

가상 MFA 디바이스

MFA가 필요한 사용자에게는 두 번째 로그인 페이지가 나타납니다. MFA code(MFA 코드) 상자에 MFA 애플리케이션에서 제공한 숫자 코드를 입력해야 합니다.

MFA 코드가 올바르면 사용자는 AWS Management Console에 액세스할 수 있습니다. 코드가 올바르지 않으면 다른 코드로 다시 시도할 수 있습니다.

가상 MFA 디바이스는 동기화되지 않을 수 있습니다. 여러 번 시도한 후에도 사용자가 AWS Management Console에 로그인할 수 없으면 가상 MFA 디바이스를 동기화하라는 메시지가 표시됩니다. 사용자는 화면에 표시되는 메시지에 따라 가상 MFA 디바이스를 동기화할 수 있습니다. AWS 계정에 속한 사용자 대신 디바이스를 동기화할 수 있는 방법에 대한 자세한 내용은 가상 및 하드웨어 MFA 디바이스 재동기화 섹션을 참조하세요.

하드웨어 TOTP 토큰

MFA가 필요한 사용자에게는 두 번째 로그인 페이지가 나타납니다. MFA code(MFA 코드) 상자에 하드웨어 TOTP 토큰에서 제공한 숫자 코드를 입력해야 합니다.

MFA 코드가 올바르면 사용자는 AWS Management Console에 액세스할 수 있습니다. 코드가 올바르지 않으면 다른 코드로 다시 시도할 수 있습니다.

하드웨어 TOTP 토큰이 동기화되지 않을 수 있습니다. 여러 번 시도하여 실패한 후에도 사용자가 AWS Management Console에 로그인할 수 없으면 MFA 토큰 디바이스를 동기화하라는 메시지가 표시됩니다. 사용자는 화면에 표시되는 메시지에 따라 MFA 토큰 디바이스를 동기화할 수 있습니다. AWS 계정에 속한 사용자 대신 디바이스를 동기화할 수 있는 방법에 대한 자세한 내용은 가상 및 하드웨어 MFA 디바이스 재동기화 섹션을 참조하세요.

AWS 계정에서 IAM 사용자 생성

> **Important**
>
> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자를 사용하는 대신, 인간 사용자가 자격 증명 공급자와의 페더레이션을 사용하여 임시 보안 인증으로 AWS에 액세스하도록 하는 것입니다.

> **Note**
>
> 웹 사이트에서 Amazon 제품을 판매하고자 Product Advertising API에 대한 정보를 찾고 있기 때문에 이 페이지를 발견한 경우 Product Advertising API 5.0 설명서를 참조하세요.
> IAM 콘솔에서 이 페이지로 이동했을 경우 로그인을 했더라도 계정에 IAM 사용자가 포함되지 않을 수 있습니다. 역할을 사용하거나, 임시 자격 증명으로 로그인하여 AWS 계정 루트 사용자로 로그인할 수 있습니다. 이러한 IAM 자격 증명에 대한 자세한 내용은 IAM 자격 증명(사용자, 사용자 그룹 및 역할) 섹션을 참조하세요.

다음 단계에 따라 사용자를 생성하고 사용자가 작업을 수행할 수 있습니다.

1. AWS Management Console, AWS CLI 또는 Tools for Windows PowerShell에서나 AWS API 작업을 사용하여 사용자를 생성합니다. AWS Management Console에서 사용자를 생성하면 선택한 항목에 따라 1~4단계는 자동으로 처리됩니다. 프로그래밍 방식으로 사용자를 생성하는 경우, 각 단계를 개별적으로 수행해야 합니다.
2. 사용자에게 필요한 액세스 유형에 따라 사용자의 자격 증명을 생성합니다.
   - 콘솔 액세스 활성화 – 선택 사항: 사용자가 AWS Management Console에 액세스해야 할 경우, 해당 사용자의 암호를 생성합니다. 사용자의 콘솔 액세스를 비활성화하면 사용자가 사용자 이름과 암호를 사용하여 AWS Management Console에 로그인하지 못합니다. 그렇더라도 사용자의 권한이 변경되거나 위임된 역할을 사용하여 콘솔에 액세스하지 못하게 되지는 않습니다.

> **Tip**
>
> 사용자에게 필요한 보안 인증만 생성하세요. 예를 들어, AWS Management Console을 통해서만 액세스해야 하는 사용자에게는 액세스 키를 생성해서는 안 됩니다.

3. 해당 사용자를 하나 이상의 그룹에 추가하여 필요한 작업을 수행할 수 있는 권한을 부여합니다. 권한 정책을 사용자에게 직접 연결하여 권한을 부여할 수도 있습니다. 하지만, 사용자를 그룹에 추가한 후 그 그룹에 연결된 정책을 통해 정책과 권한을 관리하는 것이 좋습니다. 권한 경계를 사용하여 일반적이지는 않지만 사용자에게 있는 권한을 제한할 수 있습니다.
4. (선택 사항) 태그를 연결하여 메타데이터를 사용자에게 추가합니다. IAM에서의 태그 사용에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요.
5. 사용자에게 필요한 로그인 정보를 제공합니다. 여기에는 암호를 비롯해 사용자가 자격 증명을 제공하는 계정 로그인 웹 페이지의 콘솔 URL이 포함됩니다. 자세한 내용은 IAM 사용자가 AWS에 로그인하는 방법 단원을 참조하십시오.
6. (선택 사항) 사용자에 대한 멀티 팩터 인증(MFA)을 구성합니다. MFA의 경우, 사용자가 AWS Management Console에 로그인할 때마다 일회용 코드를 입력해야 합니다.
7. (선택 사항) 사용자에게 자신의 보안 자격 증명을 관리할 권한을 부여합니다. (기본적으로 사용자는 자신의 자격 증명을 관리할 권한이 없습니다.) 자세한 내용은 IAM 사용자에게 자신의 암호 변경 허용 단원을 참조하십시오.

사용자를 생성하기 위해 필요한 권한에 대한 자세한 내용은 IAM 리소스에 액세스하는 데 필요한 권한 섹션을 참조하세요.

주제

- IAM 사용자 생성(콘솔)
- IAM 사용자 생성(AWS CLI)
- IAM 사용자 생성(AWS API)

IAM 사용자 생성(콘솔)

AWS Management Console을 사용하여 IAM 사용자를 생성할 수 있습니다.

IAM 사용자 생성하는 방법(콘솔)

1. AWS 로그인 사용 설명서의 AWS에 로그인하는 방법 항목에 설명된 대로 사용자 유형에 맞는 로그인 절차를 따르세요.
2. 콘솔 홈 페이지에서 IAM 서비스를 선택합니다.
3. 탐색 창에서 사용자와 사용자 생성을 차례로 선택합니다.
4. 사용자 세부 정보 지정 페이지의 사용자 세부 정보 아래에 있는 사용자 이름에 새 사용자의 이름을 입력합니다. 이것은 AWS에 로그인할 때 사용하는 이름입니다.

   > **Note**
   >
   > AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 단원을 참조하십시오. 사용자 이름에는 최대 64개의 문자, 숫자 및 더하기(+), 등호(=), 쉼표(,), 마침표(.), 앳(@) 및 하이픈(-) 조합을 사용할 수 있습니다. 이름은 계정 내에서 고유해야 합니다. 대소문자는 구별하지 않습니다. 예를 들어 "TESTUSER"와 "testuser"라는 두 사용자를 만들 수는 없습니다. 사용자 이름이 정책에서 또는 ARN의 일부로 사용되는 경우 이름은 대소문자를 구분합니다. 콘솔에서 고객에게 사용자 이름이 표시되는 경우(예: 로그인 프로세스 중) 사용자 이름은 대소문자를 구분하지 않습니다.

5. AWS Management Console에 사용자 액세스 제공 선택 사항을 선택하면 새 사용자의 AWS Management Console 로그인 보안 인증이 생성됩니다.

   콘솔 액세스 권한을 제공하려는지 여부를 묻는 메시지가 표시됩니다. IAM이 아니라 IAM Identity Center에서 사용자를 생성하는 것이 좋습니다.

   - IAM Identity Center에서 사용자를 생성하도록 전환하려면 Identity Center에서 사용자 지정을 선택합니다.

     IAM Identity Center를 활성화하지 않은 경우 이 옵션을 선택하면 서비스를 활성화할 수 있도록 콘솔의 서비스 페이지로 이동합니다. 이 절차에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 IAM Identity Center에서 일반 작업 시작하기를 참조하세요.

     IAM Identity Center를 활성화한 경우 이 옵션을 선택하면 IAM Identity Center의 사용자 세부 정보 지정 페이지로 이동합니다. 이 절차에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 사용자 추가를 참조하세요.
   - IAM Identity Center를 사용할 수 없는 경우 IAM 사용자를 생성하고 싶음을 선택하고 절차를 계속 진행하세요.

a. 콘솔 암호의 경우 다음 중 하나를 선택합니다.

- 자동 생성된 비밀 번호 – 각 사용자는 계정 암호 정책에 따라 임의로 생성되는 암호를 받습니다. 암호 검색 페이지에 이르면 암호를 보거나 다운로드할 수 있습니다.
- 사용자 지정 비밀 번호 – 입력란에 입력하는 암호가 각 사용자에게 할당됩니다.

b. (선택 사항) 사용자가 처음으로 로그인할 때 암호를 변경하게 하기 위해, 기본값으로 다음에 로그인할 때 사용자가 새 암호를 생성해야 함(권장)이 선택됩니다.

> **Note**
>
> 관리자가 사용자 자신의 비밀번호 변경 허용 계정 암호 정책 설정을 활성화한 경우 이 확인란은 아무 작업도 수행하지 않습니다. 그렇지 않은 경우에는 새 사용자에게 `IAMUserChangePassword`라는 AWS 관리형 정책이 자동으로 연결됩니다. 이 정책은 사용자에게 자신의 암호를 변경할 수 있는 권한을 부여합니다.

6. 다음을 선택합니다.
7. 권한 설정 페이지에서 이 사용자에 권한을 할당하는 방식을 지정합니다. 다음 세 가지 옵션 중 하나를 선택합니다.

- 그룹에 사용자 추가 – 이미 권한 정책을 보유한 하나 이상의 그룹에 사용자를 할당하고자 하는 경우, 이 옵션을 선택합니다. IAM에 계정의 그룹 목록이 연결된 정책과 함께 표시됩니다. 기존의 보안 그룹을 한 개 이상 선택하거나 그룹 생성을 선택하여 새 그룹을 만들 수 있습니다. 자세한 내용은 IAM 사용자의 권한 변경 단원을 참조하십시오.
- 권한 복사 – 기존 사용자의 그룹 멤버십, 연결된 관리형 정책, 포함된 인라인 정책 및 기존 권한 경계를 새로운 사용자로 모두 복사하려면 이 옵션을 선택합니다. IAM에 계정의 사용자 목록이 표시됩니다. 보유한 권한이 새로운 사용자의 요구 사항과 가장 근접하는 사용자를 선택합니다.
- 직접 정책 연결 – 이 옵션을 선택하여 계정의 AWS 관리형 또는 사용자 관리형 정책 목록을 봅니다. 사용자에게 연결하려는 정책을 선택하거나 정책 생성을 선택하여 새 브라우저 탭을 열고 새로운 정책을 생성합니다. 자세한 내용은 IAM 정책 생성 절차의 4단계 섹션을 참조하세요. 정책을 생성하면 탭을 닫고 원래 탭으로 돌아와 사용자에게 정책을 추가합니다.

> **Tip**
>
> 가능하면 정책을 그룹에 연결한 다음 사용자를 적절한 그룹의 멤버로 만드세요.

8. (선택 사항) 권한 경계를 선택합니다. 이는 고급 기능입니다.

권한 경계 섹션을 열고 최대 권한을 관리하기 위한 권한 경계 사용을 선택합니다. IAM에 계정의 AWS 관리형 또는 사용자 관리형 정책의 목록이 표시됩니다. 권한 경계를 사용하기 위한 정책을 선택하거나 정책 생성을 선택하여 새 브라우저 탭을 열고 새로운 정책을 생성합니다. 자세한 내용은 IAM 정책 생성 절차의 4단계 섹션을 참조하세요. 정책을 생성하면 탭을 닫고 원래 탭으로 돌아와 권한 경계에 사용할 정책을 선택합니다.

9. 다음을 선택합니다.
10. (선택 사항) 검토 및 생성 페이지의 태그에서 새 태그 추가를 선택하여 태그를 키 값 페어로 연결해 메타데이터를 사용자에게 추가합니다. IAM에서의 태그 사용에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요.
11. 이 시점까지 한 선택을 모두 검토합니다. 계속 진행할 준비가 되었으면 사용자 생성을 선택합니다.
12. 비밀번호 검색 페이지에서 사용자에게 할당된 비밀번호를 가져옵니다.

    - 암호 옆에 있는 보기를 선택하여 사용자 암호를 보고 수동으로 기록할 수 있습니다.
    - .csv 다운로드를 선택하여 안전한 위치에 저장할 수 있는.csv 파일로 사용자의 로그인 보안 인증을 다운로드합니다.
13. 이메일 로그인 지침을 선택합니다. 로컬 메일 클라이언트는 사용자 지정을 거쳐 사용자에게 발송할 수 있는 초안 형태로 열립니다. 이메일 템플릿에는 각 사용자에 대한 세부 정보가 다음과 같이 포함되어 있습니다.

    - 사용자 이름
    - 계정 로그인 페이지의 URL. 다음 예를 사용하여 정확한 계정 ID 번호 또는 계정 별칭으로 대치합니다.

    ```
    https://AWS-account-ID or alias.signin.aws.amazon.com/console
    ```

    > **Important**
    >
    > 생성된 이메일에는 사용자 암호가 포함되어 있지 않습니다. 조직의 보안 지침을 준수하는 방식으로 사용자에게 암호를 제공해야 합니다.

14. 사용자에게 프로그래밍 액세스를 위한 액세스 키도 필요한 경우 IAM 사용자의 액세스 키 관리의 내용을 참조하세요.

IAM 사용자 생성(AWS CLI)

AWS CLI를 사용하여 IAM 사용자를 생성할 수 있습니다.

IAM 사용자를 생성하려면(AWS CLI)

1. 사용자를 생성합니다.

   - aws iam create-user

2. (선택 사항) 사용자에게 AWS Management Console에 대한 액세스 권한 부여. 이를 위해서는 암호가 필요합니다. 또한 사용자에게 계정 로그인 페이지의 URL도 제공해야 합니다.

   - aws iam create-login-profile

3. (선택 사항) 사용자에게 프로그래밍 방식 액세스 권한 부여. 이를 위해서는 액세스 키가 필요합니다.

   - aws iam create-access-key
   - Tools for Windows PowerShell: New-IAMAccessKey
   - IAM API: CreateAccessKey

   > **Important**
   >
   > 보안 액세스 키는 이 때만 확인 및 다운로드가 가능하기 때문에 사용자에게 AWS API를 사용하도록 하려면 이 정보를 제공해야 합니다. 사용자의 새 액세스 키 ID와 보안 액세스 키를 안전한 장소에 보관하세요. 이 단계 이후에는 보안 키에 다시 액세스할 수 없습니다.

4. 사용자를 하나 이상의 그룹에 추가합니다. 지정하는 그룹에는 사용자에게 적절한 권한을 부여하는 연결된 정책이 있어야 합니다.

   - aws iam add-user-to-group

5. (선택 사항) 사용자 권한을 정의한 정책을 사용자에게 추가합니다. 주의:사용자에게 직접 정책을 추가하는 대신 그룹에 사용자를 추가하고 그 그룹에 정책을 추가하여 사용자 권한을 관리하시는 것이 좋습니다.

   - aws iam attach-user-policy

6. (선택 사항) 태그를 연결하여 사용자 지정 속성을 사용자에게 추가합니다. 자세한 내용은 IAM 사용자의 태그 관리(AWS CLI 또는 AWS API) 단원을 참조하십시오.

7. (선택 사항) 사용자에게 자신의 보안 자격 증명을 관리할 수 있는 권한을 부여합니다. 자세한 내용은 AWS: MFA 인증 IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용 단원을 참조하십시오.

IAM 사용자 생성(AWS API)

AWS API를 사용하여 IAM 사용자를 생성할 수 있습니다.

AWS API에서 IAM 사용자를 생성하려면

1. 사용자를 생성합니다.

   - CreateUser

2. (선택 사항) 사용자에게 AWS Management Console에 대한 액세스 권한 부여. 이를 위해서는 암호가 필요합니다. 또한 사용자에게 계정 로그인 페이지의 URL도 제공해야 합니다.

   - CreateLoginProfile

3. (선택 사항) 사용자에게 프로그래밍 방식 액세스 권한 부여. 이를 위해서는 액세스 키가 필요합니다.

   - CreateAccessKey

   > **Important**
   >
   > 보안 액세스 키는 이 때만 확인 및 다운로드가 가능하기 때문에 사용자에게 AWS API를 사용하도록 하려면 이 정보를 제공해야 합니다. 사용자의 새 액세스 키 ID와 보안 액세스 키를 안전한 장소에 보관하세요. 이 단계 이후에는 보안 키에 다시 액세스할 수 없습니다.

4. 사용자를 하나 이상의 그룹에 추가합니다. 지정하는 그룹에는 사용자에게 적절한 권한을 부여하는 연결된 정책이 있어야 합니다.

   - AddUserToGroup

5. (선택 사항) 사용자 권한을 정의한 정책을 사용자에게 추가합니다. 주의:사용자에게 직접 정책을 추가하는 대신 그룹에 사용자를 추가하고 그 그룹에 정책을 추가하여 사용자 권한을 관리하시는 것이 좋습니다.

   - AttachUserPolicy

6. (선택 사항) 태그를 연결하여 사용자 지정 속성을 사용자에게 추가합니다. 자세한 내용은 IAM 사용자의 태그 관리(AWS CLI 또는 AWS API) 단원을 참조하십시오.
7. (선택 사항) 사용자에게 자신의 보안 자격 증명을 관리할 수 있는 권한을 부여합니다. 자세한 내용은 AWS: MFA 인증 IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용 단원을 참조하십시오.

IAM 사용자 보기

AWS 계정 또는 특정 IAM 사용자 그룹에 속한 IAM 사용자, 그리고 사용자가 속한 모든 사용자 그룹을 표시할 수 있습니다. 사용자 표시에 필요한 권한에 대한 자세한 내용은 IAM 리소스에 액세스하는 데 필요한 권한 섹션을 참조하세요.

계정에 속한 모든 사용자를 표시하려면

- AWS Management Console: 탐색 창에서 사용자를 선택합니다. 콘솔에 AWS 계정에 속한 사용자가 표시됩니다.
- AWS CLI: aws iam list-users
- AWS API: ListUsers

특정 사용자 그룹에 속한 사용자를 표시하려면

- AWS Management Console: 탐색 창에서 사용자 그룹(User groups)을 선택하고, 사용자 그룹 이름을 선택한 후 사용자(Users) 탭을 선택합니다.
- AWS CLI: aws iam get-group
- AWS API: GetGroup

사용자가 속한 모든 사용자 그룹을 표시하려면

- AWS Management Console: 탐색 창에서 사용자를 선택하고, 사용자 이름을 선택한 후 그룹 탭을 선택합니다.
- AWS CLI: aws iam list-groups-for-user
- AWS API: ListGroupsForUser

다음 단계

IAM 사용자 목록이 생성되면 다음 절차를 사용하여 IAM 사용자의 이름을 바꾸거나, 삭제하거나, 비활성화할 수 있습니다.

- IAM 사용자 이름 바꾸기
- IAM 사용자 삭제 또는 비활성화

IAM 사용자 이름 바꾸기

> **Note**
>
> 가장 좋은 방법은 인간 사용자가 ID 제공업체와의 페더레이션을 사용하여 임시 보안 인증으로 AWS에 액세스하도록 하는 것입니다. 이 방법을 따르는 경우 사용자가 IAM 사용자 및 그룹을 관리하지 않습니다. 대신 사용자와 그룹은 AWS 외부에서 관리되며 페더레이션형 ID로 AWS 리소스에 액세스할 수 있습니다. AWS 페더레이션형 ID는 엔터프라이즈 사용자 디렉터리, 웹 ID 제공업체, Directory Service, Identity Center 디렉터리의 사용자 또는 ID 소스를 통해 제공된 보안 인증을 사용하여 AWS 서비스에 액세스하는 모든 사용자입니다. 페더레이션형 ID는 ID 제공업체에서 정의한 그룹을 사용합니다. AWS IAM Identity Center를 사용하는 경우 IAM Identity Center에서 사용자 및 그룹 생성에 대한 자세한 내용은 AWS IAM Identity Center User Guide(사용 설명서)의 Manage identities in IAM Identity Center(IAM Identity Center에서 ID 관리)를 참조하세요.

Amazon Web Services는 AWS 계정에 속한 IAM 사용자를 관리할 수 있는 다양한 도구를 제공합니다. 계정 또는 사용자 그룹에 속한 IAM 사용자를 나열하거나 사용자가 속한 모든 사용자 그룹을 나열할 수 있습니다. IAM 사용자의 이름을 변경하거나 경로를 변경할 수 있습니다. IAM 사용자 대신 페더레이션형 ID를 사용하도록 전환하는 경우 AWS 계정에서 IAM 사용자를 삭제하거나 사용자를 비활성화할 수 있습니다.

IAM 사용자의 관리형 정책을 추가, 변경 또는 제거하는 방법에 대한 자세한 내용은 IAM 사용자의 권한 변경 섹션을 참조하세요. IAM 사용자의 인라인 정책을 관리하는 방법에 대한 자세한 내용은 IAM 자격 증명 권한 추가 및 제거, IAM 정책 편집, IAM 정책 삭제 섹션을 참조하세요. 가장 좋은 방법은 인라인 정책 대신 관리형 정책을 사용하는 것입니다. AWS 관리형 정책은 많은 일반 사용 사례에 대한 권한을 부여합니다. AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있기 때문에 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 따라서 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한

을 줄이는 것이 좋습니다. 자세한 내용은 AWS 관리형 정책 단원을 참조하십시오. 특정 작업 기능을 위해 설계된 AWS 관리형 정책에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 섹션을 참조하세요.

IAM 정책 검증에 대한 자세한 내용은 IAM 정책 검증 섹션을 참조하세요.

> **Tip**
>
> IAM Access Analyzer는 IAM 역할이 사용하는 서비스와 작업을 분석한 다음 사용할 수 있는 세분화된 정책을 생성할 수 있습니다. 생성된 각 정책을 테스트한 후 프로덕션 환경에 해당 정책을 배포할 수 있습니다. 이렇게 하면 워크로드에 필요한 권한만 부여할 수 있습니다. 정책 생성에 대한 자세한 내용은 IAM Access Analyzer 정책 생성을 참조하세요.

IAM 사용자 암호 관리에 대한 자세한 내용은 IAM 사용자 암호 관리 섹션을 참조하세요.

IAM 사용자 이름 바꾸기

사용자의 이름 또는 경로를 변경하려면 AWS CLI, Tools for Windows PowerShell 또는 AWS API를 사용해야 합니다. 콘솔에서는 사용자의 이름을 변경할 수 있는 옵션이 없습니다. 사용자의 이름을 변경하기 위해 필요한 권한에 대한 자세한 내용은 IAM 리소스에 액세스하는 데 필요한 권한 섹션을 참조하세요.

사용자의 이름 또는 경로를 변경하면 다음과 같이 진행됩니다.

- 사용자에 연결된 정책은 이름이 변경되어도 계속 유지됩니다.
- 사용자는 전과 동일한 사용자 그룹에 새 이름으로 표시됩니다.
- 사용자의 고유 ID는 전과 같습니다. 고유 ID에 대한 자세한 내용은 고유 식별자 섹션을 참조하세요.
- 사용자를 보안 주체로 참조(해당 사용자에게 액세스가 부여됨)하는 리소스 또는 역할 정책은 새 이름 또는 경로를 사용하도록 자동 업데이트됩니다. 예를 들어 Amazon SQS의 대기열 기반 정책 또는 Amazon S3의 리소스 기반 정책은 자동 업데이트되어 새 이름과 경로를 사용합니다.

사용자를 리소스로 참조하는 정책은 새 이름 또는 경로를 사용하도록 IAM에서 자동 업데이트하지 않으므로 수동으로 업데이트해야 합니다. 예를 들어 사용자 `Richard`에게 자신의 보안 자격 증명을 관리하는 정책이 연결되어 있을 경우 관리자가 `Richard`에서 `Rich`로 이름을 변경하면 다음과 같이 리소스가 변경되도록 관리자가 정책도 업데이트해야 합니다.

```
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/Richard
```

다음으로 업데이트:

```
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/Rich
```

마찬가지로 경로를 변경할 경우에도 관리자가 사용자에 대한 새 경로를 반영하도록 정책을 업데이트해야 합니다.

사용자의 이름을 바꾸려면

- AWS CLI: aws iam update-user
- AWS API: UpdateUser

IAM 사용자 삭제 또는 비활성화

사용자가 회사를 그만두면 AWS 계정에서 해당 IAM 사용자를 삭제할 수 있습니다. 사용자가 잠시 회사를 떠난 경우 IAM 사용자 비활성화에 설명된 대로 계정에서 사용자를 삭제하는 대신 사용자의 액세스를 비활성화할 수 있습니다.

사전 조건 - 사용자 액세스 보기

사용자를 삭제하기 전에 최근 서비스 수준 활동을 검토해야 합니다. 이 기능은 사용 중인 보안 주체(사람 또는 애플리케이션)의 액세스 권한을 제거하지 않으려는 경우 중요합니다. 마지막으로 액세스한 정보 보기에 대한 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 섹션을 참조하세요.

IAM 사용자 삭제(콘솔)

AWS Management Console을 사용하여 IAM 사용자를 삭제하면 IAM에서 자동으로 다음 정보를 삭제합니다.

- 해당 사용자
- 모든 사용자 그룹 멤버십(사용자가 멤버였던 모든 IAM 사용자 그룹에서 사용자가 제거됨)
- 사용자와 연결된 모든 암호
- 사용자에게 속한 모든 액세스 키
- 사용자에게 포함된 모든 인라인 정책(사용자 그룹 권한을 통해 사용자에게 적용되는 정책은 영향을 받지 않음)

> **Note**
>
> 사용자를 삭제할 때 IAM은 사용자에 연결된 관리형 정책을 제거하지만 관리형 정책을 삭제하지는 않습니다.

- 연결된 모든 MFA 디바이스

IAM 사용자를 삭제하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 사용자(Users)를 선택한 후 삭제할 사용자 이름 옆에 있는 확인란을 선택합니다.
3. 페이지 상단에서 삭제(Delete)를 선택합니다.
4. 확인 대화 상자에서 텍스트 입력 필드에 사용자 이름을 입력하여 사용자 삭제를 확인합니다. 삭제(Delete)를 선택합니다.

IAM 사용자 삭제(AWS CLI)

AWS Management Console과 달리 AWS CLI로 사용자를 삭제할 때는 사용자에게 연결된 항목들을 수동으로 삭제해야 합니다. 다음 절차는 그 과정을 보여줍니다.

계정에서 사용자를 삭제하려면(AWS CLI)

1. 해당 사용자의 암호가 있으면 삭제합니다.

   `aws iam delete-login-profile`
2. 해당 사용자의 액세스 키가 있으면 삭제합니다.

   `aws iam list-access-keys`(사용자의 액세스 키 나열) 및 `aws iam delete-access-key`
3. 사용자 서명 인증서를 삭제합니다. 보안 자격 증명을 삭제하면 영원히 지워져 검색할 수 없습니다.

   `aws iam list-signing-certificates`(사용자의 서명 인증서 나열) 및 `aws iam delete-signing-certificate`
4. 해당 사용자의 SSH 퍼블릭 키가 있으면 삭제합니다.

   `aws iam list-ssh-public-keys`(사용자의 SSH 퍼블릭 키 나열) 및 `aws iam delete-ssh-public-key`

5. 사용자의 Git 자격 증명을 삭제합니다.

   `aws iam list-service-specific-credentials`(사용자의 Git 자격 증명 나열) 및 `aws iam delete-service-specific-credential`

6. 해당 사용자의 Multi-Factor Authentication(MFA) 디바이스가 있으면 비활성화합니다.

   `aws iam list-mfa-devices`(사용자의 MFA 디바이스 나열), `aws iam deactivate-mfa-device`(디바이스 비활성화) 및 `aws iam delete-virtual-mfa-device`(가상 MFA 디바이스를 영구 삭제)

7. 사용자의 인라인 정책을 삭제합니다.

   `aws iam list-user-policies`(사용자에 대한 인라인 정책 나열) 및 `aws iam delete-user-policy`(정책 삭제)

8. 사용자에 연결된 관리형 정책을 모두 분리합니다.

   `aws iam list-attached-user-policies`(사용자에게 연결된 관리형 정책 나열) 및 `aws iam detach-user-policy`(정책 분리)

9. 모든 사용자 그룹에서 사용자를 제거합니다.

   `aws iam list-groups-for-user`(사용자가 속한 사용자 그룹 나열) 및 `aws iam remove-user-from-group`

10. 사용자를 삭제합니다.

    `aws iam delete-user`

IAM 사용자 비활성화

IAM 사용자가 잠시 회사를 떠나 있는 동안 IAM 사용자를 비활성화해야 할 수 있습니다. IAM 사용자 자격 증명을 그대로 두고 AWS 액세스를 계속 차단할 수 있습니다.

사용자를 비활성화하려면 AWS에 대한 사용자 액세스를 거부하는 정책을 생성하고 연결합니다. 나중에 사용자의 액세스 권한을 복원할 수 있습니다.

다음은 액세스를 거부하기 위해 사용자에게 연결할 수 있는 거부 정책의 두 가지 예입니다.

다음 정책에는 시간 제한이 포함되지 않습니다. 사용자의 액세스 권한을 복원하려면 정책을 제거해야 합니다.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
      {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*"
      }
   ]
}
```

다음 정책에는 2024년 12월 24일 오후 11:59(UTC)에 시작하여 2025년 2월 28일 오후 11:59(UTC)에 종료하는 조건이 포함되어 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
          "DateGreaterThan": {"aws:CurrentTime": "2024-12-24T23:59:59Z"},
          "DateLessThan": {"aws:CurrentTime": "2025-02-28T23:59:59Z"}
          }
       }
   ]
}
```

AWS Management Console에 대한 IAM 사용자 액세스 제어

AWS Management Console을 통해 AWS 계정에 로그인하는 권한이 있는 IAM 사용자는 AWS 리소스에 액세스할 수 있습니다. 다음 목록에서는 AWS Management Console을 통해 AWS 계정 리소스에 대한 액세스 권한을 IAM 사용자에게 부여하는 방법을 보여 줍니다. 또한 IAM 사용자가 AWS 웹 사이트를 통해 다른 AWS 계정 기능에 액세스할 보여 줍니다.

> **Note**
>
> IAM 사용에는 요금이 부과되지 않습니다.

AWS Management Console은

AWS Management Console에 액세스해야 하는 각 IAM 사용자에 대해 암호를 만듭니다. 사용자는 IAM 지원 AWS 계정 로그인 페이지를 통해 콘솔에 액세스합니다. 로그인 페이지 액세스에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 로그인 방법을 참조하세요. 암호 만들기에 대한 자세한 내용은 AWS에서 사용자 암호 관리을 참조하세요.

사용자의 암호를 제거하여 IAM 사용자가 AWS Management Console에 액세스하지 못하도록 할 수 있습니다. 그러면 사용자는 자신의 로그인 보안 인증을 사용해 AWS Management Console에 로그인할 수 없습니다. 그렇더라도 사용자의 권한이 변경되거나 위임된 역할을 사용하여 콘솔에 액세스하지 못하게 되지는 않습니다. 사용자에게 활성 액세스 키가 있다면 해당 키는 계속 작동하고 AWS CLI, Tools for Windows PowerShell 또는 AWS API 또는 AWS Console Mobile Application을 통해 액세스를 허용합니다.

Amazon EC2 인스턴스, Amazon S3 버킷 등의 AWS 리소스

IAM 사용자에게 암호가 있더라도 AWS 리소스에 액세스하려면 권한이 필요합니다. IAM 사용자를 만들 때 이 사용자에게는 기본적으로 권한이 없습니다. IAM 사용자에게 필요한 권한을 부여하려면 해당 사용자에게 정책을 연결합니다. 같은 리소스로 같은 작업을 수행할 IAM 사용자가 많은 경우 해당 IAM 사용자를 그룹에 할당할 수 있습니다. 이 그룹에 권한을 할당할 수 있습니다. IAM 사용자 및 그룹 만들기에 대한 자세한 내용은 IAM 자격 증명(사용자, 사용자 그룹 및 역할)을 참조하세요. 권한 설정을 위한 정책 사용에 대한 자세한 내용은 AWS리소스에 대한 액세스 관리을 참조하세요.

AWS 토론 포럼

누구나 AWS 토론 포럼에서 게시물을 읽을 수 있습니다. AWS 토론 포럼에 질문이나 의견을 게시하고자 하는 사용자는 자신의 사용자 이름을 사용하여 그렇게 할 수 있습니다. 사용자가 처음으로 AWS 토론 포럼에 게시하면 별칭과 이메일 주소를 입력하라는 메시지가 표시됩니다. 해당 사용자만 AWS 토론 포럼에서 해당 별칭을 사용할 수 있습니다.

AWS 계정 결제 및 사용 정보

AWS 계정 결제 및 사용 정보에 대한 액세스 권한을 사용자에게 부여할 수 있습니다. 자세한 내용은 AWS Billing 사용 설명서의 결제 정보에 대한 액세스 제어를 참조하세요.

AWS 계정 프로필 정보

사용자는 계정 소유자의 AWS 계정 프로필 정보에 액세스할 수 없습니다.

AWS 계정 보안 인증 정보

사용자는 계정 소유자의 AWS 계정 보안 인증 정보에 액세스할 수 없습니다.

> **Note**
>
> IAM 정책은 인터페이스와 관계없이 액세스를 제어합니다. 예를 들어 AWS Management Console에 액세스하기 위한 암호를 사용자에게 제공할 수 있습니다. 해당 사용자(또는 사용자가 속한 그룹)에 대한 정책은 사용자가 AWS Management Console에서 수행할 수 있는 작업을 제어합니다. 또는 AWS에 대해 API 호출을 실행하기 위한 AWS 액세스 키를 사용자에게 제공할 수 있습니다. 이렇게 하면 인증을 위해 해당 액세스 키를 사용하는 라이브러리 또는 클라이언트를 통해 사용자가 호출할 수 있는 작업이 정책을 통해 제어됩니다.

IAM 사용자의 권한 변경

AWS 계정에서 IAM 사용자의 권한을 변경하려면 사용자의 그룹 멤버십을 변경하거나, 기존 사용자의 권한을 복사하거나, 사용자에 정책을 직접 연결하거나, 권한 경계를 설정할 수 있습니다. 이 권한 경계는 사용자가 가질 수 있는 최대 권한을 관리합니다. 권한 경계는 고급 AWS 기능입니다.

사용자의 권한을 수정하기 위해 필요한 권한에 대한 자세한 내용은 IAM 리소스에 액세스하는 데 필요한 권한 섹션을 참조하세요.

주제

- 사용자 액세스 보기
- 사용자의 액세스 활동을 기반으로 정책 생성
- 사용자에 권한 추가(콘솔)
- 사용자의 권한 변경(콘솔)
- 사용자에게서 권한 정책 제거(콘솔)
- 사용자에게서 권한 경계 제거(콘솔)
- 사용자 권한 추가 및 제거(AWS CLI 또는 AWS API)

사용자 액세스 보기

사용자에 대한 권한을 변경하기 전에 최근 서비스 수준 활동을 검토해야 합니다. 이 기능은 사용 중인 보안 주체(사람 또는 애플리케이션)의 액세스 권한을 제거하지 않으려는 경우 중요합니다. 마지막으로 액세스한 정보 보기에 대한 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 섹션을 참조하세요.

사용자의 액세스 활동을 기반으로 정책 생성

때로는 IAM 엔터티(사용자 또는 역할)에 필요한 것보다 많은 권한을 부여할 수도 있습니다. 부여하는 권한을 세분화할 수 있도록 엔터티의 액세스 활동을 기반으로 IAM 정책을 생성할 수 있습니다. IAM Access Analyzer는 사용자의 AWS CloudTrail 로그를 검토하고 지정된 날짜 범위에 엔터티에서 사용한 권한을 포함하는 정책 템플릿을 생성합니다. 이 템플릿을 사용하여 세분화된 권한이 포함된 관리형 정책을 생성한 다음 IAM 엔터티에 연결할 수 있습니다. 이렇게 하면 특정 사용 사례에 사용자나 역할이 AWS 리소스와 상호 작용하는 데 필요한 권한만 부여됩니다. 자세한 내용은 액세스 활동을 기반으로 정책 생성 섹션을 참조하세요.

사용자에 권한 추가(콘솔)

IAM은 사용자에 권한 정책을 추가하는 세 가지 방법을 제공합니다.

- 그룹에 사용자 추가 - 사용자를 그룹의 구성원으로 만듭니다. 그룹의 정책은 사용자로 연결됩니다.
- 기존 사용자의 권한 복사 - 소스 사용자의 모든 그룹 멤버십, 연결된 관리형 정책, 인라인 정책 및 모든 기존 권한 경계를 복사합니다.
- 정책을 사용자에 직접 연결 - 관리형 정책을 사용자에 직접 연결합니다. 더 쉬운 권한 관리를 위해 정책을 그룹에 연결한 다음 사용자를 적절한 그룹의 멤버로 만드세요.

> **Important**
>
> 사용자에게 권한 경계가 있다면 권한 경계가 허용한 권한보다 더 많은 권한을 추가할 수 없습니다.

사용자를 그룹에 추가하여 권한 추가

사용자를 그룹에 추가하여 사용자에게 바로 영향을 줍니다.

사용자를 그룹에 추가하여 사용자에게 권한을 추가하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 콘솔의 그룹 열에서 사용자에 대한 현재 그룹 멤버십을 검토합니다. 필요할 경우 다음 단계를 통해 사용자 테이블에 열을 추가합니다.

1. 테이블 위 맨 오른쪽에서 설정 기호
   ( )
   를 선택합니다.
2. 열 관리(Manage Columns) 대화 상자에서 그룹(Groups) 열을 선택합니다. 필요할 경우 사용자 테이블에 표시하지 않으려는 열이 있으면 해당 열의 확인란 선택을 취소하면 됩니다.
3. 닫기를 선택하여 사용자 목록으로 돌아갑니다.

그룹 열에는 사용자가 속한 그룹이 표시됩니다. 열에는 최대 2개 그룹에 대한 그룹 이름이 표시됩니다. 사용자가 3개 이상 그룹의 구성원인 경우 처음 두 개 그룹만 알파벳 순서대로 표시되고 나머지 그룹 멤버십 수가 표시됩니다. 예를 들어 사용자가 그룹 A, 그룹 B, 그룹 C, 그룹 D에 속한 경우 필드에 Group A, Group B + 2 more(그룹 A, 그룹 B 외 2개)라고 표시됩니다. 사용자가 속한 총 그룹 수를 보려면 사용자 테이블에 Group count(그룹 수) 열을 추가합니다.

4. 권한을 수정하려는 사용자의 이름을 선택합니다.
5. 권한 탭을 선택한 다음 Add permissions(권한 추가)를 선택합니다. 사용자를 그룹에 추가를 선택합니다.
6. 사용자를 귀속시키려는 각 그룹의 확인란을 선택합니다. 그 목록에는 각 그룹의 이름과 사용자가 그 그룹의 구성원이 되면 받는 정책이 표시됩니다.
7. (선택 사항) 기존 그룹에서 선택할 수 있을 뿐 아니라 다음과 같이 그룹 생성을 선택하여 새 그룹을 정의할 수 있습니다.

   a. 새로운 탭에서 사용자 그룹 이름으로 새로운 그룹의 이름을 입력합니다.

   > **Note**
   >
   > AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 단원을 참조하십시오. 그룹 이름에는 최대 128개의 알파벳, 숫자 및 더하기(+), 등호(=), 쉼표(,), 마침표(.), 앳(@), 그리고 하이픈(-) 조합을 사용할 수 있습니다. 이름은 계정 내에서 고유해야 합니다. 대소문자는 구별하지 않습니다. 예를 들어 "TESTGROUP"과 "testgroup"이라는 두 그룹을 만들 수는 없습니다.

   b. 그룹에 연결하고자 하는 관리형 정책에 대해 한 개 이상의 확인란을 선택합니다. 정책 생성을 선택하여 새로운 관리형 정책을 만들 수도 있습니다. 이렇게 하는 경우, 새 정책이 완료되면 이 브라우저 탭 또는 창으로 돌아가 새로 고침을 선택한 다음, 그룹에 연결할 새로운 정책을 선택합니다. 자세한 내용은 고객 관리형 정책으로 사용자 지정 IAM 권한 정의 단원을 참조하십시오.

c. 사용자 그룹 만들기를 선택합니다.

d. 기존 탭으로 반환하고 그룹 목록을 새로 고침합니다. 새로운 그룹에 대한 확인란을 선택합니다.

8. 다음을 선택하여 사용자에 추가될 그룹 멤버십의 목록을 확인합니다. 그런 다음 Add permissions(권한 추가)를 선택합니다.

다른 사용자에게서 복사하여 권한 추가

권한 복사는 사용자에게 바로 적용됩니다.

다른 사용자에게서 권한을 복사하여 사용자에게 권한을 추가하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택한 다음, 권한을 수정할 사용자의 이름을 선택하고 권한 탭을 선택합니다.
3. Add permissions(권한 추가)를 선택한 다음, Copy permissions from existing user(기존 사용자에서 권한 복사)를 선택합니다. 목록에는 사용 가능한 사용자들이 그들의 그룹 멤버십 및 연결된 정책과 함께 표시됩니다. 그룹 또는 정책의 전체 목록이 한 줄에 표시되지 않는 경우 및 ***n***개 더(and n more) 링크를 선택할 수 있습니다. 그러면 새 브라우저 탭이 열리고 정책(권한 탭) 및 그룹(그룹 탭)의 전체 목록을 볼 수 있습니다.
4. 복사하고자 하는 권한을 보유한 사용자 옆에 있는 라디오 버튼을 선택합니다.
5. 다음을 선택하여 사용자에 대한 변경 사항의 목록을 확인합니다. 그런 다음 Add permissions(권한 추가)를 선택합니다.

사용자에 정책을 직접 연결하여 권한 추가

정책 연결은 사용자에게 바로 적용됩니다.

관리형 정책을 직접 연결하여 사용자에게 권한을 추가하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택한 다음, 권한을 수정할 사용자의 이름을 선택하고 권한 탭을 선택합니다.
3. 권한 추가를 선택한 다음, 정책 직접 연결을 선택합니다.

4. 사용자에 연결하고자 하는 관리형 정책에 대해 한 개 이상의 확인란을 선택합니다. 정책 생성을 선택하여 새로운 관리형 정책을 만들 수도 있습니다. 이렇게 하는 경우, 새 정책이 완료되면 이 브라우저 탭 또는 창으로 돌아가 새로 고침을 선택한 다음, 사용자에게 연결할 새로운 정책 확인란을 선택합니다. 자세한 내용은 고객 관리형 정책으로 사용자 지정 IAM 권한 정의 단원을 참조하십시오.
5. 다음을 선택하여 사용자에 연결될 정책의 목록을 확인합니다. 그런 다음 Add permissions(권한 추가)를 선택합니다.

사용자의 권한 경계 설정

권한 경계 설정은 사용자에게 바로 적용됩니다.

사용자에 대한 권한 경계를 설정하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 권한 경계를 변경하려는 사용자의 이름을 선택합니다.
4. 권한 탭을 선택합니다. 필요하다면 권한 경계 섹션을 열고 권한 경계 설정을 선택합니다.
5. 정책을 선택하여 원하는 권한 경계를 사용하세요.
6. Set boundary(경계 설정)을 선택합니다.

사용자의 권한 변경(콘솔)

IAM을 사용하면 다음과 같은 방법으로 사용자와 연결된 권한을 변경할 수 있습니다.

- 권한 정책 편집 - 사용자 인라인 정책, 사용자 그룹의 인라인 정책을 편집하거나 사용자에 직접 연결되거나 그룹에서 연결된 관리형 정책을 편집합니다. 사용자에게 권한 경계가 있다면 권한 경계로 사용된 정책이 허용한 권한보다 더 많은 권한을 제공할 수 없습니다.
- 권한 경계 변경 - 사용자의 권한 경계로 사용되는 정책을 변경합니다. 이로써 사용자가 가질 수 있는 최대 권한을 확장 또는 제한할 수 있습니다.

사용자에 연결된 권한 정책 편집

권한 변경은 사용자에게 바로 적용됩니다.

사용자의 연결된 관리형 정책을 편집하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 권한 정책을 변경하려는 사용자의 이름을 선택합니다.
4. 권한 탭을 선택합니다. 필요하다면 Permissions policies(권한 정책) 부분을 엽니다.
5. 정책에 대한 세부 정보를 보기 위해서 편집하고자 하는 정책 이름을 선택합니다. 정책 사용 탭을 선택하여 정책을 편집함으로써 영향을 받을 다른 개체를 봅니다.
6. 그런 다음 Permissions tab(권한 탭)을 정책이 허용한 권한을 검토합니다. 그런 다음 정책 편집을 선택합니다.
7. 정책을 편집하고 모든 정책 검사 권장 사항을 해결합니다. 자세한 내용은 IAM 정책 편집 단원을 참조하십시오.
8. 정책 검토를 선택한 다음 정책 요약을 검토한 후 변경 사항 저장을 선택합니다.

사용자의 권한 경계 변경

권한 경계 변경은 사용자에게 바로 적용됩니다.

사용자의 권한 경계 설정에 사용된 정책을 변경하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 권한 경계를 변경하려는 사용자의 이름을 선택합니다.
4. 권한 탭을 선택합니다. 필요하다면 Permissions boundary(권한 경계) 섹션을 열고 Change boundary(경계 변경)을 선택합니다.
5. 정책을 선택하여 원하는 권한 경계를 사용하세요.
6. Set boundary(경계 설정)을 선택합니다.

사용자에게서 권한 정책 제거(콘솔)

정책 제거는 사용자에게 바로 적용됩니다.

IAM 사용자의 권한을 제거하는 방법

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 권한 경계를 제거하려는 사용자의 이름을 선택합니다.
4. 권한 탭을 선택합니다.
5. 기존 정책을 제거하여 권한을 제거하려면 제거를 선택하여 정책을 제거하기 전에 유형을 보고 사용자가 해당 정책을 가져오는 방법을 파악합니다.

   - 그 정책이 그룹 멤버십 때문에 적용되는 경우, 제거를 선택하면 사용자가 그룹에서 제거됩니다. 한 그룹에 여러 정책이 연결될 수 있습니다. 따라서 그룹에서 사용자를 제거할 경우 사용자는 그 그룹의 멤버십을 통해 받은 모든 정책에 대한 액세스 권한을 잃게 됩니다.
   - 정책이 사용자에 직접 연결된 관리형 정책인 경우 제거를 선택하면 정책이 사용자와 분리됩니다. 이렇게 해도 정책 자체 또는 그 정책이 연결되어 있을 수 있는 다른 개체에는 영향을 미치지 않습니다.
   - 정책이 인라인 포함 정책인 경우, X를 선택하면 정책이 IAM에서 제거됩니다. 사용자에 직접 연결된 인라인 정책은 해당 사용자에만 존재합니다.

사용자에게서 권한 경계 제거(콘솔)

권한 경계 제거는 사용자에게 바로 적용됩니다.

사용자(콘솔)에게서 권한 경계를 제거하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 권한 경계를 제거하려는 사용자의 이름을 선택합니다.
4. 권한 탭을 선택합니다. 필요하다면 Permissions boundary(권한 경계) 섹션을 열고 Remove boundary(경계 제거)를 선택합니다.
5. 경계 제거를 선택하여 권한 경계를 제거합니다.

사용자 권한 추가 및 제거(AWS CLI 또는 AWS API)

프로그래밍 방식으로 권한을 추가 또는 제거하려면 그룹 멤버십을 추가 또는 제거하거나 관리형 정책을 연결 또는 분리하거나 인라인 정책을 추가 또는 삭제해야 합니다. 자세한 정보는 다음 주제를 참조하세요.

- IAM 사용자 그룹의 사용자 편집
- IAM 자격 증명 권한 추가 및 제거

AWS에서 사용자 암호 관리

계정 IAM 사용자에 대한 암호를 관리할 수 있습니다. IAM 사용자는 AWS Management Console에 액세스하기 위해 암호가 필요합니다. 사용자가 AWS CLI, Tools for Windows PowerShell, AWS SDK 또는 API를 사용하여 프로그래밍 방식으로 AWS 리소스에 액세스하는 경우 암호가 필요 없습니다. 이러한 환경의 경우 IAM 사용자에게 액세스 키를 할당하는 옵션이 있습니다. 그러나 액세스 키에 대한 다른 더 안전한 대안이 있습니다. 먼저 이를 고려해 보는 것이 좋습니다. 자세한 내용은 AWS 보안 인증 정보 단원을 참조하십시오.

내용

- IAM 사용자의 계정 암호 정책 설정
- IAM 사용자 암호 관리
- IAM 사용자에게 자신의 암호 변경 허용
- IAM 사용자가 자신의 암호를 변경하는 방법

IAM 사용자의 계정 암호 정책 설정

AWS 계정에서 사용자 지정 암호 정책을 설정하여 IAM 사용자 암호의 복잡성 요건과 의무적인 교체 주기를 지정할 수 있습니다. 사용자 지정 암호 정책을 설정하지 않은 경우 IAM 사용자 암호는 기본 AWS 암호 정책을 충족해야 합니다. 자세한 내용은 사용자 지정 암호 정책 옵션 단원을 참조하십시오.

주제

- 암호 정책 설정에 대한 규칙
- 암호 정책을 설정하는 데 필요한 권한
- 기본 암호 정책
- 사용자 지정 암호 정책 옵션
- 암호 정책 설정(콘솔)

- 암호 정책 설정(AWS CLI)
- 암호 정책 설정(AWS API)

암호 정책 설정에 대한 규칙

IAM 암호 정책은 AWS 계정 루트 사용자 암호 또는 IAM 사용자 액세스 키에 적용되지 않습니다. 암호가 만료되면 IAM 사용자는 AWS Management Console에 로그인할 수 없지만 액세스 키를 계속 사용할 수 있습니다.

암호 정책을 생성 또는 변경하더라도 대부분의 암호 정책 설정은 사용자가 다음에 자신의 암호를 변경할 때 적용됩니다. 하지만 일부 설정은 바로 적용됩니다. 예:

- 최소 길이 및 문자 유형 요건이 변경되는 경우 이러한 설정은 다음 번에 사용자가 자신의 암호를 변경할 때 적용됩니다. 기존 암호가 업데이트된 암호 정책을 따르지 않는 경우에도 사용자들은 기존 암호를 변경할 필요는 없습니다.
- 암호 만료 기간을 설정하면 만료 기간이 바로 적용됩니다. 예를 들어 암호 만료 기간을 90일로 설정한 경우, 기존 암호가 90일 이상된 모든 IAM 사용자의 암호가 만료됩니다. 이러한 사용자는 다음에 로그인할 때 암호를 변경해야 합니다.

지정된 횟수의 로그인 시도가 실패한 후에는 계정에서 사용자를 잠그는 "잠금 정책'을 생성할 수 없습니다. 보안을 강화하려면 멀티 팩터 인증(MFA)과 함께 강력한 암호 정책을 사용하는 것이 좋습니다. MFA에 대한 자세한 내용은 IAM의 AWS 다중 인증 섹션을 참조하세요.

암호 정책을 설정하는 데 필요한 권한

IAM 엔터티(사용자 또는 역할)가 계정 암호 정책을 보거나 편집하도록 허용하려면 권한을 구성해야 합니다. IAM 정책에 다음과 같은 암호 정책 작업을 포함할 수 있습니다.

- `iam:GetAccountPasswordPolicy` - 엔터티가 계정의 암호 정책을 볼 수 있도록 허용합니다.
- `iam:DeleteAccountPasswordPolicy` - 엔터티가 계정의 사용자 지정 암호 정책을 삭제하고 기본 암호 정책으로 되돌릴 수 있도록 허용합니다.
- `iam:UpdateAccountPasswordPolicy` - 엔터티가 계정의 사용자 지정 암호 정책을 생성하거나 변경할 수 있도록 허용합니다.

다음 정책은 계정 암호 정책을 보고 편집할 수 있는 모든 액세스 권한을 허용합니다. 이 예제 JSON 정책 문서를 사용하여 IAM 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessPasswordPolicy",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy",
                "iam:DeleteAccountPasswordPolicy",
                "iam:UpdateAccountPasswordPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

IAM 사용자가 자신의 암호를 변경하는 데 필요한 권한에 대한 자세한 내용은 IAM 사용자에게 자신의 암호 변경 허용 섹션을 참조하세요.

기본 암호 정책

관리자가 사용자 지정 암호 정책을 설정하지 않은 경우 IAM 사용자 암호는 기본 AWS 암호 정책을 충족해야 합니다.

기본 암호 정책은 다음 조건을 적용합니다.

- 최소 8자부터 최대 128자의 암호 길이
- 대문자, 소문자, 숫자 및 영숫자가 아닌 문자(! @ # $ % ^ & * ( ) _ + - = [ ] { } | ') 중에서 세 가지 이상의 문자 유형 혼합
- AWS 계정 이름 또는 이메일 주소와 동일하지 않아야 함
- 암호 만료 안 함

사용자 지정 암호 정책 옵션

계정에 대한 사용자 지정 암호 정책을 구성할 때 다음 조건을 지정할 수 있습니다.

- 암호 최소 길이 - 최소 6자, 최대 128자를 지정할 수 있습니다.
- 암호 강도 - 다음 확인란을 선택하여 IAM 사용자 암호의 강도를 정의할 수 있습니다.
  - 라틴 문자에서 하나 이상의 대문자 필요(A~Z)

  - 라틴 문자에서 하나 이상의 소문자 필요(a~z)
  - 1개 이상의 숫자 필수
  - 영숫자가 아닌 하나 이상의 문자 필요(`! @ # $ % ^ & * ( ) _ + - = [ ] { } | '`)
- Turn on password expiration(암호 만료 활성화) - IAM 사용자 암호가 설정된 후 유효한 기간(최소 1일부터 최대 1,095일)을 선택하여 지정할 수 있습니다. 예를 들어 만료일을 90일로 지정하면 모든 사용자에게 즉시 영향을 미칩니다. 암호를 변경한 지 90일 이상인 사용자의 경우, 콘솔에 로그인할 때 새 암호를 설정해야 합니다. 75~89일이 지난 암호를 사용 중인 사용자는 암호 만료에 대한 AWS Management Console 경고를 수신합니다. 권한이 있는 IAM 사용자는 언제든지 자신의 암호를 변경할 수 있습니다. 새 암호를 설정하면 암호 만료 기간이 다시 시작됩니다. IAM 사용자는 한 번에 유효 암호 하나만 사용할 수 있습니다.
- 암호 만료 시 관리자 재설정 필요(Password expiration requires administrator reset) - 암호가 만료된 후 IAM 사용자가 AWS Management Console을 사용하여 자신의 암호를 업데이트하지 못하도록 하려면 이 옵션을 선택합니다. 이 옵션을 선택하기 전에 AWS 계정에 IAM 사용자 암호 재설정을 위한 관리자 권한을 가진 사용자가 2명 이상인지 확인해야 합니다. `iam:UpdateLoginProfile` 권한이 있는 관리자는 IAM 사용자 암호를 재설정할 수 있습니다. `iam:ChangePassword` 권한과 활성 액세스 키가 있는 IAM 사용자는 프로그래밍 방식으로 자신의 IAM 사용자 콘솔 암호를 재설정할 수 있습니다. 이 확인란의 선택을 취소하면 암호가 만료된 IAM 사용자가 AWS Management Console에 액세스하기 전에 새 암호를 설정해야 합니다.
- 사용자 자신의 암호 변경 허용(Allow users to change their own password) - 계정의 모든 IAM 사용자가 자신의 암호를 변경하도록 허용할 수 있습니다. 이렇게 하면 사용자에게 해당 사용자의 `iam:ChangePassword` 작업과 `iam:GetAccountPasswordPolicy` 작업에 대한 액세스 권한만 부여됩니다. 이 옵션은 각 사용자에게 권한 정책을 연결하지 않습니다. 대신, IAM은 모든 사용자에 대해 계정 수준에서 권한을 적용합니다. 또는 일부 사용자만 자신의 암호를 관리하도록 허용할 수 있습니다. 이렇게 하려면 이 확인란을 선택 취소합니다. 암호 관리 제한 정책의 사용에 대한 자세한 내용은 IAM 사용자에게 자신의 암호 변경 허용 섹션을 참조하세요.
- 암호 재사용 제한 - IAM 사용자가 지정된 수의 이전 암호를 재사용하지 못하도록 제한할 수 있습니다. 최소 1부터 최대 24개의 반복할 수 없는 이전 암호를 지정할 수 있습니다.

암호 정책 설정(콘솔)

AWS Management Console에서 사용자 지정 암호 정책을 생성, 변경 또는 삭제할 수 있습니다.

사용자 지정 암호 정책을 생성하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.

2. 탐색 창에서 계정 설정(Account settings)를 선택합니다.
3. Password policy(암호 정책) 섹션에서 Edit(편집)를 선택합니다.
4. Custom(사용자 지정)을 선택하여 사용자 지정 암호 정책을 사용합니다.
5. 암호 정책에 적용할 옵션을 선택하고 [변경 사항 저장(Save changes)]을 선택합니다.
6. Set custom(사용자 지정 설정)을 선택하여 사용자 지정 암호 정책의 설정을 확인합니다.

사용자 지정 암호 정책을 변경하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 계정 설정(Account settings)를 선택합니다.
3. Password policy(암호 정책) 섹션에서 Edit(편집)를 선택합니다.
4. 암호 정책에 적용할 옵션을 선택하고 [변경 사항 저장(Save changes)]을 선택합니다.
5. Set custom(사용자 지정 설정)을 선택하여 사용자 지정 암호 정책의 설정을 확인합니다.

사용자 지정 암호 정책을 삭제하려면 다음을 수행하세요(콘솔).

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 계정 설정(Account settings)를 선택합니다.
3. Password policy(암호 정책) 섹션에서 Edit(편집)를 선택합니다.
4. IAM default(IAM 기본값)를 선택하여 사용자 지정 암호 정책을 삭제하고 Save changes(변경 사항 저장)를 선택합니다.
5. Set default(기본값 설정)를 선택하여 IAM 기본 암호 정책의 설정을 확인합니다.

암호 정책 설정(AWS CLI)

AWS Command Line Interface를 사용하여 암호 정책을 설정할 수 있습니다.

AWS CLI에서 사용자 지정 계정 암호 정책을 관리하려면

다음 명령을 실행합니다.

- 사용자 지정 암호 정책을 생성하거나 변경하려면: `aws iam update-account-password-policy`

- 암호 정책을 보려면: `aws iam get-account-password-policy`
- 사용자 지정 암호 정책을 삭제하려면: `aws iam delete-account-password-policy`

암호 정책 설정(AWS API)

AWS API 작업을 사용하여 암호 정책을 설정할 수 있습니다.

AWS API에서 사용자 지정 계정 암호 정책을 관리하려면

다음 연산을 호출합니다.

- 사용자 지정 암호 정책을 생성하거나 변경하려면: `UpdateAccountPasswordPolicy`
- 암호 정책을 보려면: `GetAccountPasswordPolicy`
- 사용자 지정 암호 정책을 삭제하려면: `DeleteAccountPasswordPolicy`

IAM 사용자 암호 관리

AWS Management Console을 사용하여 AWS 리소스를 작업하는 IAM 사용자가 로그인하려면 암호가 필요합니다. AWS 계정에 속한 IAM 사용자의 암호를 생성, 변경 또는 삭제할 수 있습니다.

사용자에게 암호를 할당한 후 사용자는 다음과 같은 계정의 로그인 URL을 사용하여 AWS Management Console에 로그인할 수 있습니다.

```
https://12-digit-AWS-account-ID or alias.signin.aws.amazon.com/console
```

IAM 사용자가 AWS Management Console에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS에 로그인하는 방법을 참조하세요.

사용자에게 암호가 있더라도 AWS 리소스에 액세스하려면 권한이 필요합니다. 기본적으로 사용자에게는 권한이 없습니다. 사용자에게 필요한 권한을 부여하려면 해당 사용자 또는 사용자가 속한 그룹에 정책을 할당합니다. 사용자 및 그룹 만들기에 대한 자세한 내용은 IAM 자격 증명(사용자, 사용자 그룹 및 역할)을 참조하세요. 권한 설정을 위한 정책 사용에 대한 자세한 내용은 IAM 사용자의 권한 변경을 참조하세요.

자신의 암호를 변경할 권한을 사용자에게 부여할 수 있습니다. 자세한 내용은 IAM 사용자에게 자신의 암호 변경 허용 단원을 참조하십시오. 사용자가 계정 로그인 페이지에 액세스하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS에 로그인하는 방법을 참조하세요.

주제

- IAM 사용자 암호 생성, 변경 또는 삭제(콘솔)
- IAM 사용자 암호 생성, 변경 또는 삭제(AWS CLI)
- IAM 사용자 암호 생성, 변경 또는 삭제(AWS API)

IAM 사용자 암호 생성, 변경 또는 삭제(콘솔)

AWS Management Console을 사용하여 IAM 사용자의 암호를 관리할 수 있습니다.

사용자가 조직을 떠나거나 AWS 액세스가 더 이상 필요하지 않은 경우 사용 중인 자격 증명을 찾아서 더 이상 작동하지 않도록 해야 합니다. 더 이상 필요 없는 자격 증명을 삭제하는 것이 가장 좋습니다. 나중에 필요한 경우가 생기면 언제든지 다시 생성할 수 있습니다. 적어도 그 자격 증명을 변경하여 이전 사용자가 더 이상 액세스할 수 없게 해야 합니다.

IAM 사용자의 암호를 추가하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 암호를 생성하려는 사용자의 이름을 선택합니다.
4. 보안 자격 증명 탭을 선택한 다음 콘솔 로그인에서 콘솔 액세스 활성화를 선택합니다.
5. 콘솔 액세스 활성화에서 콘솔 암호에 대해 IAM이 암호를 자동으로 생성할지, 아니면 사용자 지정 암호를 만들지를 선택합니다.
   - IAM에서 암호를 자동으로 생성하려면 자동 생성 암호(Autogenerated password)를 선택합니다.
   - 사용자 지정 암호를 만들려면 사용자 지정 암호(Custom password)를 선택하고 암호를 입력합니다.

   > **Note**
   > 생성하는 암호는 계정의 암호 정책을 충족해야 합니다.

6. 사용자가 로그인할 때 새 암호를 만들도록 요구하려면 다음 로그인 시 사용자가 새 암호를 생성해야 함을 선택합니다. 그런 다음 콘솔 액세스 활성화를 선택합니다.

> **Important**
>
> 다음 로그인 시 사용자가 새 암호를 생성해야 함 옵션을 선택할 경우, 사용자에게 자신의 암호를 변경할 권한이 있는지 확인하세요. 자세한 내용은 IAM 사용자에게 자신의 암호 변경 허용 단원을 참조하십시오.

7. 암호를 확인하고 사용자와 공유하려면 콘솔 암호 대화 상자에서 표시를 선택합니다.

> **Important**
>
> 이 단계를 완료한 후에는 보안상의 이유로 암호에 액세스할 수 없지만, 언제든지 새 암호를 만들 수 있습니다.

IAM 사용자의 암호를 변경하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 암호를 변경할 사용자의 이름을 선택합니다.
4. 보안 자격 증명 탭을 선택한 다음 콘솔 로그인에서 콘솔 액세스 관리를 선택합니다.
5. 콘솔 액세스 관리에서 암호 재설정을 아직 선택하지 않은 경우 암호 재설정을 선택합니다. 콘솔 액세스가 비활성화되어 있는 경우에는 암호가 필요 없습니다.
6. 콘솔 액세스에서 IAM이 암호를 자동으로 생성할지, 아니면 사용자 지정 암호를 만들지를 선택합니다.
   - IAM에서 암호를 자동으로 생성하려면 자동 생성 암호(Autogenerated password)를 선택합니다.
   - 사용자 지정 암호를 만들려면 사용자 지정 암호(Custom password)를 선택하고 암호를 입력합니다.

   > **Note**
   >
   > 만드는 암호는 계정의 암호 정책(정책을 설정한 경우)에 부합해야 합니다.
7. 사용자가 로그인할 때 새 암호를 만들도록 요구하려면 다음 로그인 시 사용자가 새 암호를 생성해야 함을 선택합니다.

> **Important**
>
> 다음 로그인 시 사용자가 새 암호를 생성해야 함 옵션을 선택할 경우, 사용자에게 자신의 암호를 변경할 권한이 있는지 확인하세요. 자세한 내용은 IAM 사용자에게 자신의 암호 변경 허용 단원을 참조하십시오.

8. 사용자의 활성 콘솔 세션을 취소하려면 활성 콘솔 세션 취소를 선택합니다. 그런 다음, 적용을 선택합니다.

   사용자에 대한 활성 콘솔 세션을 취소하면 IAM은 모든 작업에 대한 모든 권한을 거부하는 새 인라인 정책을 해당 사용자에게 연결합니다. 여기에는 권한을 취소한 시점 이전에 생성된 세션과 약 30초 후의 세션에만 제한을 적용하는 조건이 포함됩니다. 사용자가 권한을 취소한 이후에 새로운 세션을 생성한 경우 이 사용자에게는 거부 정책이 적용되지 않습니다. 사용자가 이 방법을 사용하여 자신의 활성 콘솔 세션을 취소하면 즉시 AWS Management Console에서 로그아웃됩니다.

   > **Important**
   >
   > 사용자에 대한 활성 콘솔 세션을 취소하려면 해당 사용자에 대한 `PutUserPolicy` 권한이 있어야 합니다. 이렇게 하면 해당 사용자에게 `AWSRevokeOlderSessions` 인라인 정책을 연결할 수 있게 됩니다.

9. 암호를 확인하고 사용자와 공유하려면 콘솔 암호 대화 상자에서 표시를 선택합니다.

   > **Important**
   >
   > 이 단계를 완료한 후에는 보안상의 이유로 암호에 액세스할 수 없지만, 언제든지 새 암호를 만들 수 있습니다.

IAM 사용자 암호를 삭제(비활성화)하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 암호를 삭제할 사용자의 이름을 선택합니다.
4. 보안 자격 증명 탭을 선택한 다음 콘솔 로그인에서 콘솔 액세스 관리를 선택합니다.

5. 콘솔 액세스 관리에서 콘솔 액세스 비활성화를 선택합니다(선택되어 있지 않은 경우). 콘솔 액세스가 비활성화되어 있는 경우에는 암호가 필요 없습니다.
6. 사용자의 활성 콘솔 세션을 취소하려면 활성 콘솔 세션 취소를 선택합니다. 그런 다음 액세스 비활성화를 선택합니다.

> **Important**
>
> 사용자에 대한 활성 콘솔 세션을 취소하려면 해당 사용자에 대한 `PutUserPolicy` 권한이 있어야 합니다. 이렇게 하면 해당 사용자에게 `AWSRevokeOlderSessions` 인라인 정책을 연결할 수 있게 됩니다.

사용자에 대한 활성 콘솔 세션을 취소하면 IAM은 모든 작업에 대한 모든 권한을 거부하는 새 인라인 정책을 해당 IAM 사용자에게 삽입합니다. 여기에는 권한을 취소한 시점 이전에 생성된 세션과 약 30초 후의 세션에만 제한을 적용하는 조건이 포함됩니다. 사용자가 권한을 취소한 이후에 새로운 세션을 생성한 경우 이 사용자에게는 거부 정책이 적용되지 않습니다. 사용자가 이 방법을 사용하여 자신의 활성 콘솔 세션을 취소하면 즉시 AWS Management Console에서 로그아웃됩니다.

> **Important**
>
> 사용자의 암호를 제거하여 IAM 사용자가 AWS Management Console에 액세스하지 못하도록 할 수 있습니다. 그러면 사용자는 자신의 로그인 보안 인증을 사용해 AWS Management Console에 로그인할 수 없습니다. 그렇더라도 사용자의 권한이 변경되거나 위임된 역할을 사용하여 콘솔에 액세스하지 못하게 되지는 않습니다. 사용자에게 활성 액세스 키가 있다면 해당 키는 계속 작동하고 AWS CLI, Tools for Windows PowerShell 또는 AWS API 또는 AWS Console Mobile Application을 통해 액세스를 허용합니다.

IAM 사용자 암호 생성, 변경 또는 삭제(AWS CLI)

AWS CLI API를 이용해 IAM 사용자의 암호를 관리할 수 있습니다.

암호를 생성하려면(AWS CLI)

1. (선택 사항) 사용자에게 암호가 있는지 확인하려면 aws iam get-login-profile 명령을 실행합니다.
2. 암호를 생성하려면 aws iam create-login-profile 명령을 실행합니다.

사용자의 암호를 변경하려면(AWS CLI)

1. (선택 사항) 사용자에게 암호가 있는지 확인하려면 aws iam get-login-profile 명령을 실행합니다.
2. 암호를 변경하려면 aws iam update-login-profile 명령을 실행합니다.

사용자의 암호를 삭제(비활성화)하려면(AWS CLI)

1. (선택 사항) 사용자에게 암호가 있는지 확인하려면 aws iam get-login-profile 명령을 실행합니다.
2. (선택 사항) 사용자의 암호가 마지막으로 사용된 시간을 확인하려면 aws iam get-user 명령을 실행합니다.
3. 암호를 삭제하려면 aws iam delete-login-profile 명령을 실행합니다.

> **Important**
>
> 사용자의 암호를 삭제하면 해당 사용자가 더 이상 AWS Management Console에 로그인할 수 없습니다. 사용자에게 활성 액세스 키가 있다면 해당 키는 계속 작동하고 AWS CLI, Tools for Windows PowerShell 또는 AWS API 함수 호출을 통해 액세스를 허용합니다. AWS CLI, Tools for Windows PowerShell 또는 AWS API를 사용하여 AWS 계정에서 사용자를 삭제하는 경우 먼저 이 작업을 사용하여 암호를 삭제해야 합니다. 자세한 내용은 IAM 사용자 삭제(AWS CLI) 단원을 참조하십시오.

지정된 시간 이전에 사용자의 활성 콘솔 세션을 취소하려면(AWS CLI)

1. 지정된 시간 전에 IAM 사용자의 활성 콘솔 세션을 취소하는 인라인 정책을 내장하려면 다음 인라인 정책을 사용하고 aws iam put-user-policy 명령을 실행합니다.

   이 인라인 정책은 모든 권한을 거부하며 `aws:TokenIssueTime` 조건 키를 포함합니다. 인라인 정책의 `Condition` 요소에 지정된 시간이 지나기 전에 사용자의 활성 콘솔 세션을 취소합니다. `aws:TokenIssueTime` 조건 키 값을 사용자의 고유한 값으로 교체합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
```

```
      "DateLessThan": {
        "aws:TokenIssueTime": "2014-05-07T23:47:00Z"
      }
    }
  }
}
```

2. (선택 사항) IAM 사용자에게 포함된 인라인 정책의 이름을 나열하려면 aws iam list-user-policies 명령을 실행하세요.
3. (선택 사항) IAM 사용자에 포함된 명명된 인라인 정책을 보려면 aws iam get-user-policy 명령을 실행합니다.

IAM 사용자 암호 생성, 변경 또는 삭제(AWS API)

AWS API를 이용해 IAM 사용자의 암호를 관리할 수 있습니다.

암호를 생성하려면(AWS API)

1. (선택 사항) 사용자에게 암호가 있는지 확인하려면 GetLoginProfile 연산을 호출합니다.
2. 암호를 생성하려면 CreateLoginProfile 연산을 호출합니다.

사용자의 암호를 변경하려면(AWS API)

1. (선택 사항) 사용자에게 암호가 있는지 확인하려면 GetLoginProfile 연산을 호출합니다.
2. 암호를 변경하려면 UpdateLoginProfile 연산을 호출합니다.

사용자의 암호를 삭제(비활성화)하려면(AWS API)

1. (선택 사항) 사용자에게 암호가 있는지 확인하려면 GetLoginProfile 명령을 실행합니다.
2. (선택 사항) 사용자의 암호가 마지막으로 사용된 시간을 확인하려면 GetUser 명령을 실행합니다.
3. 암호를 삭제하려면 DeleteLoginProfile 명령을 실행합니다.

> **Important**
>
> 사용자의 암호를 삭제하면 해당 사용자가 더 이상 AWS Management Console에 로그인할 수 없습니다. 사용자에게 활성 액세스 키가 있다면 해당 키는 계속 작동하고 AWS CLI, Tools for Windows PowerShell 또는 AWS API 함수 호출을 통해 액세스를 허용합니다. AWS CLI, Tools

for Windows PowerShell 또는 AWS API를 사용하여 AWS 계정에서 사용자를 삭제하는 경우 먼저 이 작업을 사용하여 암호를 삭제해야 합니다. 자세한 내용은 IAM 사용자 삭제(AWS CLI) 단원을 참조하십시오.

지정된 시간 이전에 사용자의 활성 콘솔 세션을 취소하려면(AWS API)

1. 지정된 시간 전에 IAM 사용자의 활성 콘솔 세션을 취소하는 인라인 정책을 내장하려면 다음 인라인 정책을 사용하고 PutUserPolicy 명령을 실행하세요.

   이 인라인 정책은 모든 권한을 거부하며 `aws:TokenIssueTime` 조건 키를 포함합니다. 인라인 정책의 `Condition` 요소에 지정된 시간이 지나기 전에 사용자의 활성 콘솔 세션을 취소합니다. `aws:TokenIssueTime` 조건 키 값을 사용자의 고유한 값으로 교체합니다.

   ```
   {
     "Version": "2012-10-17",
     "Statement": {
       "Effect": "Deny",
       "Action": "*",
       "Resource": "*",
       "Condition": {
         "DateLessThan": {
           "aws:TokenIssueTime": "2014-05-07T23:47:00Z"
         }
       }
     }
   }
   ```

2. (선택 사항) IAM 사용자에 포함된 인라인 정책의 이름을 나열하려면 ListUserPolicies 명령을 실행합니다.
3. (선택 사항) IAM 사용자에 포함된 명명된 인라인 정책을 보려면 GetUserPolicy 명령을 실행합니다.

IAM 사용자에게 자신의 암호 변경 허용

> **Note**
>
> 페더레이션형 ID를 가진 사용자는 ID 제공업체가 정의한 프로세스를 사용하여 암호를 변경합니다. 가장 좋은 방법은 인간 사용자가 ID 제공업체와의 페더레이션을 사용하여 임시 보안 인증으로 AWS에 액세스하도록 하는 것입니다.

IAM 사용자에게 AWS Management Console에 로그인할 때 사용하는 자신의 암호를 변경할 권한을 부여할 수 있습니다. 이 작업을 두 가지 방법으로 수행할 수 있습니다.

- 계정의 모든 IAM 사용자에게 자신의 암호 변경을 허용합니다.
- 선택된 IAM 사용자에게만 자신의 암호 변경을 허용합니다. 이 시나리오에서는 모든 사용자의 암호 변경 옵션을 비활성화한 후 IAM 정책을 사용하여 일부 사용자에게만 자신의 암호를 변경할 수 있는 권한을 부여합니다. 이 방법을 사용하면 사용자가 자신의 암호를 변경할 수 있으며 필요에 따라 액세스 키와 같은 다른 자격 증명을 변경할 수 있습니다.

> **Important**
>
> IAM 사용자에게 강력한 암호를 생성할 것을 요구하는 사용자 지정 암호 정책을 설정하는 것이 좋습니다.

모든 IAM 사용자에게 자신의 암호 변경을 허용하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 [계정 설정(Account settings)]을 클릭합니다.
3. Password policy(암호 정책) 섹션에서 Edit(편집)를 선택합니다.
4. Custom(사용자 지정)을 선택하여 사용자 지정 암호 정책을 사용합니다.
5. 사용자 자신의 암호 변경 허용(Allow users to change their own password)을 선택한 다음 변경 사항 저장(Save changes)을 선택합니다. 이렇게 하면 계정의 모든 사용자에게 해당 사용자의 `iam:ChangePassword` 작업과 `iam:GetAccountPasswordPolicy` 작업에 대한 액세스 권한만 허용됩니다.

6. 사용자에게 암호 변경에 대한 다음 지침을 제공합니다. IAM 사용자가 자신의 암호를 변경하는 방법.

계정의 암호 정책(모든 사용자가 직접 암호를 변경하게 하는 정책 포함) 변경에 사용할 수 있는 AWS CLI, Tools for Windows PowerShell 및 API 명령에 대한 자세한 내용은 암호 정책 설정(AWS CLI) 섹션을 참조하세요.

선택된 IAM 사용자에게 자신의 암호 변경을 허용하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 [계정 설정(Account settings)]을 클릭합니다.
3. [암호 정책(Password policy)] 섹션에서 [사용자 자신의 암호 변경 허용(Allow users to change their own password)] 확인란이 선택 취소되어 있는지 확인합니다. 이 확인란을 선택하면 모든 사용자가 직접 암호를 변경할 수 있게 됩니다. (위 절차 참조).
4. 암호 변경을 허용할 사용자가 아직 없다면 사용자를 만듭니다. 세부 정보는 AWS 계정에서 IAM 사용자 생성을 참조하세요.
5. (선택 사항) 직접 암호를 변경하게 할 IAM 사용자 그룹을 만든 다음 앞 단계에서 만든 사용자를 그룹에 추가합니다. 세부 정보는 IAM 사용자 그룹을 참조하세요.
6. 그룹에 다음 정책을 할당합니다. 자세한 내용은 IAM 정책 관리 단원을 참조하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:GetAccountPasswordPolicy",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:ChangePassword",
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    }
  ]
}
```

이 정책은 암호 변경 작업에 대한 액세스 권한을 부여하여 사용자가 콘솔, AWS CLI, Tools for Windows PowerShell 또는 API로부터 본인의 암호만을 변경할 수 있게 합니다. 또한 사용자가 현재 암호 정책을 볼 수 있도록 GetAccountPasswordPolicy 작업에 대한 액세스 권한도 부여합니다. 이 권한은 사용자가 [암호 변경(Change password)] 페이지에서 계정 암호를 보는 데 필요합니다. 사용자가 현재 암호 정책을 읽고 변경된 암호가 정책의 요건을 충족하는지 확인할 수 있도록 허용해야 합니다.

7. 사용자에게 암호 변경에 대한 다음 지침을 제공합니다. IAM 사용자가 자신의 암호를 변경하는 방법.

자세한 정보

자격 증명 관리에 대한 자세한 내용은 다음 주제를 참조하세요.

- IAM 사용자에게 자신의 암호 변경 허용
- AWS에서 사용자 암호 관리
- IAM 사용자의 계정 암호 정책 설정
- IAM 정책 관리
- IAM 사용자가 자신의 암호를 변경하는 방법

IAM 사용자가 자신의 암호를 변경하는 방법

자신의 IAM 사용자 암호를 변경할 수 있는 권한이 부여된 경우 AWS Management Console의 특별 페이지를 사용하여 이 작업을 수행할 수 있습니다. AWS CLI 또는 AWS API도 사용할 수 있습니다.

주제

- 필요한 권한
- IAM 사용자가 자신의 암호를 변경하는 방법(콘솔)
- IAM 사용자가 자신의 암호를 변경하는 방법(AWS CLI 또는 AWS API)

필요한 권한

자신의 IAM 사용자에 대한 암호를 변경하려면 AWS: IAM 사용자가 보안 인증 페이지에서 자신의 콘솔 암호를 변경할 수 있도록 허용 정책에 따른 권한이 있어야 합니다.

IAM 사용자가 자신의 암호를 변경하는 방법(콘솔)

다음 절차는 IAM 사용자가 AWS Management Console을 사용하여 자신의 암호를 변경하는 방법을 설명합니다.

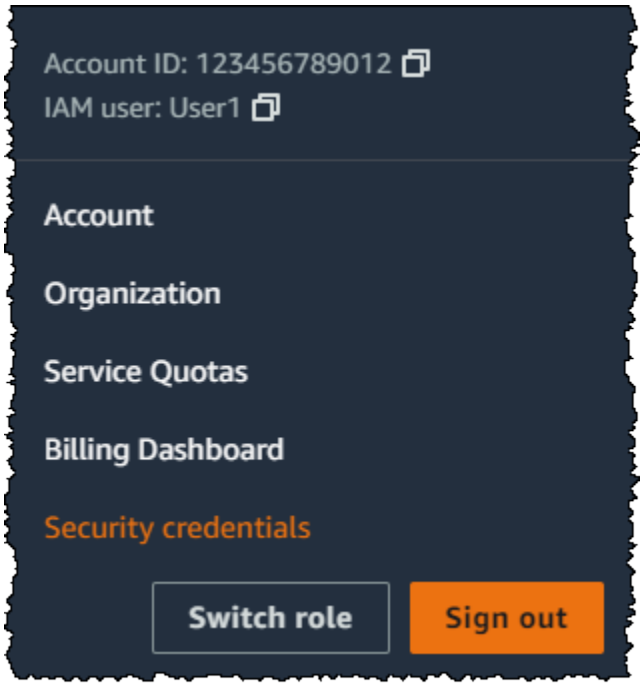
자신의 IAM 사용자 암호를 변경하려면(콘솔)

1. AWS 계정 ID나 계정 별칭, IAM 사용자 이름 및 암호를 사용하여 IAM 콘솔에 로그인합니다.

   > **Note**
   >
   > 사용자 편의를 위해 AWS 로그인 페이지는 브라우저 쿠키를 사용하여 IAM 사용자 이름 및 계정 정보를 기억합니다. 이전에 다른 사용자로 로그인한 경우 페이지 하단 근처의 다른 계정에 로그인(Sign in to a different account)을 선택하여 기본 로그인 페이지로 돌아갑니다. 여기서 AWS 계정 ID 또는 계정 별칭을 입력하면 계정의 IAM 사용자 로그인 페이지로 리디렉션됩니다.

   AWS 계정 ID를 받으려면 관리자에게 문의하세요.

2. 오른쪽 상단의 탐색 모음에서 사용자 이름을 선택한 다음 Security credentials(보안 자격 증명)를 선택합니다.



3. AWS IAM 자격 증명 탭에서 암호 업데이트를 선택합니다.

4. Current password(현재 암호)에 현재 암호를 입력합니다. New password(새 암호) 및 Confirm new password(새 암호 확인)에 새 암호를 입력합니다. 그런 다음 암호 업데이트를 선택합니다.

> **Note**
>
> 새 암호는 계정 암호 정책의 요건을 따라야 합니다. 자세한 내용은 IAM 사용자의 계정 암호 정책 설정 단원을 참조하십시오.

IAM 사용자가 자신의 암호를 변경하는 방법(AWS CLI 또는 AWS API)

다음 절차는 IAM 사용자가 AWS CLI 또는 AWS API를 사용하여 자신의 암호를 변경하는 방법을 설명합니다.

자신의 IAM 암호를 변경하려면 다음을 사용하세요.

- AWS CLI: `aws iam change-password`
- AWS API: `ChangePassword`

IAM 사용자의 액세스 키 관리

Follow us on Twitter

> **Important**
>
> 모범 사례는 액세스 키와 같은 장기 보안 인증을 생성하는 대신 임시 보안 인증(예: IAM 역할)을 사용하는 것입니다. 액세스 키를 생성하기 전에 장기 액세스 키의 대안을 검토합니다.

액세스 키는 IAM 사용자 또는 AWS 계정 루트 사용자에 대한 장기 보안 인증입니다. 액세스 키를 사용하여 AWS CLI 또는 AWS API에 대한 프로그래밍 요청에 서명할 수 있습니다(직접 또는 AWS SDK를 사용하여). 자세한 내용은 API 요청용 AWS Signature Version 4 단원을 참조하십시오.

액세스 키는 액세스 키 ID(예: `AKIAIOSFODNN7EXAMPLE`)와 비밀 액세스 키(예: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`)의 두 가지 부분으로 구성됩니다. 액세스 키 ID와 보안 액세스 키를 함께 사용하여 요청을 인증해야 합니다.

액세스 키 페어를 생성할 때는 액세스 키 ID와 보안 액세스 키를 안전한 위치에 저장합니다. 보안 액세스 키는 생성할 때만 사용할 수 있습니다. 보안 액세스 키를 분실한 경우 액세스 키를 삭제하고 새 키를

생성해야 합니다. 자세한 내용은 분실하거나 잊어버린 AWS 암호 또는 액세스 키 재설정를 참조하세요.

사용자당 최대 2개의 액세스 키를 가질 수 있습니다.

> **Important**
>
> 액세스 키를 안전하게 관리하세요. 계정 식별자를 찾는 데 도움이 되더라도 액세스 키를 권한 없는 당사자에게 제공하지 마세요. 이로 인해 다른 사람에게 계정에 대한 영구 액세스를 제공하게 될 수 있습니다.

다음 주제에서는 액세스 키와 관련된 관리 작업에 대한 자세한 정보를 제공합니다.

주제

- 액세스 키를 관리하는 데 필요한 권한
- 액세스 키 관리(콘솔)
- 액세스 키 관리(AWS CLI)
- 액세스 키 관리(AWS API)
- 액세스 키 업데이트
- 액세스 키 보안
- Amazon Keyspaces(Apache Cassandra용)와 함께 IAM 사용

액세스 키를 관리하는 데 필요한 권한

> **Note**
>
> `iam:TagUser`은 액세스 키에 설명을 추가하고 편집할 수 있는 선택적 권한입니다. 자세한 내용은 IAM 사용자 태깅 단원을 참조하세요.

자신의 IAM 사용자에 대한 액세스 키를 생성하려면 다음 정책에 따른 권한이 있어야 합니다.

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

```
            "Sid": "CreateOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:CreateAccessKey",
                "iam:GetUser",
                "iam:ListAccessKeys",
                "iam:TagUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

자신의 IAM 사용자에 대한 액세스 키를 업데이트하려면 다음 정책에 따른 권한이 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ManageOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:CreateAccessKey",
                "iam:DeleteAccessKey",
                "iam:GetAccessKeyLastUsed",
                "iam:GetUser",
                "iam:ListAccessKeys",
                "iam:UpdateAccessKey",
                "iam:TagUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

액세스 키 관리(콘솔)

AWS Management Console을 사용하여 IAM 사용자의 액세스 키를 관리할 수 있습니다.

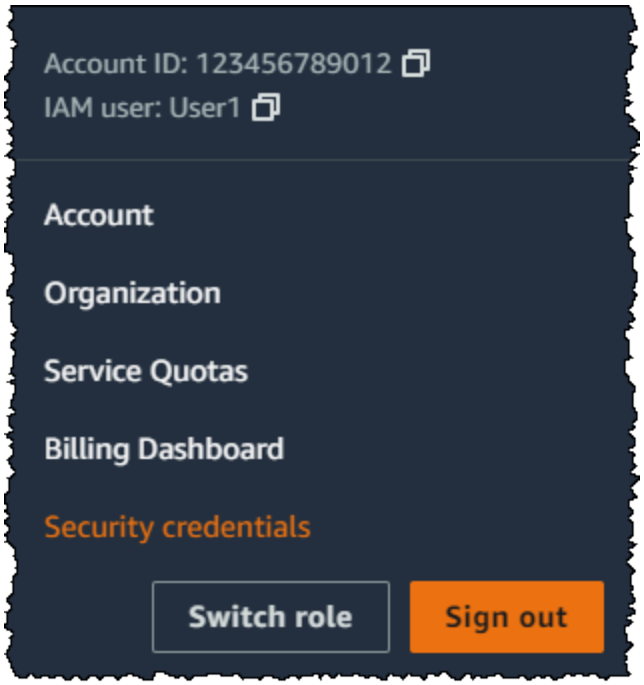
자신의 액세스 키를 생성, 수정 또는 삭제하려면(콘솔)

1. AWS 계정 ID나 계정 별칭, IAM 사용자 이름 및 암호를 사용하여 IAM 콘솔에 로그인합니다.

> **Note**
>
> 사용자 편의를 위해 AWS 로그인 페이지는 브라우저 쿠키를 사용하여 IAM 사용자 이름 및 계정 정보를 기억합니다. 이전에 다른 사용자로 로그인한 경우 페이지 하단 근처의 다른 계정에 로그인(Sign in to a different account)을 선택하여 기본 로그인 페이지로 돌아갑니다. 여기서 AWS 계정 ID 또는 계정 별칭을 입력하면 계정의 IAM 사용자 로그인 페이지로 리디렉션됩니다.

AWS 계정 ID를 받으려면 관리자에게 문의하세요.

2. 오른쪽 상단의 탐색 모음에서 사용자 이름을 선택한 다음 Security credentials(보안 자격 증명)를 선택합니다.



다음 중 하나를 수행합니다.

액세스 키 생성

1. 액세스 키 섹션에서 액세스 키 생성을 선택합니다. 이미 두 개의 액세스 키가 있는 경우, 이 버튼은 비활성화되어 있으며 액세스 키를 삭제해야 새로 생성할 수 있습니다.

2. Access key best practices & alternatives(액세스 키 모범 사례 및 대안) 페이지에서 사용 사례를 선택하여 장기 액세스 키 생성 방지에 도움이 되는 추가 옵션에 대해 알아봅니다. 사용 사례에 여전히 액세스 키가 필요하다고 판단되면 Other(기타), Next(다음)를 차례로 선택합니다.
3. (선택 사항) 액세스 키에 대한 설명 태그 값을 설정합니다. 이렇게 하면 IAM 사용자에게 태그 키-값 페어가 추가됩니다. 이는 나중에 액세스 키를 식별하고 업데이트하는 데 도움이 됩니다. 태그 키는 액세스 키 ID로 설정됩니다. 태그 값은 사용자가 지정하는 액세스 키 설명으로 설정됩니다. 모두 마쳤으면 Create access key(액세스 키 생성)를 선택합니다.
4. Retrieve access keys(액세스 키 검색) 페이지에서 Show(표시)를 선택하여 사용자의 비밀 액세스 키 값을 표시하거나 Download .csv file(.csv 파일 다운로드)을 선택합니다. 이것이 비밀 액세스 키를 저장할 수 있는 유일한 기회입니다. 비밀 액세스 키를 안전한 위치에 저장한 후 Done(완료)을 선택합니다.

액세스 키를 비활성화하려면 다음을 수행하세요.

- Access keys(액세스 키) 섹션에서 비활성화하려는 키를 찾은 다음 Actions(작업), Deactivate(비활성화)를 차례로 선택합니다. 확인 메시지가 나타나면 Deactivate(비활성화)를 클릭합니다. 비활성화된 액세스 키는 여전히 두 개의 액세스 키 제한에 포함됩니다.

액세스 키를 생성하려면 다음을 수행하세요.

- Access keys(액세스 키) 섹션에서 활성화하려는 키를 찾은 다음 Actions(작업), Activate(활성화)를 차례로 선택합니다.

더 이상 필요하지 않은 액세스 키를 삭제하려면 다음을 수행하세요.

- Access keys(액세스 키) 섹션에서 삭제하려는 키를 찾은 다음 Actions(작업), Delete(삭제)를 차례로 선택합니다. 대화 상자의 지침에 따라 먼저 Deactivate(비활성화)를 수행한 다음 삭제를 확인합니다. 액세스 키를 영구적으로 삭제하기 전에 액세스 키가 더 이상 사용되고 있지 않은지 확인하는 것이 좋습니다.

다른 IAM 사용자의 액세스 키를 생성, 수정 또는 삭제하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 액세스 키를 관리하려는 사용자 이름을 선택한 다음 보안 자격 증명 탭을 선택합니다.

4. 액세스 키 섹션에서 다음 작업을 수행합니다.

   - 액세스 키를 생성하려면 Create access key(액세스 키 생성)을 선택합니다. 버튼이 비활성화된 경우 기존 키 중 하나를 삭제해야 새 키를 생성할 수 있습니다. Access key best practices & alternatives(액세스 키 모범 사례 및 대안) 페이지에서 모범 사례와 대안을 검토합니다. 사용 사례를 선택하여 장기 액세스 키 생성 방지에 도움이 되는 추가 옵션에 대해 알아봅니다. 사용 사례에 여전히 액세스 키가 필요하다고 판단되면 Other(기타), Next(다음)를 차례로 선택합니다. Retrieve access key page(액세스 키 검색) 페이지에서 Show(표시)를 선택하여 사용자의 비밀 액세스 키 값을 표시합니다. 액세스 키 ID 및 보안 액세스 키를 컴퓨터의 안전한 위치에 `.csv` 파일로 저장하려면 Download .csv file(.csv 파일 다운로드) 버튼을 선택합니다. 사용자를 위한 액세스 키를 생성하면 해당 키 페어가 기본적으로 활성화되며 사용자는 이 페어를 즉시 사용할 수 있습니다.
   - 활성 액세스 키를 비활성화하려면 Actions(작업), Deactivate(비활성화)를 차례로 선택합니다.
   - 비활성 액세스 키를 활성화하려면 Actions(작업), Activate(활성화)를 차례로 선택합니다.
   - 액세스 키를 삭제하려면 Actions(작업), Delete(삭제)를 차례로 선택합니다. 대화 상자의 지침에 따라 먼저 Deactivate(비활성화)를 수행한 다음 삭제를 확인합니다. AWS에서는 이 작업을 수행하기 전에 먼저 키를 비활성화하고 키가 더 이상 사용되지 않는지 테스트할 것을 권장합니다. AWS Management Console을 사용하는 경우 키를 삭제하기 전에 키를 비활성화해야 합니다.

IAM 사용자에 대한 액세스 키를 나열하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 해당 사용자의 이름을 선택한 후 보안 자격 증명 탭을 선택합니다. Access keys(액세스 키) 섹션에는 사용자의 액세스 키와 각 키의 상태가 표시됩니다.

> **Note**
>
> 사용자의 액세스 키 ID만 표시됩니다. 비밀 액세스 키는 키를 만들 때만 가져올 수 있습니다.

여러 IAM 사용자의 액세스 키 ID를 나열하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 필요할 경우 다음 단계를 통해 사용자 테이블에 액세스 키 ID 열을 추가합니다.
   a. 테이블 위 맨 오른쪽에서 설정 아이콘 ( ) 을 선택합니다.
   b. Manage columns(열 관리)에서 액세스 키 ID를 선택합니다.
   c. 닫기를 선택하여 사용자 목록으로 돌아갑니다.
4. 액세스 키 ID 열에는 각 액세스 키 ID가 표시되고 그 다음에 키의 상태가 표시됩니다. 예: 234782070278420732307623740 23 (Active) 또는 220937402396702370248434203 27 (Inactive).

   이 정보를 사용하여 한 개 또는 두 개의 액세스 키를 가진 사용자의 액세스 키를 보고 복사할 수 있습니다. 액세스 키가 없는 사용자는 이 열에 없음이라고 표시됩니다.

> **Note**
> 사용자의 액세스 키 ID와 상태만 표시됩니다. 비밀 액세스 키는 키를 만들 때만 가져올 수 있습니다.

어떤 IAM 사용자가 특정 액세스 키를 소유하고 있는지 확인하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 검색 상자에 해당 사용자의 액세스 키 ID를 입력하거나 붙여 넣습니다.
4. 필요할 경우 다음 단계를 통해 사용자 테이블에 액세스 키 ID 열을 추가합니다.
   a. 테이블 위 맨 오른쪽에서 설정 아이콘 ( ) 을 선택합니다.

b. Manage columns(열 관리)에서 액세스 키 ID를 선택합니다.

c. 닫기를 선택하여 사용자 목록으로 돌아간 후 지정된 액세스 키를 소유하는 사용자로 필터링 되었는지 확인합니다.

액세스 키 관리(AWS CLI)

AWS CLI에서 IAM 사용자 액세스 키를 관리하려면 다음 명령을 실행합니다.

- 액세스 키 생성: `aws iam create-access-key`
- 액세스 키를 비활성화하거나 활성화하려면: `aws iam update-access-key`
- 사용자의 액세스 키를 나열하려면: `aws iam list-access-keys`
- 가장 최근에 액세스 키를 사용한 시기 확인: `aws iam get-access-key-last-used`
- 액세스 키 삭제: `aws iam delete-access-key`

액세스 키 관리(AWS API)

AWS API에서 IAM 사용자의 액세스 키를 관리하려면 다음 작업을 호출합니다.

- 액세스 키 생성: `CreateAccessKey`
- 액세스 키를 비활성화하거나 활성화하려면: `UpdateAccessKey`
- 사용자의 액세스 키를 나열하려면: `ListAccessKeys`
- 가장 최근에 액세스 키를 사용한 시기 확인: `GetAccessKeyLastUsed`
- 액세스 키 삭제: `DeleteAccessKey`

액세스 키 업데이트

보안 모범 사례로, 필요할 때(예: 직원 퇴사) IAM 사용자 액세스 키를 업데이트하는 것이 좋습니다. 필요한 권한이 부여되면 IAM 사용자는 자신의 액세스 키를 업데이트할 수 있습니다.

IAM 사용자에게 자신의 액세스 키를 업데이트할 수 있는 권한을 부여하는 방법에 대한 자세한 내용은 AWS: IAM 사용자가 보안 인증 페이지에서 자신의 암호, 액세스 키 및 SSH 퍼블릭 키를 관리할 수 있도록 허용 섹션을 참조하세요. 모든 IAM 사용자가 주기적으로 암호를 업데이트하도록 요구하는 암호 정책(빈도 포함)을 계정에 적용할 수도 있습니다. 자세한 내용은 IAM 사용자의 계정 암호 정책 설정 단원을 참조하십시오.

주제

- IAM 사용자 액세스 키 업데이트(콘솔)
- 액세스 키 업데이트(AWS CLI)
- 액세스 키 업데이트(AWS API)

IAM 사용자 액세스 키 업데이트(콘솔)

AWS Management Console에서 액세스 키를 업데이트할 수 있습니다.

애플리케이션을 중단하지 않고 IAM 사용자의 액세스 키를 업데이트하려면(콘솔)

1. 최초 액세스 키가 활성 상태일 때 두 번째 액세스 키를 만듭니다.

   a. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

   b. 탐색 창에서 사용자를 선택합니다.

   c. 해당 사용자의 이름을 선택한 후 보안 자격 증명 탭을 선택합니다.

   d. 액세스 키 섹션에서 액세스 키 생성을 선택합니다. Access key best practices & alternatives(액세스 키 모범 사례 및 대안) 페이지에서 Other(기타), Next(다음)를 차례로 선택합니다.

   e. (선택 사항) 액세스 키에 대한 설명 태그 값을 설정하여 이 IAM 사용자에게 태그 키-값 페어를 추가합니다. 이는 나중에 액세스 키를 식별하고 업데이트하는 데 도움이 됩니다. 태그 키는 액세스 키 ID로 설정됩니다. 태그 값은 사용자가 지정하는 액세스 키 설명으로 설정됩니다. 모두 마쳤으면 Create access key(액세스 키 생성)를 선택합니다.

   f. Retrieve access keys(액세스 키 검색) 페이지에서 Show(표시)를 선택하여 사용자의 비밀 액세스 키 값을 표시하거나 Download .csv file(.csv 파일 다운로드)을 선택합니다. 이것이 비밀 액세스 키를 저장할 수 있는 유일한 기회입니다. 비밀 액세스 키를 안전한 위치에 저장한 후 Done(완료)을 선택합니다.

      사용자를 위한 액세스 키를 생성하면 해당 키 페어가 기본적으로 활성화되며 사용자는 이 페어를 즉시 사용할 수 있습니다. 따라서 사용자에게 두 개의 활성 액세스 키가 생깁니다.

2. 새 액세스 키를 사용하도록 모든 애플리케이션과 도구를 업데이트합니다.
3. 가장 오래된 액세스 키의 Last used(마지막 사용) 정보를 검토하여 최초 액세스 키가 아직 사용 중인지 확인합니다. 한 가지 접근 방식은 며칠을 기다린 다음 계속하기 전에 사용된 적이 있는 기존 액세스 키가 있는지 확인하는 것입니다.

4. Last used(마지막 사용) 정보에 오래된 키가 사용된 적이 없다고 표시되더라도 최초 액세스 키를 바로 삭제하지 않는 것이 좋습니다. 대신 Actions(작업), Deactivate(비활성화)를 차례로 선택하여 첫 번째 액세스 키를 비활성화합니다.
5. 새 액세스 키만 사용하여 애플리케이션이 작동 중인지 확인합니다. 원래 액세스 키를 계속 사용하는 어떤 애플리케이션도 AWS 리소스에 더 이상 액세스할 수 없기 때문에 이 시점에 작업을 중단합니다. 그러한 애플리케이션 또는 도구를 찾는다면 최초 액세스 키를 다시 활성화할 수 있습니다. 그런 다음 Step 3 단원으로 돌아가 이 애플리케이션을 업데이트하여 새 키를 사용하세요.
6. 일정 기간 기다린 후 모든 애플리케이션과 도구가 업데이트되었는지 확인한 뒤에 최초 액세스 키를 삭제할 수 있습니다:
   a. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
   b. 탐색 창에서 사용자를 선택합니다.
   c. 해당 사용자의 이름을 선택한 후 보안 자격 증명 탭을 선택합니다.
   d. Access keys(액세스 키) 섹션에서 삭제하려는 액세스 키에 대해 Actions(작업), Delete(삭제)를 차례로 선택합니다. 대화 상자의 지침에 따라 먼저 Deactivate(비활성화)를 수행한 다음 삭제를 확인합니다.

업데이트하거나 삭제해야 하는 액세스 키를 결정하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 필요할 경우 다음 단계를 통해 사용자 테이블에 Access key age(액세스 키 수명) 열을 추가합니다.
   a. 테이블 위 맨 오른쪽에서 설정 아이콘
   ()
   을 선택합니다.
   b. Manage Columns(열 관리)에서 Access key age(액세스 키 수명)을 선택합니다.
   c. 닫기를 선택하여 사용자 목록으로 돌아갑니다.
4. Access key age(액세스 키 수명) 열에는 가장 오래된 활성 액세스 키가 생성된 이후로 경과한 일수가 표시됩니다. 이 정보를 사용하여 업데이트하거나 삭제해야 하는 액세스 키를 소유한 사용자를 찾을 수 있습니다. 액세스 키가 없는 사용자는 이 열에 없음이라고 표시됩니다.

액세스 키 업데이트(AWS CLI)

AWS Command Line Interface에서 액세스 키를 업데이트할 수 있습니다.

애플리케이션을 중단하지 않고 액세스 키를 업데이트하려면(AWS CLI)

1. 최초 액세스 키가 활성 상태일 때 두 번째 액세스 키를 만들면 이 키도 기본적으로 활성 상태가 됩니다. 다음 명령 실행:

   - `aws iam create-access-key`

   따라서 사용자에게 두 개의 활성 액세스 키가 생깁니다.
2. 새 액세스 키를 사용하도록 모든 애플리케이션과 도구를 업데이트합니다.
3. 다음 명령을 사용하여 최초 액세스 키가 아직 사용 중인지 확인합니다.

   - `aws iam get-access-key-last-used`

   한 가지 접근 방식은 며칠을 기다린 다음 계속하기 전에 사용된 적이 있는 기존 액세스 키가 있는지 확인하는 것입니다.
4. Step 3 단계를 통해 기존 키를 사용한 적이 없다는 것이 밝혀진 경우 최초의 액세스 키를 즉시 삭제하지 말 것을 권장합니다. 그 대신 다음 명령을 사용하여 최초 액세스 키의 상태를 `Inactive`로 변경하세요.

   - `aws iam update-access-key`
5. 새 액세스 키만 사용하여 애플리케이션이 작동 중인지 확인합니다. 원래 액세스 키를 계속 사용하는 어떤 애플리케이션도 AWS 리소스에 더 이상 액세스할 수 없기 때문에 이 시점에 작업을 중단합니다. 그러한 애플리케이션 또는 도구를 찾는다면 그 상태를 `Active`로 되돌려 최초 액세스 키를 다시 활성화할 수 있습니다. 그런 다음 Step 2 단계로 돌아가 이 애플리케이션을 업데이트해 새 키를 사용하세요.
6. 일정 기간 기다린 후 모든 애플리케이션과 도구가 업데이트되었는지 확인한 뒤에 다음 명령을 사용하여 최초 액세스 키를 삭제할 수 있습니다.

   - `aws iam delete-access-key`

액세스 키 업데이트(AWS API)

AWS API를 사용하여 액세스 키를 업데이트할 수 있습니다.

애플리케이션을 중단하지 않고 액세스 키를 업데이트하려면(AWS API)

1. 최초 액세스 키가 활성 상태일 때 두 번째 액세스 키를 만들면 이 키도 기본적으로 활성 상태가 됩니다. 다음 작업을 호출합니다.

   - `CreateAccessKey`

   따라서 사용자에게 두 개의 활성 액세스 키가 생깁니다.
2. 새 액세스 키를 사용하도록 모든 애플리케이션과 도구를 업데이트합니다.
3. 다음 연산을 호출하여 최초 액세스 키가 아직 사용 중인지 확인합니다.

   - `GetAccessKeyLastUsed`

   한 가지 접근 방식은 며칠을 기다린 다음 계속하기 전에 사용된 적이 있는 기존 액세스 키가 있는지 확인하는 것입니다.
4. Step 3 단계를 통해 기존 키를 사용한 적이 없다는 것이 밝혀진 경우 최초의 액세스 키를 즉시 삭제하지 말 것을 권장합니다. 그 대신 다음 연산을 호출하여 최초 액세스 키의 상태를 `Inactive`로 변경하세요.

   - `UpdateAccessKey`
5. 새 액세스 키만 사용하여 애플리케이션이 작동 중인지 확인합니다. 원래 액세스 키를 계속 사용하는 어떤 애플리케이션도 AWS 리소스에 더 이상 액세스할 수 없기 때문에 이 시점에 작업을 중단합니다. 그러한 애플리케이션 또는 도구를 찾는다면 그 상태를 `Active`로 되돌려 최초 액세스 키를 다시 활성화할 수 있습니다. 그런 다음 Step 2 단계로 돌아가 이 애플리케이션을 업데이트해 새 키를 사용하세요.
6. 일정 기간 기다린 후 모든 애플리케이션과 도구가 업데이트되었는지 확인한 뒤에 다음 연산을 호출하여 최초 액세스 키를 삭제할 수 있습니다.

   - `DeleteAccessKey`

액세스 키 보안

액세스 키를 보유한 사람은 누구든지 AWS 리소스에 대해 사용자와 동일한 권한으로 액세스할 수 있습니다. 따라서 AWS에서는 사용자의 액세스 키를 보호하기 위해 최선을 다하며, 사용자도 Shared Responsibility Model에 부합하도록 노력해야 합니다.

액세스 키를 보호하는 데 도움이 되는 지침을 보려면 다음 섹션을 확장하세요.

> **Note**
>
> 사용자 조직의 보안 요구 사항과 정책은 이 주제에 설명된 것과 다를 수 있습니다. 여기에 제공된 제안 사항은 일반적인 지침입니다.

AWS 계정 루트 사용자 액세스 키 제거(또는 생성하지 않음)

계정을 보호하는 가장 좋은 방법 중 하나는 AWS 계정 루트 사용자에 대한 액세스 키를 보유하지 않는 것입니다. 매우 드물지만 루트 사용자 액세스 키가 필요한 경우를 제외하고는 액세스 키를 생성하지 않는 것이 좋습니다. 대신 일상적인 관리 작업을 위한 관리 사용자를 AWS IAM Identity Center에서 생성합니다. IAM Identity Center에서 관리 사용자를 생성하는 방법에 대한 자세한 내용은 IAM Identity Center 사용 설명서의 Getting started를 참조하세요.

계정에 대한 루트 사용자 액세스 키가 이미 있는 경우 해당 액세스 키(있는 경우)를 현재 사용 중인 애플리케이션의 위치를 확인하고 루트 사용자 액세스 키를 IAM 사용자 액세스 키로 교체하는 것이 좋습니다. 그런 다음 루트 사용자 액세스 키를 비활성화하고 제거합니다. 액세스 키 업데이트 방법에 대한 자세한 내용은 액세스 키 업데이트 섹션을 참조하세요.

장기 액세스 키 대신 임시 보안 인증(IAM 역할) 사용

대부분의 시나리오에서는 IAM 사용자와 같이 만료되지 않는 장기 액세스 키가 필요하지 않습니다. 대신, IAM 역할을 만들고 임시 보안 인증 정보를 생성할 수 있습니다. 임시 보안 자격 증명은 액세스 키 ID와 비밀 액세스 키로 구성되지만, 자격 증명이 만료되는 시간을 나타내는 보안 토큰을 포함합니다.

장기 액세스 키(IAM 사용자 및 루트 사용자에 연결된 액세스 키)는 수동으로 취소할 때까지 유효하게 유지됩니다. 하지만 IAM 역할과 AWS Security Token Service의 기타 기능을 통해 얻는 임시 보안 인증은 단기간 내에 만료됩니다. 임시 보안 자격 증명을 사용하면 자격 증명이 실수로 노출된 경우에 위험을 줄일 수 있습니다.

다음 시나리오에서는 IAM 역할과 임시 보안 인증을 사용합니다.

- Amazon EC2 인스턴스에서 실행 중인 애플리케이션 또는 AWS CLI 스크립트가 있는 경우. 애플리케이션에서 액세스 키를 직접 사용하지 않도록 합니다. 액세스 키를 애플리케이션에 전달하거나, 애플리케이션에 포함하거나, 애플리케이션에서 소스로부터 액세스 키를 읽지 않도록 합니다. 대신, 애플리케이션에 대한 적절한 권한을 가진 IAM 역할을 정의하고 EC2의 역할을 사용하여 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 시작합니다. 그러면 IAM 역할이 Amazon EC2 인스턴스에 연결됩니다. 이 방법을 통해 애플리케이션은 AWS에 대한 프로그래밍 방식의 직접

호출에 사용할 수 있는 임시 보안 인증을 얻을 수 있습니다. AWS SDK 및 AWS Command Line Interface(AWS CLI)에서는 역할에서 임시 보안 인증을 자동으로 가져올 수 있습니다.

- 교차 계정 액세스 권한을 부여해야 합니다. IAM 역할을 사용하여 계정 사이에 신뢰를 구축한 다음, 한 계정의 사용자에게 신뢰할 수 있는 계정에 액세스할 수 있는 제한된 권한을 부여합니다. 자세한 내용은 튜토리얼: IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임 단원을 참조하십시오.
- 모바일 앱을 사용합니다. 암호화된 스토리지를 포함하여 앱에 액세스 키를 포함하지 않도록 합니다. 대신, Amazon Cognito를 사용하여 앱에서 사용자 자격 증명을 관리합니다. 이 서비스를 사용하면 Login with Amazon, Facebook, Google 또는 OpenID Connect(OIDC) 호환 자격 증명 공급자를 통해 사용자를 인증할 수 있습니다. 그런 다음 Amazon Cognito 보안 인증 공급자를 사용하여 앱에서 AWS에 요청하는 데 사용하는 보안 인증을 관리할 수 있습니다.
- AWS에 연동하려고 하고 조직에서 SAML 2.0을 지원합니다. SAML 2.0을 지원하는 자격 증명 공급자가 있는 조직에서 작업하는 경우 SAML을 사용하도록 공급자를 구성합니다. SAML을 사용하여 인증 정보를 AWS와 교환하고 임시 보안 자격 증명 세트를 다시 가져올 수 있습니다. 자세한 내용은 SAML 2.0 연동 단원을 참조하십시오.
- AWS에 연동하려고 하고 조직에 온프레미스 자격 증명 스토어가 있습니다. 사용자가 조직 내부에서 인증할 수 있는 경우 AWS 리소스에 액세스하기 위한 임시 보안 자격 증명을 발급하는 애플리케이션을 작성할 수 있습니다. 자세한 내용은 사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기 단원을 참조하십시오.

> **Note**
>
> AWS 리소스에 프로그래밍 방식으로 액세스해야 하는 애플리케이션에서 Amazon EC2 인스턴스를 사용하고 있나요? 그렇다면 EC2용 IAM 역할을 사용하세요.

IAM 사용자 액세스 키의 올바른 관리

AWS에 대한 프로그래밍 방식 액세스를 위해 액세스 키를 생성해야 하는 경우 IAM 사용자를 위한 액세스 키를 생성하여 필요한 권한만 사용자에게 부여합니다.

IAM 사용자 액세스 키를 보호하려면 다음 예방 조치를 준수합니다.

- 액세스 키를 코드에 직접 포함하지 마십시오. AWS SDK 및 AWS 명령줄 도구를 사용하여 액세스 키를 알려진 위치에 보관합니다. 그러면 코드에 포함할 필요가 없습니다.

  액세스 키를 다음 중 한 곳에 보관하십시오.

- AWS 자격 증명 파일. AWS SDK 및 AWS CLI에서는 AWS 자격 증명 파일에 저장된 자격 증명을 자동으로 사용합니다.

  AWS 자격 증명 파일을 사용하는 방법에 대한 자세한 내용은 SDK 설명서를 참조하십시오. 예를 들어, AWS SDK for Java 개발자 안내서의 Set AWS Credentials and Region 및 AWS Command Line Interface 사용 설명서의 구성 및 보안 인증 파일을 참조하세요.

  AWS SDK for .NET 및 AWS Tools for Windows PowerShell에 대한 보안 인증을 저장하려면 SDK 스토어를 사용하는 것이 좋습니다. 자세한 내용은 AWS SDK for .NET 개발자 안내서의 Using the SDK Store를 참조하세요.

- 환경 변수. 다중 테넌트 시스템에서 시스템 환경 변수가 아닌 사용자 환경 변수를 선택합니다.

  환경 변수를 사용하여 보안 인증을 저장하는 방법에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 환경 변수를 참조하세요.

- 애플리케이션마다 서로 다른 액세스 키를 사용합니다. 그러면 권한을 격리하고 액세스 키가 노출된 경우 개별 애플리케이션에 대해 액세스 키를 취소할 수 있습니다. 또한 애플리케이션별로 별도의 액세스 키를 사용하면 AWS CloudTrail 로그 파일에 개별 항목이 생성됩니다. 이 구성을 사용하면 특정 작업을 수행한 애플리케이션을 쉽게 확인할 수 있습니다.
- 필요한 경우 액세스 키를 업데이트합니다. 액세스 키가 손상될 위험이 있는 경우 액세스 키를 업데이트하고 이전 액세스 키를 삭제합니다. 자세한 내용은 액세스 키 업데이트을 참조하세요.
- 미사용 액세스 키를 제거합니다. 사용자가 조직을 떠나는 경우 해당 IAM 사용자를 제거합니다. 그러면 사용자는 해당 리소스에 더 이상 액세스할 수 없습니다. 액세스 키가 마지막으로 사용된 시간을 확인하려면 `GetAccessKeyLastUsed` API(AWS CLI 명령: `aws iam get-access-key-last-used`)를 사용합니다.
- 가장 민감한 API 작업에 대해 임시 보안 인증을 사용하고 멀티 팩터 인증을 구성합니다. 사용자가 호출할 수 있는 API 작업을 IAM 정책을 사용해 지정할 수 있습니다. 어떤 경우에는 특히 민감한 작업을 수행하도록 허용하기 전에 AWS MFA로 사용자를 인증하도록 요구하는 추가 보안이 필요할 수 있습니다. 예를 들어 사용자가 Amazon EC2 `RunInstances`, `DescribeInstances` 및 `StopInstances` 작업을 수행하도록 허용하는 정책이 있을 수 있습니다. 하지만 `TerminateInstances`처럼 안전하지 않은 작업의 경우 이를 제한해 사용자가 AWS MFA 디바이스에서 인증할 때만 작업을 수행하도록 해야 할 필요가 있을 수 있습니다. 자세한 내용은 MFA를 통한 보안 API 액세스 단원을 참조하십시오.

AWS 액세스 키를 사용하여 모바일 앱 액세스

AWS 모바일 앱을 사용하여 일부 AWS 서비스 및 기능에 액세스할 수 있습니다. 모바일 앱을 사용하여 이동 중에도 인시던트 대응을 지원할 수 있습니다. 자세한 내용을 확인하고 앱을 다운로드하려면 AWS 콘솔 모바일 애플리케이션을 참조하십시오.

콘솔 암호 또는 액세스 키를 사용하여 모바일 앱에 로그인할 수 있습니다. 모범 사례로, 루트 사용자 액세스 키는 사용하지 않도록 합니다. 대신 모바일 디바이스에서 암호 또는 생체 인식 잠금을 사용하는 방법과 함께 모바일 앱을 사용하여 AWS 리소스 관리를 위한 특별히 IAM 사용자를 생성합니다. 모바일 디바이스를 분실한 경우 IAM 사용자의 액세스 권한을 제거할 수 있습니다.

액세스 키를 사용하여 로그인하려면(모바일 앱)

1. 모바일 디바이스에서 모바일 앱을 엽니다.
2. 디바이스에 ID를 처음 추가하는 경우 Add an identity(ID 추가)를 선택한 다음 Access keys(액세스 키)를 선택합니다.

   다른 ID를 사용하여 이미 로그인한 경우 메뉴 아이콘을 선택하고 Switch identity(ID 전환)를 선택합니다. 그러고 나서 Sign in as a different identity(다른 ID로 로그인)을 선택한 다음 Access keys(액세스 키)를 선택합니다.
3. Access keys(액세스 키) 페이지에 정보를 입력합니다.

   - Access key ID – 액세스 키 ID를 입력합니다.
   - Secret access key - 비밀 액세스 키를 입력합니다.
   - Identity name - 모바일 앱에 표시할 ID 이름을 입력합니다. IAM 사용자 이름과 일치하지 않아도 됩니다.
   - Identity PIN - 이후 로그인에 사용할 개인 식별 번호(PIN)를 생성합니다.

   > **Note**
   >
   > AWS 모바일 앱에 생체 인식을 활성화하면 PIN 대신 확인을 위해 지문 또는 안면 인식을 사용하라는 메시지가 표시됩니다. 생체 인식에 실패하면 PIN을 입력하라는 메시지가 대신 표시될 수 있습니다.
4. Verify and add keys(확인하고 키 추가)를 선택합니다.

   이제 모바일 앱을 사용하여 일부 리소스에 액세스할 수 있습니다.

관련 정보

다음 주제에서는 액세스 키를 사용하도록 AWS SDK 및 AWS CLI를 설정하는 지침을 제공합니다.

- AWS SDK for Java 개발자 안내서의 Set AWS credentials and Region
- AWS SDK for .NET 개발자 안내서의 Using the SDK Store
- AWS SDK for PHP 개발자 안내서의 Providing Credentials to the SDK
- Boto 3(Python용 AWS SDK) 문서의 Configuration
- AWS Tools for Windows PowerShell 사용 설명서의 AWS 보안 인증 사용
- AWS Command Line Interface 사용 설명서의 구성 및 보안 인증 파일
- AWS SDK for .NET 개발자 안내서의 Granting access using an IAM role
- AWS SDK for Java 2.x의 Configure IAM roles for Amazon EC2

액세스 키 감사

코드에서 AWS 액세스 키를 살펴보면 키가 자신의 계정에 속한 것인지 알 수 있습니다. 액세스 키 ID는 `aws sts get-access-key-info` AWS CLI 명령 또는 `GetAccessKeyInfo` AWS API 작업을 사용해 전달할 수 있습니다.

AWS CLI 및 AWS API 작업은 액세스 키가 속한 AWS 계정의 ID를 반환합니다. `AKIA`로 시작하는 액세스 키 ID는 IAM 사용자 또는 AWS 계정 루트 사용자을 위한 장기 자격 증명입니다. `ASIA`로 시작하는 액세스 키 ID는 AWS STS 작업으로 생성된 임시 자격 증명입니다. 응답으로 반환되는 계정이 자신의 소유라면 루트 사용자로 로그인하여 루트 사용자 액세스 키를 살펴볼 수 있습니다. 그런 다음 자격 증명 보고서를 가져와서 키를 소유하고 있는 IAM 사용자를 알아볼 수 있습니다. `ASIA` 액세스 키의 경우 누가 임시 자격 증명을 요청했는지 알아보려면 CloudTrail 로그에서 AWS STS 이벤트를 확인하세요.

보안을 위해 AWS CloudTrail 로그를 검토하여 AWS에서 누가 작업을 수행했는지 확인할 수 있습니다. 역할 신뢰 정책에서 `sts:SourceIdentity` 조건 키를 사용하여 사용자가 역할을 수임할 때 자격 증명을 지정하도록 요구할 수 있습니다. 예를 들어 IAM 사용자가 자신의 사용자 이름을 소스 자격 증명으로 지정하도록 요구할 수 있습니다. 이를 통해 AWS에서 특정 작업을 수행한 사용자를 확인할 수 있습니다. 자세한 내용은 `sts:SourceIdentity` 단원을 참조하십시오.

이 작업은 액세스 키의 상태를 표시하지 않지만 키는 활성, 비활성 또는 삭제된 상태일 수 있습니다. 활성 키에도 작업을 실행할 수 있는 권한이 없는 경우도 있습니다. 삭제된 액세스 키를 입력하면 키가 존재하지 않는다는 오류 메시지가 반환될 수 있습니다.

Amazon Keyspaces(Apache Cassandra용)와 함께 IAM 사용

Amazon Keyspaces(Apache Cassandra용)는 고가용성의 확장 가능한 관리형 Apache Cassandra 호환 데이터베이스 서비스입니다. AWS Management Console을 통해 또는 프로그래밍 방식으로 Amazon Keyspaces에 액세스할 수 있습니다. 서비스별 자격 증명을 사용하여 프로그래밍 방식으로 Amazon Keyspaces에 액세스하려면 `cqlsh` 또는 오픈 소스 Cassandra 드라이버를 사용할 수 있습니다. 서비스별 자격 증명에는 Cassandra가 인증 및 액세스 관리에 사용하는 것과 같은 사용자 이름과 암호가 포함됩니다. 사용자당 지원되는 각 서비스에 대해 최대 2개의 서비스별 보안 인증 세트를 보유할 수 있습니다.

AWS 액세스 키를 사용하여 프로그래밍 방식으로 Amazon Keyspaces에 액세스하려면 SigV4 플러그인과 함께 AWS SDK, AWS Command Line Interface(AWS CLI) 또는 오픈 소스 Cassandra 드라이버를 사용할 수 있습니다. 자세한 내용은 Amazon Keyspaces (for Apache Cassandra) Developer Guide(Amazon Keyspaces(Apache Cassandra용) 개발자 안내서)의 Connecting programmatically to Amazon Keyspaces(프로그래밍 방식으로 Amazon Keyspaces에 연결)를 참조하세요.

> **Note**
>
> 콘솔을 통해서만 Amazon Keyspaces와 상호 작용하려는 경우에는 서비스별 자격 증명을 생성할 필요가 없습니다. 자세한 내용은 Amazon Keyspaces (for Apache Cassandra) Developer Guide(Amazon Keyspaces(Apache Cassandra용) 개발자 안내서)의 Accessing Amazon Keyspaces using the console(콘솔을 사용하여 Amazon Keyspaces 액세스)을 참조하세요.

Amazon Keyspaces 액세스에 필요한 권한에 대한 자세한 내용은 Amazon Keyspaces(Apache Cassandra용) 개발자 안내서에서 Amazon Keyspaces(Apache Cassandra용) 자격 증명 기반 정책 예를 참조하세요.

Amazon Keyspaces 자격 증명 생성(콘솔)

AWS Management Console을 사용하여 IAM 사용자에 대한 Amazon Keyspaces(Apache Cassandra용) 자격 증명을 생성할 수 있습니다.

Amazon Keyspaces 서비스별 자격 증명을 생성하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택한 다음 자격 증명이 필요한 사용자의 이름을 선택합니다.

3. 보안 자격 증명(Security Credentials) 탭의 Amazon Keyspaces(Apache Cassandra용)에 대한 자격 증명(Credentials for Amazon Keyspaces (for Apache Cassandra)) 아래에서 자격 증명 생성(Generate credentials)을 선택합니다.
4. 이제 서비스별 자격 증명을 사용할 수 있습니다. 이 때가 암호를 보거나 다운로드 할 수 있는 유일한 시간입니다. 나중에 복구할 수 없습니다. 그러나 언제든지 암호를 재설정할 수 있습니다. 나중에 필요하므로 사용자와 암호를 안전한 위치에 저장하세요.

Amazon Keyspaces 자격 증명 생성(AWS CLI)

AWS CLI을 사용하여 IAM 사용자에 대한 Amazon Keyspaces(Apache Cassandra용) 자격 증명을 생성할 수 있습니다.

Amazon Keyspaces 서비스별 자격 증명을 생성하려면(AWS CLI)

- 다음 명령을 사용합니다.
  - aws iam create-service-specific-credential

Amazon Keyspaces 자격 증명 생성(AWS API)

AWS API를 사용하여 IAM 사용자에 대한 Amazon Keyspaces(Apache Cassandra용) 자격 증명을 생성할 수 있습니다.

Amazon Keyspaces 서비스별 자격 증명을 생성하려면(AWS API)

- 다음 작업을 완료합니다.
  - CreateServiceSpecificCredential

분실하거나 잊어버린 AWS 암호 또는 액세스 키 재설정

Important

AWS에 로그인하는 데 문제가 있나요? 사용자 유형에 맞는 올바른 AWS 로그인 페이지에 있는지 확인합니다. AWS 계정 루트 사용자(계정 소유자)인 경우 AWS 계정을 생성할 때 설정한 보안 인증을 사용하여 AWS에 로그인할 수 있습니다. IAM 사용자인 경우 계정 관리자가 AWS에 로그인하는 데 사용할 수 있는 자격 증명을 제공할 수 있습니다. 지원을 요청해야 하는 경우 이 페이지의 피드백 링크를 사용하지 마세요. 이 양식은 AWS Support가 아닌 AWS 설명

서 팀에서 접수합니다. 대신 Contact Us(문의처) 페이지에서 Still unable to log into your AWS account(여전히 계정에 로그인할 수 없음)을 선택한 다음 사용 가능한 지원 옵션 중 하나를 선택합니다.

기본 로그인 페이지에서 루트 사용자로 로그인하려면 이메일 주소를 입력하고 IAM 사용자로 로그인하려면 계정 ID를 입력해야 합니다. 사용자 유형과 일치하는 로그인 페이지에서만 암호를 제공할 수 있습니다. 자세한 내용은 AWS Management Console에 로그인을 참조하세요.

올바른 로그인 페이지에 있고 암호 또는 액세스 키를 분실하거나 잊어버린 경우 IAM에서 해당 항목을 복구할 수 없습니다. 그 대신 다음과 같은 방법으로 재설정할 수는 있습니다.

- AWS 계정 루트 사용자 암호 – 루트 사용자 암호를 잊어버린 경우, AWS Management Console에서 암호를 재설정할 수 있습니다. 자세한 내용은 이 주제의 후반부에 나오는 the section called “잊거나 분실한 루트 사용자 암호 재설정” 섹션을 참조하세요.
- AWS 계정 액세스 키 - 계정 액세스 키를 잊었다면 기존 액세스 키를 비활성화하지 않고 액세스 키를 새로 생성해야 합니다. 기존 키를 사용하고 있지 않았다면 삭제하면 됩니다. 자세한 내용은 루트 사용자용 액세스 키 생성 및 루트 사용자용 액세스 키 삭제 섹션을 참조하세요.
- IAM 사용자 암호 - IAM 사용자인데 암호를 잊었다면 관리자에게 암호를 재설정해 달라고 요청해야 합니다. 관리자가 암호를 관리하는 방법에 대한 자세한 내용은 IAM 사용자 암호 관리 섹션을 참조하세요.
- IAM 사용자 액세스 키 - IAM 사용자인데 액세스 키를 잊은 경우에는 새 액세스 키가 필요합니다. 고유한 액세스 키를 생성할 권한이 있다면 액세스 키 관리(콘솔) 단원에서 새 액세스 키 생성에 관한 지침을 찾아보세요. 필요한 권한이 없으면 관리자에게 액세스 키를 새로 생성해 달라고 부탁해야 합니다. 예전 키를 아직 사용하고 있다면 관리자에게 예전 키를 삭제하지 말라고 요청하세요. 관리자가 액세스 키를 관리하는 방법에 대한 자세한 내용은 IAM 사용자의 액세스 키 관리 섹션을 참조하세요.

IAM의 AWS 다중 인증

보안 강화를 위해 멀티 팩터 인증(MFA)을 구성하여 AWS 리소스를 보호하는 것이 좋습니다. AWS 계정 루트 사용자와 IAM 사용자에 대해 MFA를 활성화할 수 있습니다. 루트 사용자에 대해 활성화한 MFA는 루트 사용자 자격 증명에만 영향을 줍니다. 계정의 IAM 사용자는 자신의 자격 증명에 더하여 별도로 자격 증명을 갖게 되며, 이 별도의 자격 증명에 고유의 MFA가 구성됩니다.

AWS 계정 루트 사용자 및 IAM 사용자는 어떤 유형이든 최대 8개의 MFA 디바이스를 등록할 수 있습니다. 여러 MFA 디바이스를 등록하면 유연성이 향상되고 디바이스가 분실 또는 손상된 경우에도 액세스

가 중단될 위험을 줄일 수 있습니다. AWS Management Console에 로그인하거나 AWS CLI를 통해 세션을 생성하는 데 하나의 MFA 디바이스만 필요합니다.

> **Note**
>
> 인간 사용자가 AWS에 액세스할 때 임시 보안 인증을 사용하도록 하는 것이 좋습니다. AWS IAM Identity Center 사용을 고려해 보셨나요? IAM Identity Center를 사용하여 여러 AWS 계정에 대한 액세스를 중앙에서 관리하고 사용자에게 한 곳에서 할당된 모든 계정에 대한 MFA 보호 Single Sign-On 액세스를 제공할 수 있습니다. IAM Identity Center를 사용하면 IAM Identity Center에서 사용자 자격 증명을 생성하고 관리하거나 기존 SAML 2.0 호환 자격 증명 제공업체에 쉽게 연결할 수 있습니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 IAM Identity Center란 무엇인가요? 섹션을 참조하세요.

MFA는 AWS 웹 사이트 또는 서비스에 액세스할 때 사용자의 로그인 자격 증명 외에도 AWS가 지원되는 MFA 메커니즘의 고유 인증을 제출하라고 요청함으로써 보안을 더욱 강화합니다.

MFA 유형

AWS는 다음과 같은 MFA 유형을 지원합니다.

목차

패스키 및 보안 키

AWS Identity and Access Management는 MFA용 패스키 및 보안 키를 지원합니다. FIDO 표준에 기반한 패스키는 퍼블릭 키 암호화 기법을 사용하여 암호보다 안전한 강력한 피싱 방지 인증을 제공합니다. AWS는 디바이스 바운드 패스키(보안 키)와 동기화된 패스키라는 두 가지 유형의 패스키를 지원합니다.

- 보안 키: YubiKey처럼 2차 인증 요소로 사용되는 물리적 디바이스입니다. 하나의 보안 키가 여러 루트 사용자 계정과 IAM 사용자를 지원할 수 있습니다.
- 동기화된 패스키: Google, Apple, Microsoft 계정 같은 공급자와 1Password, Dashlane, Bitwarden 같은 서드 파티 서비스의 자격 증명 관리자를 2차 인증 요소로 사용합니다.

Apple MacBook의 Touch ID와 같은 내장된 생체 인식 인증자를 사용하여 자격 증명 관리자의 잠금을 해제하고 AWS에 로그인할 수 있습니다. 패스키는 지문, 얼굴 또는 디바이스 PIN을 사용하여 선택한 공급자와 함께 생성됩니다. 디바이스 간에 패스키를 동기화하여 AWS 로그인을 용이하게 하고 사용성과 복구 가능성을 높일 수 있습니다.

IAM은 Windows Hello에 대한 로컬 패스키 등록을 지원하지 않습니다. 패스키를 생성하고 사용하기 위해 Windows 사용자는 크로스 디바이스 인증(CDA)을 사용해야 합니다. 모바일 디바이스나 하드웨어 보안 키와 같은 한 디바이스에 있는 CDA 패스키로 노트북 등의 다른 디바이스에 로그인할 수 있습니다.

FIDO Alliance는 FIDO 사양과 호환되는 모든 FIDO 인증 제품 목록을 유지 관리합니다.

패스키 및 보안 키 활성화에 대한 자세한 내용은 루트 사용자용 패스키 또는 보안 키 활성화(콘솔) 섹션을 참조하세요.

가상 인증 애플리케이션

가상 인증 애플리케이션은 전화 또는 기타 디바이스에서 실행되고 물리적 디바이스를 에뮬레이트합니다. 가상 인증 앱은 시간 기반 일회용 암호(TOTP) 알고리즘을 구현하고 단일 디바이스에서 여러 토큰을 지원합니다. 사용자는 로그인 중에 안내에 따라 디바이스의 유효 코드를 입력해야 합니다. 사용자에게 할당된 각 토큰은 고유해야 합니다. 사용자는 다른 사용자의 토큰의 코드를 입력하여 인증할 수 없습니다.

하드웨어 구매 승인을 기다리는 동안 또는 하드웨어 도착을 기다리는 동안 가상 MFA 디바이스를 사용하는 것이 좋습니다. 가상 MFA 디바이스로 사용할 수 있는 몇 가지 지원되는 앱의 목록은 다중 인증(MFA) 섹션을 참조하세요.

IAM 사용자를 위한 가상 MFA 디바이스 설정 지침은 AWS Management Console에서 가상 MFA 디바이스 할당 섹션을 참조하세요.

하드웨어 TOTP 토큰

하드웨어 디바이스가 시간 기반 일회용 암호(TOTP) 알고리즘에 따라 6자리 숫자 코드를 생성합니다. 사용자는 로그인할 때 두 번째 웹페이지에서 디바이스의 유효 코드를 입력해야 합니다.

- 사용자에게 할당된 각 MFA 디바이스는 고유해야 합니다. 사용자는 다른 사용자의 디바이스의 코드를 입력하여 인증받을 수 없습니다. 지원되는 하드웨어 MFA 디바이스에 대한 자세한 내용은 다중 인증(MFA) 섹션을 참조하세요.
- 물리적 MFA 디바이스를 사용하려는 경우 하드웨어 TOTP 디바이스 대신 보안 키를 사용하는 것이 좋습니다. 보안 키는 배터리 요구 사항이 없고, 피싱 방지 기능이 있으며, 단일 디바이스에서 여러 사용자를 지원합니다.

패스키 또는 보안 키는 AWS Management Console에서만 활성화할 수 있으며, AWS CLI 또는 AWS API에서는 활성화할 수 없습니다. 보안 키를 활성화하려면 디바이스에 물리적으로 액세스할 수 있어야 합니다.

IAM 사용자를 위한 하드웨어 TOTP 토큰 설정 지침은 AWS Management Console에서 하드웨어 TOTP 토큰 할당 섹션을 참조하세요.

> **Note**
>
> SMS 문자 메시지 기반 MFA - AWS는 SMS 다중 인증(MFA) 활성화에 대한 지원을 종료했습니다. SMS 문자 메시지 기반 MFA를 사용하는 IAM 사용자가 있는 고객은 패스키 또는 보안 키, 가상(소프트웨어 기반) MFA 디바이스 또는 하드웨어 MFA 디바이스와 같은 대체 방법 중 하나로 전환하는 것이 좋습니다. 계정의 사용자 중에서 SMS MFA 디바이스가 할당된 사용자를 식별할 수 있습니다. IAM 콘솔의 탐색 창에서 사용자(Users)를 선택하고 표의 MFA 열에 SMS가 표시된 사용자를 찾습니다.

MFA 권장 사항

AWS 자격 증명을 안전하게 보호할 수 있도록 다음의 MFA 인증 권장 사항을 따르세요.

- AWS 계정 루트 사용자 및 AWS 계정의 IAM 사용자에 대해 여러 MFA 디바이스를 활성화하는 것이 좋습니다. 이를 통해 AWS 계정의 보안 기준을 높이고 AWS 계정 루트 사용자 등의 권한이 높은 사용자에 대한 액세스 관리를 간소화할 수 있습니다.
- 현재 지원되는 MFA 유형을 조합하여 최대 8개의 MFA 디바이스를 AWS 계정 루트 사용자 및 IAM 사용자에게 등록할 수 있습니다. MFA 디바이스가 여러 개인 경우 AWS Management Console에 로그인하거나 해당 사용자로 AWS CLI를 통해 세션을 생성하는 데 하나의 MFA 디바이스만 필요합니다. 추가 MFA 디바이스를 활성화하거나 비활성화하려면 IAM 사용자가 기존 MFA 디바이스로 인증해야 합니다.
- MFA 디바이스를 분실했거나, 도난당했거나, 액세스할 수 없는 경우 나머지 MFA 디바이스 중 하나를 사용하여 AWS 계정 복구 절차를 수행하지 않고 AWS 계정에 액세스할 수 있습니다. MFA 디바이스를 분실하거나 도난당한 경우 해당 디바이스와 연결된 IAM 보안 주체에서 연결을 해제해야 합니다.
- 여러 MFA를 사용하면 지리적으로 분산된 위치에 있거나 원격으로 작업하는 경우 직원 간에 단일 하드웨어 디바이스의 물리적 교환을 조정할 필요 없이 하드웨어 기반 MFA를 사용하여 AWS에 액세스할 수 있습니다.

- IAM 보안 주체에 추가 MFA 디바이스를 사용하면 일상적인 사용에 하나 이상의 MFA를 사용할 수 있으며, 백업 및 중복성을 위해 보관실 또는 금고와 같은 안전한 물리적 위치에 물리적 MFA 디바이스를 보관할 수 있습니다.

> 참고
>
> - FIDO 보안 키의 MFA 정보를 AWS STS API 작업으로 전달하여 임시 자격 증명을 요청할 수 없습니다.
> - AWS CLI 명령 또는 AWS API 작업을 사용하여 FIDO 보안 키를 활성화할 수 없습니다.
> - 둘 이상의 루트 또는 IAM MFA 디바이스에 동일한 이름을 사용할 수 없습니다.

추가 리소스

다음 리소스는 IAM MFA에 대해 자세히 알아보는 데 도움이 됩니다.

- AWS를 사용하여 리소스에 액세스하는 방법에 대한 자세한 내용은 MFA 지원 로그인 단원을 참조하세요.
- IAM Identity Center를 활용하여 AWS 액세스 포털, IAM Identity Center 통합 앱, AWS CLI에 대한 보안 MFA 액세스를 활성화할 수 있습니다. 자세한 내용은 IAM Identity Center에서 MFA 활성화를 참조하세요.

AWS Management Console에서 패스키 또는 보안 키 할당

패스키는 AWS 리소스를 보호하기 위해 사용할 수 있는 다중 인증(MFA) 디바이스의 한 유형입니다. AWS는 동기화된 패스키와 디바이스 바운드 패스키(보안 키라고도 함)를 지원합니다.

동기화된 패스키를 사용하면 IAM 사용자가 모든 계정에서 모든 디바이스를 다시 등록할 필요 없이 새 디바이스를 비롯한 여러 디바이스에서 FIDO 로그인 자격 증명에 액세스할 수 있습니다. 동기화된 패스키에는 Google, Apple, Microsoft 같은 퍼스트 파티 자격 증명 관리자와 1Password, Dashlane, Bitwarden 같은 서드 파티 자격 증명 관리자가 2차 인증 요소로 포함됩니다. 온디바이스 생체 인식(예: TouchID)을 사용하여 선택한 자격 증명 관리자의 잠금을 해제하여 패스키를 사용할 수도 있습니다.

또는 디바이스 바운드 패스키를 FIDO 보안 키에 바인딩하고 FIDO 보안 키를 컴퓨터의 USB 포트에 꽂은 다음 메시지가 표시되면 탭하여 로그인 절차를 안전하게 완료할 수 있습니다. 이미 다른 서비스에 FIDO 보안 키를 사용 중이고 AWS가 지원되는 구성(예: Yubico의 YubiKey 5 시리즈)을 보유한 경우

AWS에도 사용할 수 있습니다. 그렇지 않은 경우 AWS의 MFA에 WebAuthn을 사용하려면 FIDO 보안 키를 구입해야 합니다. 또한 FIDO 보안 키는 동일한 기기에서 여러 IAM 또는 루트 사용자를 지원하여 계정 보안을 위한 유틸리티를 강화할 수 있습니다. 두 디바이스 유형의 사양 및 구입 관련 정보는 멀티 팩터 인증 섹션을 참조하세요.

현재 지원되는 MFA 유형을 조합하여 최대 8개의 MFA 디바이스를 AWS 계정 루트 사용자 및 IAM 사용자에게 등록할 수 있습니다. MFA 디바이스가 여러 개인 경우 AWS Management Console에 로그인하거나 해당 사용자로 AWS CLI를 통해 세션을 생성하는 데 하나의 MFA 디바이스만 필요합니다. 여러 개의 MFA 디바이스를 등록하는 것이 좋습니다. 예를 들어 내장된 인증자를 등록하고 물리적으로 안전한 위치에 보관하는 보안 키를 등록할 수 있습니다. 내장된 인증자를 사용할 수 없는 경우 등록된 보안 키를 사용할 수 있습니다. 인증 애플리케이션의 경우 인증 앱이 포함된 디바이스를 분실하거나 고장이 발생한 경우 계정에 대한 액세스 권한을 잃지 않도록 해당 앱에서 클라우드 백업 또는 동기화 기능을 활성화하는 것이 좋습니다.

> **Note**
>
> 인간 사용자가 AWS에 액세스할 때 임시 보안 인증을 사용하도록 하는 것이 좋습니다. 사용자는 ID 제공업체와 함께 AWS에 연동하여 회사 보안 인증 및 MFA 구성으로 인증할 수 있습니다. AWS 및 비즈니스 애플리케이션에 대한 액세스를 관리하려면 IAM Identity Center를 사용하는 것이 좋습니다. 자세한 내용은 IAM Identity Center 사용 설명서를 참조하세요.

주제

- 필요한 권한
- 자체 IAM 사용자 패스키 또는 보안 키 활성화(콘솔)
- 다른 IAM 사용자에 대한 패스키 또는 보안 키 활성화(콘솔)
- 패스키 또는 보안 키 교체
- 패스키 및 보안 키 사용이 지원되는 구성

필요한 권한

민감한 MFA 관련 작업을 보호하면서 자체 IAM 사용자의 FIDO 패스키를 관리하려면 다음 정책에 따른 권한이 있어야 합니다.

> **Note**
>
> ARN 값은 정적 값이며 인증을 등록하는 데 사용된 프로토콜을 나타내는 지표가 아닙니다. U2F는 더 이상 사용되지 않으므로 모든 새 구현에서 WebAuthn을 사용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowManageOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "DenyAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

자체 IAM 사용자 패스키 또는 보안 키 활성화(콘솔)

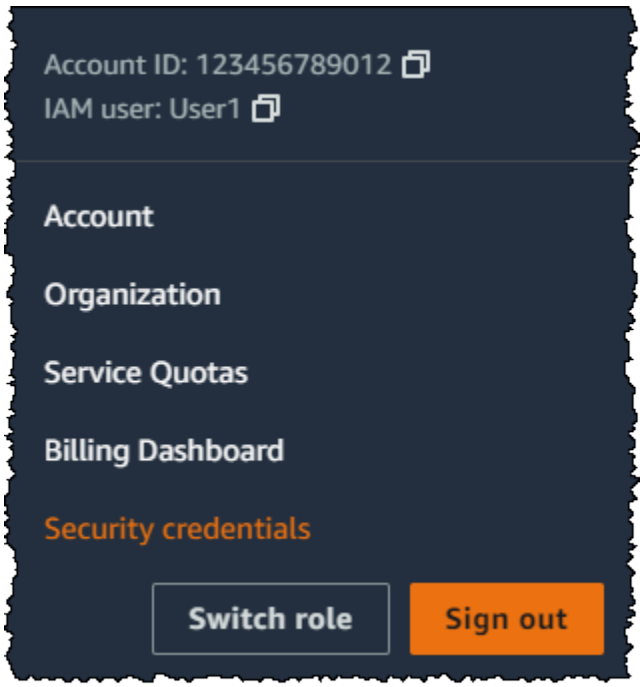
자체 IAM 사용자의 패스키 또는 보안 키는 AWS Management Console에서만 활성화할 수 있으며, AWS CLI 또는 AWS API에서는 활성화할 수 없습니다. 보안 키를 활성화하려면 디바이스에 물리적으로 액세스할 수 있어야 합니다.

자체 IAM 사용자 패스키 또는 보안 키를 활성화하려면(콘솔)

1. AWS 계정 ID나 계정 별칭, IAM 사용자 이름 및 암호를 사용하여 IAM 콘솔에 로그인합니다.

   > **Note**
   >
   > 사용자 편의를 위해 AWS 로그인 페이지는 브라우저 쿠키를 사용하여 IAM 사용자 이름 및 계정 정보를 기억합니다. 이전에 다른 사용자로 로그인한 경우 페이지 하단 근처의 다른 계정에 로그인(Sign in to a different account)을 선택하여 기본 로그인 페이지로 돌아갑니다. 여기서 AWS 계정 ID 또는 계정 별칭을 입력하면 계정의 IAM 사용자 로그인 페이지로 리디렉션됩니다.

   AWS 계정 ID를 받으려면 관리자에게 문의하세요.
2. 오른쪽 상단의 탐색 모음에서 사용자 이름을 선택한 다음 Security credentials(보안 자격 증명)를 선택합니다.



3. 선택한 IAM 사용자의 페이지에서 보안 자격 증명 탭을 선택합니다.

4. Multi-factor authentication (MFA)(다중 인증(MFA)) 섹션에서 Assign MFA device(MFA 디바이스 할당)를 선택합니다.
5. MFA 디바이스 이름 페이지에서 디바이스 이름을 입력하고 패스키 또는 보안 키를 선택한 후 다음을 선택합니다.
6. 디바이스 설정에서 패스키를 설정합니다. 얼굴이나 지문 같은 생체 인식 데이터 또는 디바이스 PIN을 사용하거나 컴퓨터의 USB 포트에 FIDO 보안 키를 삽입한 다음 탭하여 패스키를 생성합니다.
7. 브라우저의 지침을 따른 다음 계속을 선택합니다.

이제 AWS에서 사용할 패스키 또는 보안 키를 등록했습니다. AWS Management Console의 MFA 사용 방법에 대한 자세한 내용은 MFA 지원 로그인 섹션을 참조하세요.

다른 IAM 사용자에 대한 패스키 또는 보안 키 활성화(콘솔)

AWS Management Console에서만 다른 IAM 사용자에 대한 패스키 또는 보안 키를 활성화할 수 있으며, AWS CLI 또는 AWS API에서는 활성화할 수 없습니다.

다른 IAM 사용자에 대한 패스키 또는 보안 키를 활성화하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 사용자에서 MFA를 활성화하려는 사용자의 이름을 선택합니다.
4. 선택한 IAM 사용자 페이지에서 보안 자격 증명명 탭을 선택합니다.
5. Multi-factor authentication (MFA)(다중 인증(MFA)) 섹션에서 Assign MFA device(MFA 디바이스 할당)를 선택합니다.
6. MFA 디바이스 이름 페이지에서 디바이스 이름을 입력하고 패스키 또는 보안 키를 선택한 후 다음을 선택합니다.
7. 디바이스 설정에서 패스키를 설정합니다. 얼굴이나 지문 같은 생체 인식 데이터 또는 디바이스 PIN을 사용하거나 컴퓨터의 USB 포트에 FIDO 보안 키를 삽입한 다음 탭하여 패스키를 생성합니다.
8. 브라우저의 지침을 따른 다음 계속을 선택합니다.

이제 AWS에서 사용할 다른 IAM 사용자의 패스키 또는 보안 키를 등록했습니다. AWS Management Console의 MFA 사용 방법에 대한 자세한 내용은 MFA 지원 로그인 섹션을 참조하세요.

패스키 또는 보안 키 교체

현재 지원되는 MFA 유형을 조합하여 한 번에 최대 8개의 MFA 디바이스를 사용자에게 AWS 계정 루트 사용자 및 IAM 사용자와 함께 할당할 수 있습니다. 사용자가 FIDO 인증을 분실했거나 어떤 이유로든 교체해야 하는 경우 먼저 이전 FIDO 인증을 비활성화해야 합니다. 그런 다음, 해당 사용자를 위한 새 MFA 디바이스를 추가할 수 있습니다.

- 현재 IAM 사용자와 연결된 디바이스를 비활성화하는 방법은 MFA 디바이스 비활성화 섹션을 참조하세요.
- IAM 사용자에 대한 새 FIDO 보안 키를 추가하려면 자체 IAM 사용자 패스키 또는 보안 키 활성화(콘솔) 섹션을 참조하세요.

새로운 패스키 또는 보안 키에 대한 액세스 권한이 없는 경우 새로운 가상 MFA 디바이스 또는 하드웨어 TOTP 토큰을 활성화할 수 있습니다. 관련 지침을 보려면 다음 중 하나를 참조하세요.

- AWS Management Console에서 가상 MFA 디바이스 할당
- AWS Management Console에서 하드웨어 TOTP 토큰 할당

패스키 및 보안 키 사용이 지원되는 구성

현재 지원되는 구성을 사용하여 IAM에서 FIDO2 디바이스 바운드 패스키(보안 키라고도 함)를 다중 인증(MFA) 방법으로 사용할 수 있습니다. 여기에는 IAM에서 지원하는 FIDO2 디바이스와 FIDO2를 지원하는 브라우저가 포함됩니다. FIDO2 장치를 등록하기 전에 최신 브라우저 및 운영 체제(OS) 버전을 사용하고 있는지 확인하세요. 기능은 브라우저, 인증자, OS 클라이언트마다 다르게 작동할 수 있습니다. 한 브라우저에서 디바이스 등록이 실패하는 경우 다른 브라우저로 등록을 시도할 수 있습니다.

FIDO2는 퍼블릭 키 암호화를 기반으로 동일한 높은 수준의 보안을 제공하는 개방형 인증 표준이자 FIDO U2F의 확장입니다. FIDO2는 W3C 웹 인증 사양(WebAuthn API)과 애플리케이션 계층 프로토콜인 FIDO Alliance CTAP(Client-to-Authenticator Protocol)로 구성됩니다. CTAP는 외부 인증자를 사용하여 브라우저나 운영 체제와 같은 플랫폼이나 클라이언트 간의 통신을 가능하게 합니다. AWS에서 FIDO 인증 인증자를 활성화하면 보안 키는 AWS에서만 사용할 새 키 페어를 생성합니다. 먼저 자격 증명을 입력합니다. 메시지가 나타나면 보안 키를 탭하여 AWS가 발급한 인증 챌린지에 응답합니다. FIDO2 표준에 대한 자세한 내용을 알아보려면 FIDO2 Project(FIDO2 프로젝트)를 참조하세요.

AWS에서 지원하는 FIDO2 디바이스

IAM은 USB, Bluetooth 또는 NFC를 통해 디바이스에 연결하는 FIDO2 보안 디바이스를 지원합니다. IAM은 TouchID, FaceID 등의 플랫폼 인증자도 지원합니다. IAM은 Windows Hello에 대한 로컬 패스키

등록을 지원하지 않습니다. 패스키를 생성하고 사용하려면 Windows 사용자는 모바일 디바이스와 같은 한 디바이스의 패스키나 하드웨어 보안 키를 사용하여 노트북 등의 다른 디바이스에 로그인하는 크로스 디바이스 인증을 사용해야 합니다.

> **Note**
>
> AWS의 경우 FIDO2 디바이스 검사를 위해 컴퓨터에 있는 물리적 USB 포트에 대한 액세스가 필요합니다. 보안 키는 가상 머신, 원격 연결 또는 브라우저의 익명 모드에서 작동하지 않습니다.

FIDO Alliance는 FIDO 사양과 호환되는 모든 FIDO2 제품 목록을 유지 관리합니다.

FIDO2 지원 브라우저

웹 브라우저에서 실행되는 FIDO2 보안 장치의 사용 가능 여부는 어떤 브라우저와 운영 체제를 함께 사용하는지에 따라 달라집니다. 현재 보안 키 사용을 지원하는 브라우저는 다음과 같습니다.

웹 브라우저	macOS 10.15 이상	Windows 10	Linux	iOS 14.5 이상	Android 7 이상
Chrome	예	예	예	예	아니요
Safari	예	아니요	아니요	예	아니요
Edge	예	예	아니요	예	아니요
Firefox	예	예	아니요	예	아니요

> **Note**
>
> 현재 FIDO2를 지원하는 대부분의 Firefox 버전에서는 기본적으로 지원을 활성화하지 않습니다. Firefox에서 FIDO2 지원을 활성하기 위한 지침은 FIDO 보안 키 문제 해결 섹션을 참조하세요.

FIDO2 인증 디바이스(예: YubiKey)의 브라우저 지원에 대한 자세한 내용은 FIDO2 및 U2F에 대한 운영 체제 및 웹 브라우저 지원을 참조하세요.

브라우저 플러그 인

AWS는 FIDO2를 기본적으로 지원하는 브라우저만 지원합니다. AWS는 FIDO2 브라우저 지원을 추가하기 위한 플러그인 사용을 지원하지 않습니다. 일부 브라우저 플러그인은 FIDO2 표준과 호환되지 않으며 FIDO2 보안 키와 연결할 때 예기치 않은 결과를 초래할 수 있습니다.

브라우저 플러그인 비활성화 및 기타 문제 해결을 위한 자세한 내용은 FIDO 보안 키를 활성화할 수 없습니다. 섹션을 참조하세요.

디바이스 인증

보안 키를 등록하는 동안에만 FIPS 검증 및 FIDO 인증 등급과 같은 디바이스 관련 인증을 캡처하고 할당합니다. 디바이스 인증은 FIDO Alliance 메타데이터 서비스(MDS)에서 가져옵니다. 보안 키의 인증 상태 또는 등급이 변경될 경우, 디바이스 태그에 자동으로 반영되지 않습니다. 디바이스의 인증 정보를 업데이트하려면 디바이스를 다시 등록하여 업데이트된 인증 정보를 가져와야 합니다.

AWS는 디바이스 등록 시 FIDO MDS에서 획득한 FIPS-140-2, FIPS-140-3 및 FIDO 인증 등급과 같은 인증 유형을 조건 키로 제공합니다. 선호하는 인증 유형 및 등급에 따라 IAM 정책에서 특정 인증자의 등록을 지정할 수 있습니다. 자세한 내용은 아래의 정책을 참조하세요.

디바이스 인증을 위한 예제 정책

다음 사용 사례는 FIPS 인증을 사용하여 MFA 디바이스를 등록할 수 있도록 허용하는 샘플 정책을 보여줍니다.

주제

- 사용 사례 1: FIPS-140-2 L2 인증을 받은 디바이스만 등록 허용
- 사용 사례 2: FIPS-140-2 L2 및 FIDO L1 인증을 받은 디바이스의 등록 허용
- 사용 사례 3: FIPS-140-2 L2 또는 FIPS-140-3 L2 인증을 받은 디바이스의 등록 허용
- 사용 사례 4: FIPS-140-2 L2 인증이 있고 가상 인증 및 하드웨어 TOTP와 같은 기타 MFA 유형을 지원하는 디바이스의 등록 허용

사용 사례 1: FIPS-140-2 L2 인증을 받은 디바이스만 등록 허용

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-2-certification": "L2"
                }
            }
        }
    ]
}
```

사용 사례 2: FIPS-140-2 L2 및 FIDO L1 인증을 받은 디바이스의 등록 허용

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-2-certification": "L2",
```

```
                    "iam:FIDO-certification": "L1"
                }
            }
        }
    ]
}
```

사용 사례 3: FIPS-140-2 L2 또는 FIPS-140-3 L2 인증을 받은 디바이스의 등록 허용

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-2-certification": "L2"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-3-certification": "L2"
                }
            }
        }
```

```
    ]
}
```

사용 사례 4: FIPS-140-2 L2 인증이 있고 가상 인증 및 하드웨어 TOTP와 같은 기타 MFA 유형을 지원하는 디바이스의 등록 허용

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey": "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey": "Activate",
                    "iam:FIPS-140-2-certification": "L2"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "Null": {
                    "iam:RegisterSecurityKey": "true"
                }
            }
        }
    ]
}
```

AWS CLI 및 AWS API

AWS는 AWS Management Console에서만 패스키 및 보안 키 사용을 지원합니다. MFA에서의 패스키 및 보안 키 사용은 AWS CLI 및 AWS API 또는 MFA 보호 API 작업에 대한 액세스에서는 지원되지 않습니다.

추가 리소스

- AWS에서의 패스키 및 보안 키 사용에 대한 자세한 내용은 AWS Management Console에서 패스키 또는 보안 키 할당 섹션을 참조하세요.
- AWS에서의 패스키 및 보안 키 문제 해결에 대한 도움말은 FIDO 보안 키 문제 해결 섹션을 참조하세요.
- FIDO2 지원에 대한 일반적인 업계 정보는 FIDO2 Project(FIDO2 프로젝트)를 참조하세요.

AWS Management Console에서 가상 MFA 디바이스 할당

스마트폰 또는 기타 디바이스를 가상 MFA 디바이스로 사용할 수 있습니다. 이를 위해서는 표준 기반 TOTP(시간 기반 일회용 암호) 알고리즘인 RFC 6238과 호환되는 모바일 앱을 설치해야 합니다. 이러한 앱에서는 6자리 인증 코드가 생성됩니다. 가상 MFA는 안전하지 않은 모바일 디바이스에서 실행될 수 있으므로 FIDO 보안 키와 동일한 수준의 보안을 제공하지 않을 수 있습니다. 하드웨어 구매 승인을 기다리는 동안 또는 하드웨어 도착을 기다리는 동안 가상 MFA 디바이스를 사용하는 것이 좋습니다.

대부분의 가상 MFA 앱은 여러 개의 가상 디바이스 생성을 지원하므로 여러 개의 AWS 계정이나 사용자에게 동일한 앱을 사용할 수 있습니다. MFA 유형을 조합하여 최대 8개의 MFA 디바이스를 AWS 계정 루트 사용자 및 IAM 사용자에게 등록할 수 있습니다. AWS Management Console에 로그인하거나 AWS CLI를 통해 세션을 생성하는 데 하나의 MFA 디바이스만 필요합니다. 여러 개의 MFA 디바이스를 등록하는 것이 좋습니다. 인증 애플리케이션의 경우 디바이스를 분실하거나 고장이 발생한 경우 계정에 대한 액세스 권한을 잃지 않도록 클라우드 백업 또는 동기화 기능을 활성화하는 것이 좋습니다.

AWS에서 사용하려면 가상 MFA 앱이 6자리 OTP를 생성해야 합니다. 사용할 수 있는 가상 MFA 앱 목록은 멀티 팩터 인증 섹션을 참조하세요.

주제

- 필요한 권한
- IAM 사용자에 대한 가상 MFA 디바이스 활성화(콘솔)
- 가상 MFA 디바이스 교체

필요한 권한

IAM 사용자의 가상 MFA 디바이스를 관리하려면 AWS: MFA 인증 IAM 사용자가 보안 인증 페이지에서 자신의 MFA 디바이스를 관리할 수 있도록 허용 정책에 따른 권한이 있어야 합니다.

IAM 사용자에 대한 가상 MFA 디바이스 활성화(콘솔)

AWS Management Console에서 IAM을 사용하여 계정의 IAM 사용자를 위한 가상 MFA 디바이스를 활성화 및 관리할 수 있습니다. 가상 MFA 디바이스를 비롯한 IAM 리소스에 태그를 연결하여 액세스를 식별, 구성 및 제어할 수 있습니다. AWS CLI 또는 AWS API를 사용하는 경우에만 가상 MFA 디바이스를 태깅할 수 있습니다. AWS CLI 또는 AWS API를 사용하여 MFA 장치를 활성화하고 관리하려면 AWS CLI 또는 AWS API에서 MFA 디바이스 할당 섹션을 참조하세요. IAM 리소스 태깅에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요.

> **Note**
>
> MFA를 구성하려면 사용자의 가상 MFA 디바이스가 호스팅되는 하드웨어에 대한 물리적 액세스가 필요합니다. 예를 들어, 스마트폰에서 가상 MFA 디바이스를 실행하는 사용자에게 MFA를 구성할 수 있습니다. 이 경우 마법사를 완료하기 위해 스마트폰을 사용할 수 있어야 합니다. 이러한 이유로 사용자가 자신의 가상 MFA 디바이스를 직접 구성 및 관리할 수 있도록 허용하는 것이 좋습니다. 이 경우에는 사용자에게 필요한 IAM 작업을 수행할 권한을 부여해야 합니다. 이러한 권한을 부여하는 IAM 정책에 대한 자세한 내용과 예는 IAM 자습서: 사용자가 자신의 자격 증명 및 MFA 설정을 관리하도록 허용 및 예제 정책 AWS: MFA 인증 IAM 사용자가 보안 인증 페이지에서 자신의 MFA 디바이스를 관리할 수 있도록 허용의 내용을 참조하세요.

IAM 사용자에 대한 가상 MFA 디바이스를 활성화하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 사용자 목록에서 IAM 사용자 이름을 선택합니다.
4. [Security Credentials] 탭을 선택합니다. Multi-factor authentication (MFA)(다중 인증(MFA)) 섹션에서 Assign MFA device(MFA 디바이스 할당)를 선택합니다.
5. 마법사에서 디바이스 이름을 입력하고 인증 앱을 선택한 후 다음을 선택합니다.

IAM은 QR 코드 그래픽을 포함하여 가상 MFA 디바이스의 구성 정보를 생성 및 표시합니다. 그래픽은 QR 코드를 지원하지 않는 디바이스 상에서 수동 입력할 수 있는 '보안 구성 키'를 표시한 것입니다.

6. 가상 MFA 앱을 엽니다. 가상 MFA 디바이스의 호스팅에 사용되는 앱 목록은 멀티 팩터 인증을 참조하세요.

   가상 MFA 앱이 다수의 가상 MFA 디바이스 또는 계정을 지원하는 경우 새로운 가상 MFA 디바이스 또는 계정을 생성하는 옵션을 선택합니다.
7. MFA 앱의 QR 코드 지원 여부를 결정한 후 다음 중 한 가지를 실행합니다.

   - 마법사에서 Show QR code(QT 코드 표시)를 선택한 다음 해당 앱을 사용하여 QR 코드를 스캔합니다. 디바이스의 카메라를 사용하여 코드를 스캔하는 카메라 아이콘 또는 스캔 모드 옵션이 있을 수 있습니다.
   - 마법사에서 Show secret key(보안 키 표시)를 선택한 다음 MFA 앱에 보안 키를 입력합니다.

   모든 작업을 마치면 가상 MFA 디바이스가 일회용 암호 생성을 시작합니다.
8. MFA 디바이스 설정 페이지의 MFA 코드 1 상자에 현재 가상 MFA 디바이스에 표시된 일회용 암호를 입력합니다. 디바이스가 새로운 일회용 암호를 생성할 때까지 최대 30초 기다립니다. 그런 다음 두 번째 일회용 암호를 MFA code 2(MFA 코드 2) 상자에 입력합니다. Add MFA(MFA 추가)를 선택합니다.

   > **Important**
   >
   > 코드를 생성한 후 즉시 요청을 제출하세요. 코드를 생성한 후 너무 오래 기다렸다 요청을 제출할 경우 MFA 디바이스가 사용자와 연결은 되지만 MFA 디바이스가 동기화되지 않습니다. 이는 시간 기반 일회용 암호(TOTP)가 잠시 후에 만료되기 때문입니다. 이 경우, 디바이스를 재동기화할 수 있습니다.

이제 AWS에서 가상 MFA 디바이스를 사용할 준비가 끝났습니다. AWS Management Console의 MFA 사용 방법에 대한 자세한 내용은 MFA 지원 로그인 섹션을 참조하세요.

가상 MFA 디바이스 교체

MFA 유형을 조합하여 최대 8개의 MFA 디바이스를 AWS 계정 루트 사용자 및 IAM 사용자에게 등록할 수 있습니다. 사용자가 디바이스를 분실하거나 이유를 불문하고 교체할 필요가 있을 경우, 기존 디바이스를 비활성화해야 합니다. 그런 다음, 해당 사용자를 위한 새 디바이스를 추가할 수 있습니다.

- 현재 다른 IAM 사용자와 연결되어 있는 디바이스를 비활성화하는 방법은 MFA 디바이스 비활성화 섹션을 참조하세요.
- 다른 IAM 사용자에 대해 교체용 가상 MFA 디바이스를 추가하려면 위의 IAM 사용자에 대한 가상 MFA 디바이스 활성화(콘솔) 절차에 나와 있는 단계를 따르세요.
- AWS 계정 루트 사용자를 위한 교체용 가상 MFA 디바이스를 추가하려면 루트 사용자용 가상 MFA 디바이스 활성화(콘솔) 절차에 나와 있는 단계를 따릅니다.

AWS Management Console에서 하드웨어 TOTP 토큰 할당

하드웨어 TOTP 토큰은 시간 기반 일회용 암호(TOTP) 알고리즘에 따라 6자리 숫자 코드를 생성합니다. 사용자는 로그인 과정 중 디바이스의 유효 코드를 입력해야 합니다. 사용자에게 할당된 각 MFA 디바이스는 고유해야 합니다. 사용자는 다른 사용자의 디바이스 코드를 입력하여 인증받을 수 없습니다. MFA 디바이스는 계정이나 사용자 간에 공유할 수 없습니다.

하드웨어 TOTP 토큰과 FIDO 보안 키는 모두 본인이 구입한 물리적 디바이스여야 합니다. 하드웨어 MFA 디바이스는 AWS에 로그인할 때 인증을 위한 TOTP 코드를 생성합니다. 이는 배터리를 사용하므로 시간이 지남에 따라 교체 및 AWS 재동기화가 필요할 수 있습니다. 공개 키 암호화를 활용하는 FIDO 보안 키는 배터리가 필요 없으며 원활한 인증 프로세스를 제공합니다. 피싱 방지를 위해 TOTP 장치보다 안전한 FIDO 보안 키를 사용하는 것이 좋습니다. 또한 FIDO 보안 키는 동일한 기기에서 여러 IAM 또는 루트 사용자를 지원하여 계정 보안을 위한 유틸리티를 강화할 수 있습니다. 두 디바이스 유형의 사양 및 구입 관련 정보는 멀티 팩터 인증 섹션을 참조하세요.

AWS Management Console, 명령줄 또는 IAM API에서 IAM 사용자의 하드웨어 TOTP 토큰을 활성화할 수 있습니다. AWS 계정 루트 사용자에 따른 MFA 디바이스 활성화 방법은 루트 사용자에 대해 하드웨어 TOTP 토큰 활성화(콘솔) 단원을 참조하세요.

현재 지원되는 MFA 유형을 조합하여 최대 8개의 MFA 디바이스를 AWS 계정 루트 사용자 및 IAM 사용자에게 등록할 수 있습니다. MFA 디바이스가 여러 개인 경우 AWS Management Console에 로그인하거나 해당 사용자로 AWS CLI를 통해 세션을 생성하는 데 하나의 MFA 디바이스만 필요합니다.

> **Important**
>
> MFA 디바이스를 분실했거나 액세스할 수 없는 경우 사용자가 계정에 계속 액세스할 수 있도록 여러 MFA 디바이스를 활성화하는 것이 좋습니다.

> **Note**
>
> 명령줄에서 MFA 디바이스를 활성화하려는 경우 `aws iam enable-mfa-device`를 사용합니다. IAM API를 사용하여 MFA 디바이스를 활성화하려면 `EnableMFADevice` 작업을 사용합니다.

주제

필요한 권한

중요한 MFA 관련 작업을 보호하면서 자신의 IAM 사용자에 대한 하드웨어 TOTP 토큰을 관리하려면 다음 정책에 따른 권한이 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowManageOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "DenyAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:EnableMFADevice",
                "iam:GetUser",
```

```
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

자신의 IAM 사용자에 대해 하드웨어 TOTP 토큰 활성화(콘솔)

AWS Management Console에서 자신의 하드웨어 TOTP 토큰을 활성화할 수 있습니다.

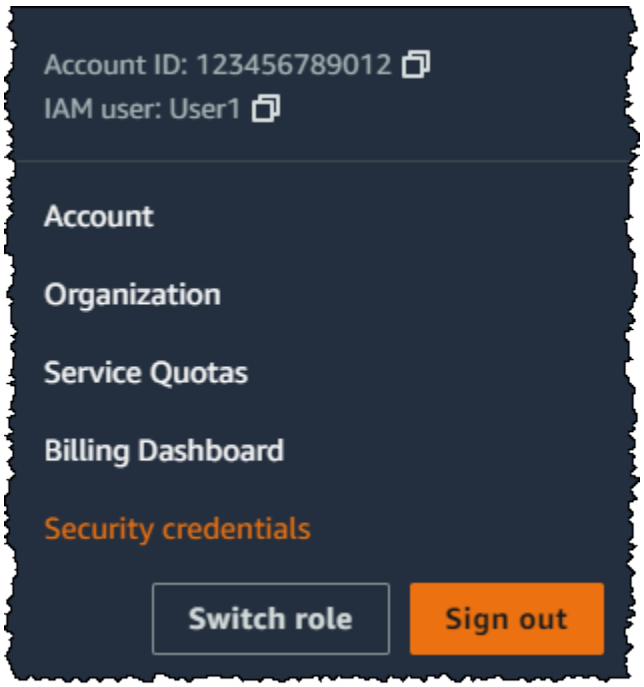
> **Note**
>
> 하드웨어 TOTP 토큰을 활성화하려면 디바이스에 물리적으로 액세스할 수 있어야 합니다.

자신의 IAM 사용자에 대해 하드웨어 TOTP 토큰을 활성화하려면 다음을 수행하세요(콘솔).

1. AWS 계정 ID나 계정 별칭, IAM 사용자 이름 및 암호를 사용하여 IAM 콘솔에 로그인합니다.

   > **Note**
   >
   > 사용자 편의를 위해 AWS 로그인 페이지는 브라우저 쿠키를 사용하여 IAM 사용자 이름 및 계정 정보를 기억합니다. 이전에 다른 사용자로 로그인한 경우 페이지 하단 근처의 다른 계정에 로그인(Sign in to a different account)을 선택하여 기본 로그인 페이지로 돌아갑니다. 여기서 AWS 계정 ID 또는 계정 별칭을 입력하면 계정의 IAM 사용자 로그인 페이지로 리디렉션됩니다.

   AWS 계정 ID를 받으려면 관리자에게 문의하세요.
2. 오른쪽 상단의 탐색 모음에서 사용자 이름을 선택한 다음 Security credentials(보안 자격 증명)를 선택합니다.



3. AWS IAM credentials 탭의 Multi-factor authentication (MFA)(다중 인증(MFA)) 섹션에서 Assign MFA device(MFA 디바이스 할당)를 선택합니다.
4. 마법사에서 Device name(디바이스 이름)을 입력하고 Hardware TOTP token(하드웨어 TOTP 토큰), Next(다음)를 차례로 선택합니다.
5. 디바이스 일련 번호를 입력합니다. 일련 번호는 보통 디바이스 후면에 있습니다.
6. MFA code 1(MFA 코드 1) 상자에 MFA 디바이스에 표시된 6자리 번호를 입력합니다. 디바이스 전면의 버튼을 눌러야 번호가 표시되는 경우도 있습니다.

7. 디바이스가 코드를 새로 고칠 때까지 30초 동안 기다린 다음 MFA code 2(MFA 코드 2) 상자에 다음 6자리 번호를 입력합니다. 다시 디바이스 전면의 버튼을 눌러야 두 번째 번호가 표시되는 경우도 있습니다.
8. Add MFA(MFA 추가)를 선택합니다.

> **Important**
>
> 인증 코드를 생성한 후 바로 요청을 제출하세요. 코드를 생성한 후 너무 오래 기다렸다 요청을 제출할 경우 MFA 디바이스가 사용자와 연결은 되지만 MFA 디바이스가 동기화되지

> 않습니다. 이는 시간 기반 일회용 암호(TOTP)가 잠시 후에 만료되기 때문입니다. 이 경우, 디바이스를 재동기화할 수 있습니다.

이제 AWS에서 디바이스를 사용할 준비가 끝났습니다. AWS Management Console의 MFA 사용 방법에 대한 자세한 내용은 MFA 지원 로그인 섹션을 참조하세요.

다른 IAM 사용자에 대해 하드웨어 TOTP 토큰 활성화(콘솔)

AWS Management Console에서 다른 IAM 사용자에 대해 하드웨어 TOTP 토큰을 활성화할 수 있습니다.

다른 IAM 사용자에 대해 하드웨어 TOTP 토큰을 활성화하려면 다음을 수행하세요(콘솔).

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. MFA를 활성화하려는 사용자의 이름을 선택합니다.
4. [Security Credentials] 탭을 선택합니다. Multi-factor authentication (MFA)(다중 인증(MFA)) 섹션에서 Assign MFA device(MFA 디바이스 할당)를 선택합니다.
5. 마법사에서 Device name(디바이스 이름)을 입력하고 Hardware TOTP token(하드웨어 TOTP 토큰), Next(다음)를 차례로 선택합니다.
6. 디바이스 일련 번호를 입력합니다. 일련 번호는 보통 디바이스 후면에 있습니다.
7. MFA code 1(MFA 코드 1) 상자에 MFA 디바이스에 표시된 6자리 번호를 입력합니다. 디바이스 전면의 버튼을 눌러야 번호가 표시되는 경우도 있습니다.

8. 디바이스가 코드를 새로 고칠 때까지 30초 동안 기다린 다음 MFA code 2(MFA 코드 2) 상자에 다음 6자리 번호를 입력합니다. 다시 디바이스 전면의 버튼을 눌러야 두 번째 번호가 표시되는 경우도 있습니다.
9. Add MFA(MFA 추가)를 선택합니다.

> **Important**
>
> 인증 코드를 생성한 후 바로 요청을 제출하세요. 코드를 생성한 후 너무 오래 기다렸다 요청을 제출할 경우 MFA 디바이스가 사용자와 연결은 되지만 MFA 디바이스가 동기화되지 않습니다. 이는 시간 기반 일회용 암호(TOTP)가 잠시 후에 만료되기 때문입니다. 이 경우, 디바이스를 재동기화할 수 있습니다.

이제 AWS에서 디바이스를 사용할 준비가 끝났습니다. AWS Management Console의 MFA 사용 방법에 대한 자세한 내용은 MFA 지원 로그인 섹션을 참조하세요.

물리적 MFA 디바이스 교체

현재 지원되는 MFA 유형을 조합하여 한 번에 최대 8개의 MFA 디바이스를 사용자에게 AWS 계정 루트 사용자 및 IAM 사용자와 함께 할당할 수 있습니다. 사용자가 디바이스를 분실하거나 이유를 불문하고 교체할 필요가 있을 경우, 먼저 기존 디바이스를 비활성화해야 합니다. 그런 다음, 해당 사용자를 위한 새 디바이스를 추가할 수 있습니다.

- 현재 어떤 사용자와 연결되어 있는 디바이스를 비활성화하는 방법은 MFA 디바이스 비활성화 섹션을 참조하세요.
- IAM 사용자에 대해 교체 하드웨어 TOTP 토큰을 추가하려면, 이 주제 앞부분의 다른 IAM 사용자에 대해 하드웨어 TOTP 토큰 활성화(콘솔) 절차에 나오는 단계를 따르세요.
- AWS 계정 루트 사용자 사용자에 대해 교체 하드웨어 TOTP 토큰을 추가하려면, 이 주제 앞부분의 절차 루트 사용자에 대해 하드웨어 TOTP 토큰 활성화(콘솔)에 나오는 단계를 따르세요.

AWS CLI 또는 AWS API에서 MFA 디바이스 할당

AWS CLI 명령 또는 AWS API 작업을 사용하여 IAM 사용자를 위한 가상 MFA 디바이스를 활성화할 수 있습니다. AWS CLI, AWS API, Tools for Windows PowerShell 또는 기타 다른 명령줄 도구를 사용하면 AWS 계정 루트 사용자에 대해 MFA 디바이스를 활성화할 수 없습니다. 그러나 AWS Management Console을 사용하여 루트 사용자의 MFA 디바이스를 활성화할 수 있습니다.

AWS Management Console에서 MFA 디바이스를 활성화할 때 콘솔이 사용자를 대신해 여러 단계를 수행합니다. 대신 AWS CLI, Tools for Windows PowerShell 또는 AWS API를 사용해 가상 디바이스를 생성한다면 수동으로 올바른 순서에 따라 단계들을 수행해야 합니다. 예를 들어 가상 MFA 디바이스를 생성하려면 IAM 객체를 생성하고, 코드를 문자열이나 QR 코드 그래픽으로 추출해야 합니다. 그런 다음 디바이스를 동기화하여 IAM 사용자와 연결해야 합니다. 자세한 정보는 New-

IAMVirtualMFADevice의 Examples 섹션을 참조하세요. 물리적 디바이스를 위해서는 생성 단계를 건너뛰고 디바이스를 동기화하고 사용자에게 직접 연결합니다.

가상 MFA 디바이스를 비롯한 IAM 리소스에 태그를 연결하여 액세스를 식별, 구성 및 제어할 수 있습니다. AWS CLI 또는 AWS API를 사용하는 경우에만 가상 MFA 디바이스를 태깅할 수 있습니다.

SDK 또는 CLI를 사용하는 IAM 사용자는 `EnableMFADevice`를 호출하여 추가 MFA 디바이스를 활성화하거나 `DeactivateMFADevice`를 호출하여 기존 MFA 디바이스를 비활성화할 수 있습니다. 이 작업을 성공적으로 수행하려면 먼저 `GetSessionToken`을 호출하고 기존 MFA 디바이스로 MFA 코드를 제출해야 합니다. 이 호출은 MFA 인증이 필요한 API 작업에 서명하는 데 사용할 수 있는 임시 보안 자격 증명을 반환합니다. 요청 및 응답의 예는 `GetSessionToken` - 신뢰할 수 없는 환경에 있는 사용자를 위한 임시 자격 증명을 참조하세요.

IAM에서 가상 디바이스 개체를 생성하여 가상 MFA 디바이스를 나타내려면

이러한 명령은 다음 명령의 많은 일련 번호 대신 사용되는 디바이스에 ARN을 제공합니다.

- AWS CLI: `aws iam create-virtual-mfa-device`
- AWS API: `CreateVirtualMFADevice`

AWS에서 사용할 목적으로 MFA 디바이스를 활성화하려면

다음 명령은 디바이스와 AWS를 동기화하여 사용자에 연결합니다. 디바이스가 가상이라면 가상 디바이스의 ARN을 일련 번호로 사용합니다.

> **Important**
>
> 인증 코드를 생성한 후 바로 요청을 제출하세요. 코드를 생성한 후 너무 오래 기다렸다 요청을 제출할 경우 MFA 디바이스가 사용자와 연결은 되지만 MFA 디바이스가 동기화되지 않습니다. 이는 시간 기반 일회용 암호(TOTP)가 잠시 후에 만료되기 때문입니다. 이 경우, 아래에서 설명하는 명령을 사용하여 디바이스를 재동기화할 수 있습니다.

- AWS CLI: `aws iam enable-mfa-device`
- AWS API: `EnableMFADevice`

디바이스를 비활성화하려면

다음 명령을 사용하여 디바이스를 사용자에게서 분리하고 비활성화합니다. 디바이스가 가상이라면 가상 디바이스의 ARN을 일련 번호로 사용합니다. 별도로 가상 디바이스 개체를 삭제해야 합니다.

- AWS CLI: `aws iam deactivate-mfa-device`
- AWS API: `DeactivateMFADevice`

가상 MFA 디바이스 개체를 표시하려면

다음 명령을 사용하여 가상 MFA 디바이스 개체의 목록을 봅니다.

- AWS CLI: `aws iam list-virtual-mfa-devices`
- AWS API: `ListVirtualMFADevices`

가상 MFA 디바이스를 태깅하려면

다음 명령을 사용하여 가상 MFA 디바이스를 태깅합니다.

- AWS CLI: `aws iam tag-mfa-device`
- AWS API: `TagMFADevice`

가상 MFA 디바이스의 태그를 나열하려면

다음 명령을 사용하여 가상 MFA 디바이스에 연결된 태그를 나열합니다.

- AWS CLI: `aws iam list-mfa-device-tags`
- AWS API: `ListMFADeviceTags`

가상 MFA 디바이스를 태깅 해제하려면

다음 명령을 사용하여 가상 MFA 디바이스에 연결된 태그를 제거합니다.

- AWS CLI: `aws iam untag-mfa-device`
- AWS API: `UntagMFADevice`

MFA 디바이스를 다시 동기화하려면

디바이스가 AWS에서 허용하지 않는 코드를 생성하는 경우 이러한 명령을 사용하세요. 디바이스가 가상이라면 가상 디바이스의 ARN을 일련 번호로 사용합니다.

- AWS CLI: `aws iam resync-mfa-device`
- AWS API: `ResyncMFADevice`

IAM에서 가상 MFA 디바이스 엔터티를 삭제하려면

디바이스가 사용자로부터 분리된 후에 디바이스 개체를 삭제할 수 있습니다.

- AWS CLI: `aws iam delete-virtual-mfa-device`
- AWS API: `DeleteVirtualMFADevice`

분실되었거나 작동하지 않는 가상 MFA 디바이스를 복구하는 방법

간혹 가상 MFA 앱이 호스팅된 사용자의 디바이스가 분실 또는 교체되었거나 작동하지 않는 경우가 있을 수 있습니다. 이러한 경우가 발생하면 사용자는 자체적으로 복구할 수 없습니다. 사용자는 관리자에게 문의하여 디바이스를 비활성화해야 합니다. 자세한 내용은 IAM에서 MFA로 보호되는 ID 복구 단원을 참조하십시오.

MFA 상태 확인

IAM 콘솔을 사용하여 AWS 계정 루트 사용자 또는 IAM 사용자가 유효한 MFA 디바이스를 활성화했는지를 확인할 수 있습니다.

루트 사용자의 MFA 상태를 확인하려면

1. 루트 사용자 자격 증명으로 AWS Management Console에 로그인한 다음 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 오른쪽 상단의 탐색 모음에서 사용자 이름을 선택한 다음 Security credentials(보안 자격 증명)를 선택합니다.
3. [멀티 팩터 인증(MFA)(Multi-factor Authentication (MFA))]에서 MFA가 활성화되었는지 비활성화되었는지 여부를 확인합니다. MFA가 활성화되어 있지 않으면 알림 기호 ( ) 가 표시됩니다.

계정에 대해 MFA를 활성화하고 싶다면 다음 중 하나를 참조하세요.

- 루트 사용자용 가상 MFA 디바이스 활성화(콘솔)
- 루트 사용자용 패스키 또는 보안 키 활성화(콘솔)

- 루트 사용자에 대해 하드웨어 TOTP 토큰 활성화(콘솔)

IAM 사용자의 MFA 상태를 확인하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 필요할 경우 다음 단계를 통해 사용자 테이블에 MFA 열을 추가합니다.
   a. 테이블 위 맨 오른쪽에서 설정 아이콘
   ( )
   을 선택합니다.
   b. 열 관리(Manage Columns)에서 MFA를 선택합니다.
   c. (선택 사항) 필요할 경우 사용자 테이블에 표시하지 않으려는 열이 있으면 해당 열의 확인란 선택을 취소하면 됩니다.
   d. 닫기를 선택하여 사용자 목록으로 돌아갑니다.
4. MFA 열에는 활성화된 MFA 디바이스가 표시됩니다. 사용자에게 활성화되어 있는 MFA 디바이스가 없으면 콘솔에 없음(None)이라고 표시됩니다. 사용자에게 활성화된 MFA 디바이스가 있으면 MFA 열에 활성화된 디바이스의 유형이 Virtual(가상), Security key(보안 키), Hardware(하드웨어) 또는 SMS 값으로 표시됩니다.

   > **Note**
   >
   > AWS는 SMS 다중 인증(MFA) 활성화에 대한 지원을 종료했습니다. SMS 문자 메시지 기반 MFA를 사용하는 IAM 사용자가 있는 고객은 가상(소프트웨어 기반) MFA 디바이스, FIDO 보안 키 또는 하드웨어 MFA 디바이스와 같은 대체 방법 중 하나로 전환하는 것이 좋습니다. 계정의 사용자 중에서 SMS MFA 디바이스가 할당된 사용자를 식별할 수 있습니다. 이렇게 하려면 IAM 콘솔로 이동하여 탐색 창에서 사용자를 선택하고 표의 MFA 열에서 SMS가 표시된 사용자를 찾습니다.

5. 사용자의 MFA 디바이스에 대한 추가 정보를 보려면 MFA 상태를 확인하려는 사용자의 이름을 선택합니다. 그런 다음 보안 자격 증명(Security credentials) 탭을 선택합니다.
6. 사용자에게 활성화되어 있는 MFA 디바이스가 없으면 콘솔에 MFA 디바이스 없음이라고 표시됩니다. 다중 인증(MFA) 섹션에서 MFA 디바이스를 할당해 AWS 환경의 보안을 개선합니다. 사용자가 MFA 디바이스를 활성화한 경우 Multi-factor authentication (MFA)(다중 인증(MFA)) 섹션에 디바이스에 대한 세부 정보가 표시됩니다.

- 디바이스 이름
- 디바이스 유형입니다.
- 물리적 디바이스의 일련 번호 또는 가상 디바이스의 AWS ARN과 같은 디바이스 식별자
- 디바이스가 생성된 시기

디바이스를 제거하거나 다시 동기화하려면 디바이스 옆에 있는 라디오 버튼을 선택하고 Remove(제거) 또는 Resync(다시 동기화)를 선택합니다.

MFA 활성화에 대한 자세한 내용은 다음을 참조하세요.

- AWS Management Console에서 가상 MFA 디바이스 할당
- AWS Management Console에서 패스키 또는 보안 키 할당
- AWS Management Console에서 하드웨어 TOTP 토큰 할당

가상 및 하드웨어 MFA 디바이스 재동기화

AWS를 사용하여 가상 및 하드웨어 멀티 팩터 인증(MFA) 디바이스를 다시 동기화할 수 있습니다. 디바이스를 사용하려고 할 때 디바이스가 동기화되지 않으면 로그인 시도가 실패하고 디바이스를 다시 동기화하라는 메시지가 IAM에 표시됩니다.

> **Note**
>
> FIDO 보안 키는 항상 동기화되어 있습니다. FIDO 보안 키를 분실했거나 도난당한 경우 비활성화할 수 있습니다. 모든 MFA 디바이스 유형의 비활성화에 대한 지침은 다른 IAM 사용자에 대해 MFA 디바이스를 비활성화하려면(콘솔) 섹션을 참조하세요.

AWS 관리자는 IAM 사용자의 가상 및 하드웨어 MFA 디바이스가 동기화 상태를 벗어난 경우 이를 재동기화할 수 있습니다.

AWS 계정 루트 사용자 MFA 디바이스가 작동하지 않는 경우 로그인 프로세스 완료 여부와 관계없이 IAM 콘솔을 사용하여 디바이스를 재동기화할 수 있습니다. 디바이스를 성공적으로 다시 동기화할 수 없는 경우 연결을 해제하고 다시 연결해야 할 수 있습니다. 이 작업을 수행하는 방법에 대한 자세한 내용은 MFA 디바이스 비활성화 및 IAM의 AWS 다중 인증 섹션을 참조하세요.

주제

- 필요한 권한
- 가상 및 하드웨어 MFA 디바이스 재동기화(IAM 콘솔)
- 가상 및 하드웨어 MFA 디바이스 재동기화(AWS CLI)
- 가상 및 하드웨어 MFA 디바이스 재동기화(AWS API)

필요한 권한

IAM 사용자의 가상 또는 하드웨어 MFA 장치를 다시 동기화하려면 다음 정책에 따른 권한이 있어야 합니다. 이 정책에서는 장치를 생성하거나 비활성화하는 것을 허용하지 않습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowListActions",
            "Effect": "Allow",
            "Action": [
                "iam:ListVirtualMFADevices"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUserToViewAndManageTheirOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "BlockAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:ListMFADevices",
                "iam:ListVirtualMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
```

```
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

가상 및 하드웨어 MFA 디바이스 재동기화(IAM 콘솔)

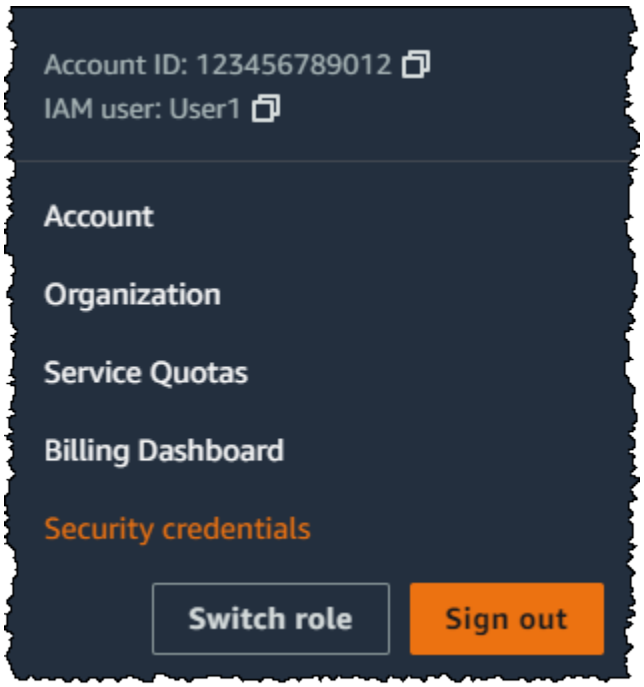
IAM 콘솔을 사용하여 가상 및 하드웨어 MFA 디바이스를 재동기화할 수 있습니다.

자신의 IAM 사용자에 대한 가상 또는 하드웨어 MFA 디바이스를 다시 동기화하려면(콘솔)

1. AWS 계정 ID나 계정 별칭, IAM 사용자 이름 및 암호를 사용하여 IAM 콘솔에 로그인합니다.

   > **Note**
   >
   > 사용자 편의를 위해 AWS 로그인 페이지는 브라우저 쿠키를 사용하여 IAM 사용자 이름 및 계정 정보를 기억합니다. 이전에 다른 사용자로 로그인한 경우 페이지 하단 근처의 다른 계정에 로그인(Sign in to a different account)을 선택하여 기본 로그인 페이지로 돌아갑니다. 여기서 AWS 계정 ID 또는 계정 별칭을 입력하면 계정의 IAM 사용자 로그인 페이지로 리디렉션됩니다.

   AWS 계정 ID를 받으려면 관리자에게 문의하세요.
2. 오른쪽 상단의 탐색 모음에서 사용자 이름을 선택한 다음 Security credentials(보안 자격 증명)를 선택합니다.



3. AWS IAM 보안 인증 탭의 다중 인증(MFA) 섹션에서 MFA 디바이스 옆의 라디오 버튼을 선택하고 다시 동기화를 선택합니다.
4. 디바이스에서 순차적으로 생성된 다음 2개의 코드를 MFA code 1(MFA 코드 1) 및 MFA code 2(MFA 코드 2)에 입력합니다. 그런 다음 Resync(다시 동기화)를 선택합니다.

> **Important**
>
> 코드를 생성한 후 즉시 요청을 제출하세요. 코드를 생성한 후 너무 오래 기다렸다 요청을 제출할 경우 요청이 처리되는 것으로 보이지만 디바이스가 동기화되지 않습니다. 이는 시간 기반 일회용 암호(TOTP)가 잠시 후에 만료되기 때문입니다.

다른 IAM 사용자에 대한 가상 및 하드웨어 MFA 디바이스를 다시 동기화하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택한 다음 MFA 디바이스를 재동기화해야 할 사용자의 이름을 선택합니다.
3. Security credentials(보안 자격 증명) 탭을 선택합니다. 다중 인증(MFA) 섹션에서 MFA 디바이스 옆의 라디오 버튼을 선택하고 다시 동기화를 선택합니다.

4. 디바이스에서 순차적으로 생성된 다음 2개의 코드를 MFA code 1(MFA 코드 1) 및 MFA code 2(MFA 코드 2)에 입력합니다. 그런 다음 Resync(다시 동기화)를 선택합니다.

> **Important**
>
> 코드를 생성한 후 즉시 요청을 제출하세요. 코드를 생성한 후 너무 오래 기다렸다 요청을 제출할 경우 요청이 처리되는 것으로 보이지만 디바이스가 동기화되지 않습니다. 이는 시간 기반 일회용 암호(TOTP)가 잠시 후에 만료되기 때문입니다.

로그인 전에 루트 사용자 MFA를 재동기화하려면(콘솔)

1. 인증 디바이스로 Amazon Web Services 로그인(Amazon Web Services Sign In With Authentication Device) 페이지에서 인증 디바이스에 문제가 있나요?(Having problems with your authentication device?)를 선택합니다. 여기를 클릭하십시오.(Click here.)

> **Note**
>
> MFA를 사용하여 로그인 및 인증 디바이스 문제 해결과 같은 다른 텍스트가 나타날 수 있습니다. 그러나 동일한 기능이 제공됩니다.

2. Re-Sync With Our Servers(서버를 통한 재동기화) 섹션에서 디바이스에서 순차적으로 생성된 다음 2개의 코드를 MFA code 1(MFA 코드 1) 및 MFA code 2(MFA 코드 2)에 입력합니다. 그런 다음 인증 디바이스 재동기화(Re-sync authentication device)를 선택합니다.
3. 필요할 경우 암호를 다시 입력하고 로그인을 선택합니다. 그런 다음 MFA 디바이스를 사용하여 로그인을 완료합니다.

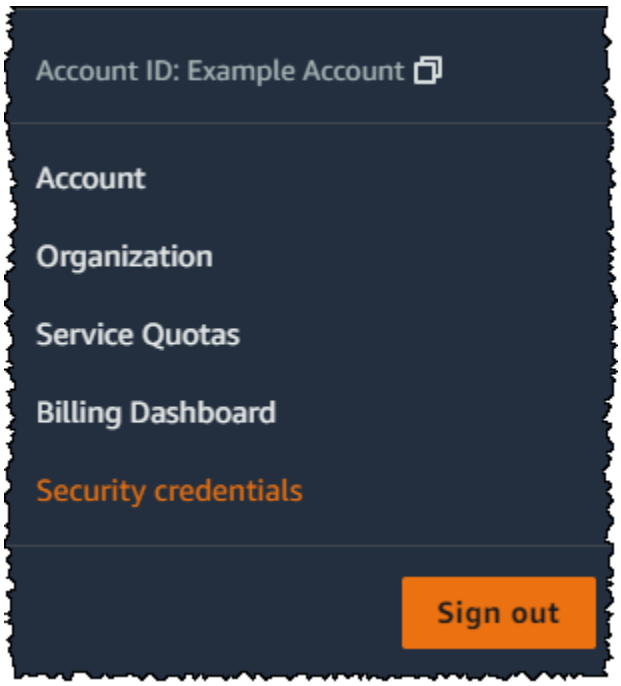
로그인 이후 루트 사용자 MFA 디바이스를 재동기화하려면(콘솔)

1. 루트 사용자를 선택하고 AWS 계정 계정 이메일 주소를 입력하여 IAM 콘솔에 계정 소유자로 로그인합니다. 다음 페이지에서 암호를 입력합니다.

> **Note**
>
> 루트 사용자는 IAM 사용자로 로그인 페이지에 로그인할 수 없습니다. IAM 사용자로 로그인 페이지가 보이면 페이지 하단에 있는 루트 사용자 이메일을 사용하여 로그인을 선택합

니다. 루트 사용자로 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 루트 사용자로 AWS Management Console 로그인을 참조하세요.

2. 탐색 모음의 오른쪽에서 계정 이름을 선택한 다음 Security credentials(보안 자격 증명)를 선택합니다. 필요한 경우 Continue to Security Credentials(보안 자격 증명으로 계속)를 선택합니다.



3. 페이지의 Multi-factor authentication (MFA)(멀티 팩터 인증(MFA)) 섹션을 확장합니다.
4. 디바이스 옆의 라디오 버튼을 선택하고 Resync(다시 동기화)를 선택합니다.
5. Resync MFA device(MFA 기기 다시 동기화) 대화 상자에서 디바이스에서 순차적으로 생성된 다음 2개의 코드를 MFA code 1(MFA 코드 1) 및 MFA code 2(MFA 코드 2)에 입력합니다. 그런 다음 Resync(다시 동기화)를 선택합니다.

> **Important**
>
> 코드를 생성한 후 즉시 요청을 제출하세요. 코드를 생성한 후 너무 오래 기다렸다 요청을 제출할 경우 MFA 디바이스가 사용자와 연결은 되지만 MFA 디바이스가 동기화되지 않습니다. 이는 시간 기반 일회용 암호(TOTP)가 잠시 후에 만료되기 때문입니다.

가상 및 하드웨어 MFA 디바이스 재동기화(AWS CLI)

AWS CLI에서 가상 및 하드웨어 MFA 디바이스를 재동기화할 수 있습니다.

IAM 사용자에 대한 가상 및 하드웨어 MFA 디바이스를 재동기화하려면(AWS CLI)

명령 프롬프트에서 aws iam resync-mfa-device 명령을 실행합니다.

- 가상 MFA 디바이스: 디바이스의 Amazon 리소스 이름(ARN)을 일련 번호로 지정합니다.

```
aws iam resync-mfa-device --user-name Richard --serial-number
 arn:aws:iam::123456789012:mfa/RichardsMFA --authentication-code1 123456 --
authentication-code2 987654
```

- 하드웨어 MFA 디바이스: 하드웨어 디바이스의 일련 번호를 일련 번호로 지정합니다. 형식은 공급업체에 따라 다릅니다. 예를 들어 Amazon에서는 gemalto 토큰을 구매할 수 있습니다. 이 토큰의 일련 번호는 일반적으로 문자 4개이며 그 뒤로 숫자 4개가 이어집니다.

```
aws iam resync-mfa-device --user-name Richard --serial-number ABCD12345678 --
authentication-code1 123456 --authentication-code2 987654
```

> **Important**
>
> 코드를 생성한 후 즉시 요청을 제출하세요. 코드를 생성한 후 너무 오래 기다렸다 요청을 제출할 경우 잠시 후 코드가 만료되기 때문에 요청이 실패합니다.

가상 및 하드웨어 MFA 디바이스 재동기화(AWS API)

IAM에는 동기화를 수행하는 API 호출이 있습니다. 이러한 경우 가상 및 하드웨어 MFA 사용자에게 API 호출에 액세스할 수 있는 권한을 부여하는 것이 좋습니다. 이때 사용자가 필요할 때마다 디바이스를 재동기화할 수 있도록 API 호출 기반 도구를 구축해야 합니다.

IAM 사용자에 대한 가상 및 하드웨어 MFA 디바이스를 재동기화하려면(AWS API)

- ResyncMFADevice 요청을 보냅니다.

MFA 디바이스 비활성화

다중 인증(MFA) 디바이스를 사용하여 IAM 사용자로 로그인하는 데 문제가 있는 경우 관리자에게 문의하여 도움을 받으세요.

관리자는 다른 IAM 사용자에 대해 디바이스를 비활성화할 수 있습니다. 이 방법을 사용하면 MFA를 사용하지 않고 로그인할 수 있습니다. MFA 디바이스가 교체되는 중이거나 디바이스가 일시적으로 사용 불가능할 때 이 방법을 임시 해결 방법으로 사용할 수 있습니다. 그러나 최대한 빨리 사용자를 위한 새 디바이스를 활성화하는 것이 좋습니다. 새 MFA 디바이스를 활성화 하는 방법에 대한 자세한 내용은 IAM의 AWS 다중 인증를 참조하세요.

> **Note**
>
> API 또는 AWS CLI를 사용하여 AWS 계정에서 사용자를 삭제하는 경우 사용자의 MFA 디바이스를 비활성화 또는 삭제해야 합니다. 이 변경 사항을 사용자 제거 과정의 일부로 활용합니다. 사용자 삭제에 대한 자세한 내용은 IAM 사용자 삭제 또는 비활성화 섹션을 참조하세요.

주제

- MFA 디바이스 비활성화(콘솔)
- MFA 디바이스 비활성화(AWS CLI)
- MFA 디바이스 비활성화(AWS API)

MFA 디바이스 비활성화(콘솔)

다른 IAM 사용자에 대해 MFA 디바이스를 비활성화하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 사용자의 MFA 디바이스를 비활성화하려면 MFA를 제거하려는 사용자의 이름을 선택합니다.
4. Security credentials(보안 자격 증명) 탭을 선택합니다.
5. 다중 인증(MFA)에서 MFA 디바이스 옆의 라디오 버튼을 선택하고 제거를 선택하고 제거를 선택합니다.

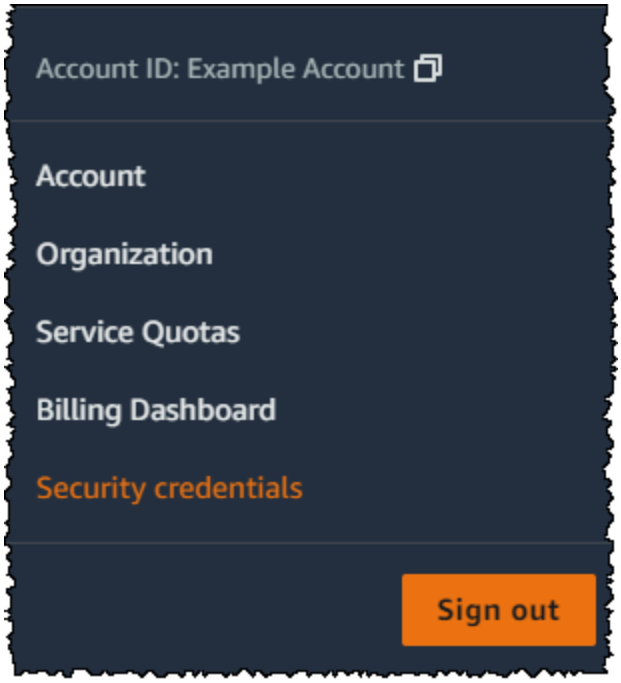
   디바이스가 AWS에서 제거됩니다. 디바이스는 다시 활성화되어 AWS 사용자 또는 AWS 계정 루트 사용자에 연결될 때까지는 로그인 또는 요청 인증에 사용할 수 없습니다.

AWS 계정 루트 사용자의 MFA 디바이스를 비활성화하려면(콘솔)

1. 루트 사용자를 선택하고 AWS 계정 계정 이메일 주소를 입력하여 IAM 콘솔에 계정 소유자로 로그인합니다. 다음 페이지에서 암호를 입력합니다.

   > **Note**
   >
   > 루트 사용자는 IAM 사용자로 로그인 페이지에 로그인할 수 없습니다. IAM 사용자로 로그인 페이지가 보이면 페이지 하단에 있는 루트 사용자 이메일을 사용하여 로그인을 선택합니다. 루트 사용자로 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 루트 사용자로 AWS Management Console 로그인을 참조하세요.

2. 탐색 모음의 오른쪽에서 계정 이름을 선택한 다음 Security credentials(보안 자격 증명)를 선택합니다. 필요한 경우 Continue to Security Credentials(보안 자격 증명으로 계속)를 선택합니다.



3. Multi-factor authentication (MFA)(다중 인증(MFA)) 섹션에서 비활성화하려는 MFA 디바이스 옆의 라디오 버튼을 선택하고 Remove(제거)를 선택합니다.
4. 제거를 선택합니다.

   AWS 계정의 MFA 디바이스가 비활성화됩니다. AWS 계정과 연결된 이메일로 Amazon Web Services의 확인 메시지가 왔는지 점검합니다. 이메일이 Amazon Web Services의 Multi-Factor Authentication(MFA)이 비활성화되었음을 알립니다. 메시지는 `@amazon.com` 또는 `@aws.amazon.com`에서 전송되어 옵니다.

MFA 디바이스 비활성화(AWS CLI)

IAM 사용자에 대해 MFA 디바이스를 비활성화하려면(AWS CLI)

- 다음 명령을 실행합니다. `aws iam deactivate-mfa-device`

MFA 디바이스 비활성화(AWS API)

IAM 사용자에 대해 MFA 디바이스를 비활성화하려면(AWS API)

- 다음 연산을 호출합니다. `DeactivateMFADevice`

IAM에서 MFA로 보호되는 ID 복구

가상 MFA 디바이스 또는 하드웨어 TOTP 토큰이 정상적으로 작동하는 것처럼 같지만, 이것을 사용하여 AWS 리소스에 액세스하지 못하는 경우에는 AWS와 동기화되지 않는 것이 원인일 수 있습니다. 가상 MFA 디바이스 또는 하드웨어 MFA 디바이스의 동기화에 대한 자세한 내용은 가상 및 하드웨어 MFA 디바이스 재동기화 섹션을 참조하세요. FIDO 보안 키는 항상 동기화되어 있습니다.

AWS 계정 루트 사용자용 MFA 디바이스가 분실 또는 손상되거나 작동하지 않는 경우, 계정에 대한 액세스를 복구할 수 있습니다. IAM 사용자는 관리자에게 문의하여 디바이스를 비활성화해야 합니다.

> **Important**
>
> 여러 개의 MFA 디바이스를 활성화하는 것이 좋습니다. 여러 개의 MFA 디바이스를 등록하면 디바이스의 분실 또는 고장이 발생한 경우에도 계속 액세스할 수 있습니다. AWS 계정 루트 사용자 및 IAM 사용자는 어떤 유형이든 최대 8개의 MFA 디바이스를 등록할 수 있습니다.

루트 사용자 MFA 디바이스 복구

AWS 계정 루트 사용자 다중 인증(MFA) 디바이스가 분실되었거나 손상되었거나 작동하지 않는 경우 동일한 AWS 계정 루트 사용자에게 등록된 다른 MFA 디바이스를 사용하여 로그인할 수 있습니다. 루트 사용자가 MFA 디바이스를 하나만 활성화한 경우 다른 인증 방법을 사용할 수 있습니다. 다시 말해, MFA 디바이스로 로그인할 수 없는 경우에 사용자 계정으로 등록된 이메일 및 기본 연락처 전화번호로 사용자 아이덴티티를 확인하여 로그인할 수 있습니다.

대체 인증 요소를 사용하여 루트 사용자로 로그인하기 전에 계정과 연결된 이메일 및 기본 연락처 전화번호에 액세스할 수 있는지 확인합니다. 기본 연락처 전화번호를 업데이트해야 하는 경우 루트 사용자 대신 관리자 액세스 권한이 모두 있는 IAM 사용자로 로그인합니다. 계정 연락처 정보 업데이트에 대한

추가 지침은 AWS Billing 사용설명서의 연락처 정보 편집을 참조하십시오. 이메일 및 기본 연락처 전화번호에 액세스할 수 없는 경우 AWS Support에 문의해야 합니다.

> **Important**
>
> 성공적인 계정 복구를 위해 루트 사용자에게 연결된 이메일 주소와 연락처 전화번호를 최신 상태로 유지하는 것이 좋습니다. 자세한 내용은 AWS Account Management참조 안내서에 있는 AWS 계정의 기본 연락처 업데이트를 참조하세요.

AWS 계정 루트 사용자로 다른 인증 요소를 사용하여 로그인하려면

1. 루트 사용자를 선택하고 AWS 계정이메일 주소를 입력하여 AWS Management Console에 계정 소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.
2. 추가 확인 필요(Additional verification required) 페이지에서 인증할 MFA 방법을 선택하고 다음(Next)을 선택합니다.

   > **Note**
   >
   > MFA를 사용하여 로그인(Sign in using MFA), 인증 디바이스 문제 해결(Troubleshoot your authentication device), MFA 문제 해결(Troubleshoot MFA)과 같은 대체 텍스트가 표시될 수 있지만 기능은 동일합니다. 대체 인증 요소를 사용하여 계정 이메일 주소와 기본 연락처 전화번호를 확인할 수 없는 경우 AWS Support에 문의하여 MFA 디바이스를 비활성하세요.

3. 사용 중인 MFA 유형에 따라 다른 페이지가 표시되지만 MFA 문제 해결(Troubleshoot MFA) 옵션은 동일하게 작동합니다. 추가 확인 필요(Additional verification required) 페이지 또는 복수 단계 인증(Multi-factor authentication) 페이지에서 MFA 문제 해결(Troubleshoot MFA)을 선택합니다.
4. 필요할 경우 암호를 다시 입력하고 로그인을 선택합니다.
5. 디바이스 인증 문제 해결(Troubleshoot your authentication device) 페이지에 있는 대체 인증 팩터를 사용하여 로그인(Sign In Using Alternative Factors of Authentication) 섹션에서 대체 팩터를 사용하여 로그인(Sign in using alternative factors)을 선택합니다.
6. 대체 인증 팩터를 사용하여 로그인(Sign In Using Alternative Factors of Authentication) 페이지에서 이메일 주소를 확인하여 계정을 인증하고 확인 이메일 보내기(Send verification email)를 선택합니다.
7. AWS 계정과 연결된 이메일에서 Amazon Web Services(recover-mfa-no-reply@verify.signin.aws)의 메시지를 확인합니다. 이메일 지침을 따릅니다.

계정에 이메일이 없는 경우에는 스팸 폴더를 확인하거나 브라우저로 돌아가 이메일 재전송(Resend the email)을 선택합니다.

8. 이메일 주소를 확인한 후에 계정 인증을 계속 진행할 수 있습니다. 기본 연락처 전화번호를 확인하려면 지금 전화하기(Call me now)를 선택합니다.
9. AWS 전화를 받고, 요구에 따라 AWS 웹사이트의 6자리 숫자를 전화 키패드에 입력합니다.

   AWS에서 전화가 오지 않을 경우에는 로그인을 선택하여 콘솔에 다시 로그인하고 처음부터 다시 시작합니다. 또는 분실하거나 사용할 수 없는 다중 인증(MFA) 디바이스(Lost or unusable Multi-Factor Authentication (MFA) device)를 참조하여 지원팀에 문의하세요.
10. 전화 번호를 확인한 후에는 콘솔에 로그인(Sign in to the console)을 선택하여 계정에 로그인할 수 있습니다.
11. 다음 단계는 사용 중인 MFA의 유형에 따라 다릅니다.

    - 가상 MFA 디바이스의 경우, 디바이스에서 계정을 제거합니다. 그런 다음 AWS 보안 자격 증명 페이지로 이동하여 기존 MFA 가상 디바이스 엔터티를 삭제한 다음 새 엔터티를 생성합니다.
    - FIDO 보안 키의 경우, AWS 보안 인증 페이지로 이동하여 기존 FIDO 보안 키를 비활성화한 다음 새 키를 활성화합니다.
    - 하드웨어 TOTP 토큰의 경우, 타사 제공업체에 연락해 디바이스 수리 또는 교체를 위한 도움을 받으세요. 새 디바이스를 받기 전까지 다른 인증 요소를 사용하여 계속 로그인할 수 있습니다. 하드웨어 MFA 디바이스를 새로 받은 후에는 AWS 보안 자격 증명 페이지로 이동하여 기존 MFA 디바이스를 삭제합니다.

> **Note**
>
> 잃어버렸거나 도난당한 MFA 디바이스를 동일한 유형의 디바이스로 대체해야 하는 것은 아닙니다. 예를 들어, FIDO 보안 키가 망가져 새로 주문한 경우, 새로운 FIDO 보안 키를 받을 때까지는 가상 MFA 또는 하드웨어 TOTP 토큰을 사용할 수 있습니다.

> **Important**
>
> MFA 디바이스가 분실 또는 도난된 경우 로그인하고 대체 MFA 디바이스를 설정한 후 루트 사용자 암호를 변경하세요. 인증 디바이스를 훔친 공격자가 사용자의 현재 암호를 알고 있을 수 있습니다. 자세한 내용은 AWS 계정 루트 사용자의 암호 변경 단원을 참조하십시오.

IAM 사용자 MFA 디바이스 복구

디바이스가 분실 또는 작동 중지된 경우 IAM 사용자는 직접 복구할 수 없습니다. 해당 사용자는 관리자에 문의하여 디바이스를 비활성화해야 합니다. 그러면 새 디바이스를 활성화할 수 있습니다.

IAM 사용자로 MFA 디바이스에 관한 도움을 받으려면

1. AWS 관리자나 그 밖에 IAM 사용자의 사용자 이름 및 암호를 제공한 담당자에게 문의합니다. MFA 디바이스 비활성화 설명대로 관리자가 MFA 디바이스를 비활성화해야 로그인할 수 있습니다.
2. 다음 단계는 사용 중인 MFA의 유형에 따라 다릅니다.

   - 가상 MFA 디바이스의 경우, 디바이스에서 계정을 제거합니다. 그런 다음 AWS Management Console에서 가상 MFA 디바이스 할당 설명대로 가상 디바이스를 활성화합니다.
   - FIDO 보안 키의 경우, 타사 공급업체에 연락해 디바이스 교체를 위한 도움을 받으세요. 새로운 FIDO 보안 키를 받은 경우, AWS Management Console에서 패스키 또는 보안 키 할당에 설명된 대로 활성화합니다.
   - 하드웨어 TOTP 토큰의 경우, 타사 제공업체에 연락해 디바이스 수리 또는 교체를 위한 도움을 받으세요. 물리적 MFA 디바이스를 새로 받은 후에는 AWS Management Console에서 하드웨어 TOTP 토큰 할당 설명대로 디바이스를 활성화합니다.

   > **Note**
   >
   > 잃어버렸거나 도난당한 MFA 디바이스를 동일한 유형의 디바이스로 대체해야 하는 것은 아닙니다. 어떤 조합이든 최대 8개의 MFA 디바이스를 보유할 수 있습니다. 예를 들어, FIDO 보안 키가 망가져 새로 주문한 경우, 새로운 FIDO 보안 키를 받을 때까지는 가상 MFA 또는 하드웨어 TOTP 토큰을 사용할 수 있습니다.

3. MFA 디바이스가 없거나 도난당한 경우에는 인증 디바이스를 훔친 공격자가 현재 암호를 알 수 있으므로 비밀번호도 변경하세요. 자세한 내용은 IAM 사용자 암호 관리 섹션을 참조하세요.

MFA를 통한 보안 API 액세스

사용자가 호출할 수 있는 API 작업을 IAM 정책을 사용해 지정할 수 있습니다. 민감한 작업을 수행할 수 있도록 허용하기 전에 사용자가 다중 인증(MFA)으로 인증하도록 요구하는 추가 보안이 필요할 수 있습니다.

예를 들어 사용자가 Amazon EC2 `RunInstances`, `DescribeInstances` 및 `StopInstances` 작업을 수행하도록 허용하는 정책이 있을 수 있습니다. 하지만 `TerminateInstances`처럼 안전하지 않은 작업의 경우 이를 제한해 사용자가 AWS MFA 디바이스에서 인증할 때만 작업을 수행하도록 해야 할 필요가 있을 수 있습니다.

주제

개요

API 작업에 MFA 보호를 추가하려면 다음과 같은 작업이 필요합니다.

1. 관리자는 MFA 인증이 필요한 API 요청을 해야 하는 각 사용자에 대해 AWS MFA 디바이스를 구성합니다. 자세한 내용은 IAM의 AWS 다중 인증 단원을 참조하십시오.
2. 관리자는 사용자가 AWS MFA 디바이스로 인증했는지 여부를 확인하는 `Condition` 요소가 포함된 사용자 정책을 생성합니다.
3. 사용자는 MFA 파라미터를 지원하는 AWS STS API 작업인 AssumeRole 또는 GetSessionToken 중 하나를 호출합니다. 사용자는 사용자와 연결된 디바이스의 디바이스 식별자를 호출에 포함시킵니다. 또한 사용자는 디바이스에 생성하는 시간 기반 일회용 암호(TOTP)도 포함시킵니다. 각각의 경우, 사용자는 AWS에 추가 요청하는 데 사용하기 위해 임시 보안 자격 증명을 다시 가져옵니다.

> **Note**
>
> 서비스의 API 작업에 대한 MFA 보호 기능은 해당 서비스에서 임시 보안 자격 증명을 지원하는 경우에만 사용 가능합니다. 이러한 서비스 목록은 임시 보안 자격 증명을 사용하여 AWS에 액세스를 참조하세요.

권한 부여에 실패한 경우 AWS는 "액세스가 거부되었습니다."라는 오류 메시지를 반환합니다(무단 액세스의 경우와 동일). MFA 보호 API 정책이 적용되는 경우, 사용자가 유효한 MFA 인증 없이 API 작업을 호출하려 하면 AWS에서는 정책에 지정된 API 작업에 대한 액세스를 거부합니다. API 작업 요청의 타임스탬프가 정책에 지정된 허용 범위를 벗어난 경우에도 작업이 거부됩니다. 사용자는 MFA 코드와 디바이스 일련 번호로 새 임시 보안 자격 증명을 요청하여 MFA 인증을 다시 해야 합니다.

MFA 조건이 포함된 IAM 정책

MFA 조건이 포함된 정책은 다음에 연결할 수 있습니다.

- IAM 사용자 또는 그룹
- Amazon S3 버킷, Amazon SQS 대기열, Amazon SNS 주제 등과 같은 리소스
- 사용자가 수임할 수 있는 IAM 역할의 신뢰 정책

정책의 MFA 조건을 사용해 다음과 같은 속성을 확인할 수 있습니다.

- 존재 - 사용자가 MFA로 인증했는지 간단히 확인하려면 `Bool` 조건에서 `aws:MultiFactorAuthPresent` 키가 `True`인지 확인합니다. 사용자가 단기 자격 증명으로 인증하는 경우에만 키가 있습니다. 액세스 키와 같은 장기 자격 증명에는 이 키가 포함되어 있지 않습니다.
- 기간 - MFA 인증 이후 지정된 시간 내에서만 액세스 권한을 부여하고 싶은 경우, 숫자 조건 유형을 사용하여 `aws:MultiFactorAuthAge` 키의 기간과 값(예: 3600초)을 비교합니다. MFA가 사용되지 않는 경우 `aws:MultiFactorAuthAge` 키가 없습니다.

다음 예는 MFA 인증의 존재를 테스트하는 MFA 조건을 포함하는 IAM 역할의 신뢰 정책을 보여줍니다. 이 정책을 통해 `Principal` 요소(`ACCOUNT-B-ID`를 유효한 AWS 계정 ID로 대체)에 지정된 AWS 계정의 사용자는 이 정책이 연결된 역할을 수임할 수 있습니다. 그러나 이러한 사용자는 MFA를 사용하여 인증을 받은 경우에만 역할을 수임할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "ACCOUNT-B-ID"},
    "Action": "sts:AssumeRole",
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }
}
```

MFA의 조건 유형에 대한 자세한 내용은 AWS 글로벌 조건 컨텍스트 키, 숫자 조건 연산자 및 조건 키의 존재를 확인하는 조건 연산자 섹션을 참조하세요.

GetSessionToken과 AssumeRole 중에서 선택

AWS STS에서는 사용자가 MFA 정보를 전달할 수 있도록 `GetSessionToken`과 `AssumeRole`이라는 두 가지 API 작업을 제공합니다. 사용자가 임시 보안 자격 증명을 가져오기 위해 호출하는 API 작업은 다음 시나리오 중 어떤 것이 적용되느냐에 따라 달라집니다.

다음 시나리오에는 **`GetSessionToken`**을 사용합니다.

- 요청을 수행하는 IAM 사용자와 동일한 AWS 계정의 리소스에 액세스하는 API 작업을 호출합니다. `GetSessionToken` 요청의 임시 자격 증명은 작업 증명 요청에 MFA 정보를 포함하는 경우에만 IAM 및 AWS STS API 작업에 액세스할 수 있습니다. `GetSessionToken`에서 반환하는 임시 자격 증명에 MFA 정보가 포함되어 있으므로 자격 증명에서 수행하는 개별 API 작업에서 MFA를 확인할 수 있습니다.
- MFA 조건이 포함된 리소스 기반 정책으로 보호되는 리소스에 액세스.

`GetSessionToken` 작업의 목적은 MFA를 사용하는 사용자를 인증하는 것입니다. 정책을 사용하여 인증 작업을 제어할 수는 없습니다.

다음 시나리오에는 **`AssumeRole`**을 사용합니다.

- 같은 또는 다른 AWS 계정의 리소스에 액세스하는 API 작업을 호출합니다. API 호출은 모든 IAM 또는 AWS STS API를 포함할 수 있습니다. 액세스를 보호하기 위해 사용자가 역할을 수임하는 시각에 MFA를 적용한다는 것에 유의하세요. `AssumeRole`에서 반환하는 임시 자격 증명은 컨텍스트에 MFA 정보를 포함하고 있지 않으므로 MFA에 대한 개별 API 작업을 확인할 수 없습니다. 이것이 바로 `GetSessionToken`을 사용해 리소스 기반 정책에 의해 보호되는 리소스에 대한 액세스를 제한해야 하는 이유입니다.

이러한 시나리오가 구현되는 방식에 대한 세부 정보는 이 문서의 후반부에 나와 있습니다.

MFA 보호 API 액세스에 대한 중요 사항

API 작업에 대한 MFA 보호가 지닌 다음과 같은 측면을 이해하는 것이 중요합니다.

- MFA 보호는 임시 보안 자격 증명을 사용하는 경우에만 제공되며, 임시 보안 자격 증명은 `AssumeRole` 또는 `GetSessionToken`을 사용해 얻어야 합니다.
- AWS 계정 루트 사용자 자격 증명으로는 MFA 보호 API 액세스를 사용할 수 없습니다.
- U2F 보안 키로는 MFA 보호 API 액세스를 사용할 수 없습니다.

- 페더레이션 사용자는 AWS 서비스에 사용할 MFA 디바이스를 할당받을 수 없으므로, MFA에서 제어하는 AWS 리소스에 액세스할 수 없습니다. (다음 참조.)
- 임시 자격 증명을 반환하는 다른 AWS STS API 작업에서는 MFA를 지원하지 않습니다. `AssumeRoleWithWebIdentity` 및 `AssumeRoleWithSAML`의 경우 사용자는 외부 공급자에 의해 인증되며 AWS에서는 그 공급자가 MFA를 요구했는지를 확인할 수 없습니다. `GetFederationToken`의 경우 MFA가 특정 사용자와 반드시 연결되는 것은 아닙니다.
- 이와 마찬가지로 장기 자격 증명(IAM 사용자 액세스 키 및 루트 사용자 액세스 키)은 만료되지 않기 때문에 MFA 보호 API 액세스를 통해 사용할 수 없습니다.
- 또한, `AssumeRole` 및 `GetSessionToken`은 MFA 정보 없이도 호출할 수 있습니다. 이 경우 호출자는 임시 보안 자격 증명을 다시 가져오지만, 그러한 임시 자격 증명의 세션 정보에는 사용자가 MFA로 인증했는지 나타나지 않습니다.
- API 작업에 대해 MFA 보호를 설정하려면 정책에 MFA 조건을 추가하면 됩니다. MFA 사용을 적용하기 위해 정책에는 `aws:MultiFactorAuthPresent` 조건 키가 포함되어 있어야 합니다. 크로스 계정 위임을 위해 역할의 신뢰 정책에는 조건 키가 포함되어 있어야 합니다.
- 다른 AWS 계정이 내 계정의 리소스에 액세스하도록 허용하는 경우, 리소스의 보안은 신뢰할 수 있는 계정, 즉 다른 계정(내 계정이 아님)의 구성에 따라 달라집니다. 이것은 멀티 팩터 인증이 필요할 때도 마찬가지입니다. 가상 MFA 디바이스를 생성할 권한이 있는 신뢰할 수 있는 계정 내의 어떤 자격 증명도 MFA 클레임을 생성하여 역할의 신뢰 정책의 해당 부분을 충족할 수 있습니다. 멀티 팩터 인증을 요구하는 AWS 리소스에 대한 액세스를 다른 계정의 멤버에게 허용하기 전에 신뢰할 수 있는 계정의 소유자가 보안 모범 사례를 따르도록 해야 합니다. 예를 들어 신뢰할 수 있는 계정에서는 MFA 디바이스 관리 API 작업과 같은 중요 API 작업에 대한 액세스를 신뢰할 수 있는 특정 자격 증명으로 제한해야 합니다.
- 정책에 MFA 조건이 포함된 경우, 사용자가 MFA에 인증되지 않거나 잘못된 MFA 디바이스 식별자 또는 잘못된 TOTP를 제공하는 경우 요청이 거부됩니다.

시나리오: 크로스 계정 위임에 대한 MFA 보호

이 시나리오에서는 다른 계정의 IAM 사용자에게 액세스 권한을 위임하려고 합니다. 단 해당 사용자가 AWS MFA 디바이스로 인증된 경우에 한합니다. (크로스 계정 위임에 대한 자세한 내용은 역할 용어 및 개념 섹션을 참조하세요.)

계정 A(액세스할 리소스를 소유한 신뢰하는 계정)에 관리자 권한이 있는 IAM 사용자 Anaya가 있다고 가정해 봅시다. 그녀는 계정 B(신뢰할 수 있는 계정)의 사용자 Richard에게 액세스 권한을 부여하고 싶지만, Richard가 MFA에 인증되었는지 확인한 후에 그가 역할을 수임하기를 원합니다.

1. 신뢰하는 계정 A에서 Anaya는 `CrossAccountRole`이라는 IAM 역할을 생성하고 해당 역할의 신뢰 정책에서 보안 주체를 계정 B의 계정 ID로 설정합니다. 이 신뢰 정책은 AWS STS `AssumeRole` 작업에 대한 권한을 부여합니다. 또한 Anaya는 다음 예와 같이 신뢰 정책에 MFA 조건을 추가합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "ACCOUNT-B-ID"},
    "Action": "sts:AssumeRole",
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }
}
```

2. Anaya는 역할이 수행할 수 있는 작업을 지정하는 역할에 권한 정책을 추가합니다. MFA 보호 기능이 포함된 역할의 권한 정책은 다른 역할 권한 정책과 다르지 않습니다. 다음 예제에서는 Anaya가 역할에 추가하는 정책을 보여줍니다. 역할을 수임한 사용자는 이 정책을 통해 계정 A의 `Books` 테이블에서 Amazon DynamoDB 작업을 수행할 수 있고, 아울러 콘솔에서 작업을 수행할 때 필요한 `dynamodb:ListTables` 작업을 할 수 있습니다.

> **Note**
>
> 권한 정책은 MFA 조건을 포함하지 않습니다. MFA 인증은 사용자가 역할을 수임할 수 있는지 여부를 결정하는 데에만 사용된다는 점을 알아두세요. 사용자가 역할을 수임하면 MFA 검사가 추가로 수행되지 않습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TableActions",
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:*:ACCOUNT-A-ID:table/Books"
        },
        {
            "Sid": "ListTables",
            "Effect": "Allow",
            "Action": "dynamodb:ListTables",
```

```
            "Resource": "*"
        }
    ]
}
```

3. 신뢰할 수 있는 계정 B에서 관리자는 IAM 사용자 Richard가 AWS MFA 디바이스로 구성되었는지, 그리고 그가 이 디바이스의 ID를 알고 있는지 확인합니다. 디바이스 ID란 하드웨어 MFA 디바이스의 경우 일련 번호이며, 가상 MFA 디바이스의 경우 해당 디바이스의 ARN입니다.
4. 계정 B에서 관리자는 사용자 Richard(또는 그가 소속된 그룹)에게 `AssumeRole` 작업을 호출할 수 있도록 허용하는 다음과 같은 정책을 연결합니다. 리소스는 Anaya가 1단계에서 생성한 역할의 ARN으로 설정됩니다. 이 정책에는 MFA 조건이 포함되어 있지 않습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sts:AssumeRole"],
    "Resource": ["arn:aws:iam::ACCOUNT-A-ID:role/CrossAccountRole"]
  }]
}
```

5. 계정 B에서 Richard(또는 Richard가 실행하는 애플리케이션)는 `AssumeRole`을 호출합니다. API 호출에는 위임할 역할의 ARN(`arn:aws:iam::ACCOUNT-A-ID:role/CrossAccountRole`), MFA 디바이스의 ID 및 Richard가 자신의 디바이스에서 가져오는 현재 TOTP가 포함되어 있습니다.

   Richard가 `AssumeRole`을 호출하면, AWS에서 그가 MFA에 대한 요건을 포함해 유효한 자격 증명을 갖고 있는지 여부를 확인합니다. 만일 Richard가 유효한 자격 증명을 갖고 있다면 성공적으로 역할을 수임해 역할의 임시 자격 증명을 사용함과 동시에 계정 A에서 `Books`라는 테이블에 대해 어떤 DynamoDB 작업도 수행할 수 있습니다.

   `AssumeRole`을 호출하는 프로그램의 예는 MFA 인증이 포함된 AssumeRole 호출 섹션을 참조하세요.

시나리오: 현재 계정의 API 작업에 대한 액세스를 위한 MFA 보호

이 시나리오에서는 AWS 계정의 사용자가 AWS MFA 디바이스를 사용해 인증받은 경우에만 중요한 API 작업에 액세스할 수 있는지 확인해야 합니다.

계정 A에 EC2 인스턴스로 작업해야 하는 개발자 그룹이 있다고 가정해 봅시다. 일반적인 개발자들은 이 인스턴스를 사용할 수 있지만, `ec2:StopInstances` 또는 `ec2:TerminateInstances` 작업에

대한 권한은 없습니다. 그와 같은 "안전하지 않은" 권한이 있는 작업을 일부 신뢰할 수 있는 사용자만 액세스할 수 있게 제한하고자 하여, 이러한 민감한 Amazon EC2 작업을 허용하는 정책에 MFA 보호를 추가합니다.

이 시나리오에서 신뢰할 수 있는 사용자 중 한 명은 사용자 Sofía입니다. 사용자 Anaya는 계정 A의 관리자입니다.

1. Anaya는 Sofia가 AWS MFA 디바이스로 구성되었는지, 그리고 Sofia가 이 디바이스의 ID를 알고 있는지 확인합니다. 디바이스 ID란 하드웨어 MFA 디바이스의 경우 일련 번호이며, 가상 MFA 디바이스의 경우 해당 디바이스의 ARN입니다.
2. Anaya는 `EC2-Admins`라는 그룹을 생성하고 이 그룹에 사용자 Sofía를 추가합니다.
3. Anaya는 `EC2-Admins` 그룹에 다음과 같은 정책을 연결합니다. 이 정책은 사용자에게 Amazon EC2 `StopInstances` 및 `TerminateInstances` 작업을 호출할 권한을 부여하는데, 단 이 사용자가 MFA를 사용하여 인증되었을 경우에 한합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "ec2:StopInstances",
      "ec2:TerminateInstances"
    ],
    "Resource": ["*"],
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }]
}
```

4.
> **Note**
>
> 이 정책의 효력이 발생하려면 사용자는 먼저 로그아웃한 한 후 다시 로그인해야 합니다.

   사용자 Sofía가 Amazon EC2 인스턴스를 중지하거나 종료해야 하는 경우, Sofia(또는 Sofia가 실행하는 애플리케이션)는 `GetSessionToken`을 호출합니다. 이 API 작업에서는 MFA 디바이스의 ID와 Sofia가 자신의 디바이스에서 가져오는 현재 TOTP를 전달합니다.
5. 사용자 Sofía(또는 Sofía가 사용하는 애플리케이션)는 `GetSessionToken`에서 제공하는 임시 자격 증명을 사용하여 Amazon EC2 `StopInstances` 또는 `TerminateInstances` 작업을 호출합니다.

`GetSessionToken`을 호출하는 프로그램의 예는 이 문서의 후반부에 있는 MFA 인증이 포함된 GetSessionToken 호출 섹션을 참조하세요.

시나리오: 리소스 기반 정책이 있는 리소스에 대한 MFA 보호

이 시나리오에서는 S3 버킷, SQS 대기열 또는 SNS 주제의 소유자입니다. 리소스에 액세스하는 모든 AWS 계정 사용자가 AWS MFA 디바이스로 인증되었는지 확인하려고 합니다.

이 시나리오는 사용자가 역할을 먼저 수임하지 않고도 크로스 계정 MFA 보호를 제공하는 방법을 설명합니다. 이 경우 사용자는 세 가지 조건이 충족되면 리소스에 액세스할 수 있습니다. 즉 사용자는 MFA로 인증을 받아야 하고, `GetSessionToken`에서 임시 보안 자격 증명을 가져올 수 있어야 하며, 리소스의 정책에서 신뢰하는 계정에 로그인해 있어야 합니다.

계정 A에 속해 있고 S3 버킷을 생성한다고 가정해 봅시다. 여러 AWS 계정에 속한 사용자에게 이 버킷에 대한 액세스를 부여하되, 사용자가 MFA로 인증한 경우에 한하고자 합니다.

이 시나리오에서 사용자 Anaya는 계정 A의 관리자입니다. 사용자 Nikhil은 계정 C의 IAM 사용자입니다.

1. 계정 A에서 Anaya는 `Account-A-bucket`이라는 버킷을 생성합니다.
2. Anaya는 이 버킷에 버킷 정책을 추가합니다. 이 정책은 계정 A, 계정 B 또는 계정 C의 모든 사용자가 이 버킷에서 Amazon S3 `PutObject` 및 `DeleteObject` 작업을 수행하도록 허용합니다. 이 정책에는 MFA 조건이 포함되어 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"AWS": [
      "ACCOUNT-A-ID",
      "ACCOUNT-B-ID",
      "ACCOUNT-C-ID"
    ]},
    "Action": [
      "s3:PutObject",
      "s3:DeleteObject"
    ],
    "Resource": ["arn:aws:s3:::ACCOUNT-A-BUCKET-NAME/*"],
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }]
```

```
}
```

> **Note**
>
> Amazon S3는 루트 계정 액세스에 대해(서만) MFA Delete 기능을 제공합니다. 버킷의 버전 관리 상태를 설정할 때 Amazon S3 MFA Delete를 활성화할 수 있습니다. Amazon S3 MFA Delete는 IAM 사용자에 적용할 수 없으며 MFA 보호 API 액세스와 별개로 관리됩니다. 버킷을 삭제할 권한이 있는 IAM 사용자도 Amazon S3 MFA Delete가 활성화된 버킷은 삭제할 수 없습니다. Amazon S3 MFA Delete에 대한 자세한 내용은 MFA Delete를 참조하세요.

3. 계정 C에서 관리자는 사용자 Nikhil이 AWS MFA 디바이스로 구성되어 있고 해당 디바이스의 ID를 알고 있는지 확인합니다. 디바이스 ID란 하드웨어 MFA 디바이스의 경우 일련 번호이며, 가상 MFA 디바이스의 경우 해당 디바이스의 ARN입니다.
4. 계정 C에서 Nikhil(또는 그가 실행하는 애플리케이션)은 `GetSessionToken`을 호출합니다. 이 호출에는 MFA 디바이스의 ID 또는 ARN과 Nikhil이 자신의 디바이스에서 가져오는 현재 TOTP가 포함되어 있습니다.
5. Nikhil(또는 그가 사용하는 애플리케이션)은 `GetSessionToken`에서 반환하는 임시 자격 증명을 사용하여 `Account-A-bucket`으로 파일을 업로드하는 Amazon S3 `PutObject` 작업을 호출합니다.

   `GetSessionToken`을 호출하는 프로그램의 예는 이 문서의 후반부에 있는 MFA 인증이 포함된 GetSessionToken 호출 섹션을 참조하세요.

> **Note**
>
> `AssumeRole`이 반환하는 임시 자격 증명은 이 경우에는 유효하지 않습니다. 사용자는 역할 수임을 위해 MFA 정보를 제공할 수 있지만 `AssumeRole`에서 반환하는 임시 자격 증명에는 MFA 정보가 포함되어 있지 않습니다. 이 정보는 정책의 MFA 조건을 충족하기 위해 필요합니다.

샘플 코드: 멀티 팩터 인증이 포함된 자격 증명 요청

다음 예에서는 `GetSessionToken` 및 `AssumeRole` 작업을 호출하고 MFA 인증 파라미터를 전달하는 방법을 보여줍니다. 권한이 없어도 `GetSessionToken`을 호출할 수 있지만, `AssumeRole`을 호출할 수 있게 허용하는 정책이 있어야 합니다. 반환된 자격 증명은 계정 내 모든 S3 버킷의 목록을 나열하는 데 사용됩니다.

MFA 인증이 포함된 GetSessionToken 호출

다음 예는 GetSessionToken을 호출하고 MFA 인증 정보를 전달하는 방법을 보여 줍니다. GetSessionToken 작업에서 반환하는 임시 보안 자격 증명은 이어서 계정 내 모든 S3 버킷의 목록을 나열하는 데 사용됩니다.

이 코드를 실행하는 사용자(또는 사용자가 속한 그룹)에게 연결된 정책에서는 반환된 임시 자격 증명에 대한 권한을 제공합니다. 이 코드 예의 경우 정책에서 사용자에게 Amazon S3 ListBuckets 작업을 요청할 수 있는 권한을 부여해야 합니다.

다음 코드 예제는 GetSessionToken의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 자격 증명용 단기 보안 인증 정보 세트 가져오기

다음 get-session-token 명령은 호출을 위한 IAM 자격 증명용 단기 보안 인증 정보 세트를 검색합니다. 정책에 따라 다중 인증(MFA)이 필요한 경우 요청에 이 보안 인증 정보를 사용할 수 있습니다. 보안 인증 정보는 생성되고 15분 후에 만료됩니다.

```
aws sts get-session-token \
    --duration-seconds 900 \
    --serial-number "YourMFADeviceSerialNumber" \
    --token-code 123456
```

출력:

```
{
    "Credentials": {
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2020-05-19T18:06:10+00:00"
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 임시 보안 자격 증명 요청을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetSessionToken을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 설정된 기간 동안 유효한 임시 자격 증명이 포함된 **Amazon.RuntimeAWSCredentials** 인스턴스를 반환합니다. 임시 자격 증명을 요청하는 데 사용되는 자격 증명은 현재 쉘 기본값에서 유추됩니다. 다른 자격 증명을 지정하려면 -ProfileName 또는 -AccessKey/-SecretKey 파라미터를 사용합니다.

```
Get-STSSessionToken
```

출력:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

예제 2: 1시간 동안 유효한 임시 자격 증명이 포함된 **Amazon.RuntimeAWSCredentials** 인스턴스를 반환합니다. 요청에 사용되는 자격 증명은 지정된 프로파일에서 가져옵니다.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

출력:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

예제 3: 프로파일 'myprofilename'에 자격 증명이 지정된 계정과 연결된 MFA 디바이스의 식별 번호와 디바이스에서 제공한 값을 사용하여 1시간 동안 유효한 임시 자격 증명이 들어 있는 **Amazon.RuntimeAWSCredentials** 인스턴스를 반환합니다.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

출력:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetSessionToken을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

MFA 토큰을 전달하여 세션 토큰을 가져와 계정에 대한 Amazon S3 버킷을 나열하는 데 사용합니다.

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
```

```
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 GetSessionToken를 참조하십시오.

MFA 인증이 포함된 AssumeRole 호출

다음 예는 AssumeRole을(를) 호출하고 MFA 인증 정보를 전달하는 방법을 보여줍니다. AssumeRole에서 반환한 임시 보안 자격 증명은 계정의 모든 Amazon S3 버킷을 나열하는 데 사용됩니다.

이 시나리오에 대한 자세한 내용은 시나리오: 크로스 계정 위임에 대한 MFA 보호를 참조하세요.

다음 코드 예제는 AssumeRole의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

namespace AssumeRoleExample
{
    class AssumeRole
    {
        /// <summary>
        /// This example shows how to use the AWS Security Token
        /// Service (AWS STS) to assume an IAM role.
        ///
        /// NOTE: It is important that the role that will be assumed has a
        /// trust relationship with the account that will assume the role.
        ///
        /// Before you run the example, you need to create the role you want to
        /// assume and have it trust the IAM account that will assume that role.
        ///
        /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html
        /// for help in working with roles.
        /// </summary>

        private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

        static async Task Main()
        {
            // Create the SecurityToken client and then display the identity of
 the
            // default user.
```

```
            var roleArnToAssume = "arn:aws:iam::123456789012:role/
testAssumeRole";

            var client = new
 Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

            // Get and display the information about the identity of the default
 user.
            var callerIdRequest = new GetCallerIdentityRequest();
            var caller = await client.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"Original Caller: {caller.Arn}");

            // Create the request to use with the AssumeRoleAsync call.
            var assumeRoleReq = new AssumeRoleRequest()
            {
                DurationSeconds = 1600,
                RoleSessionName = "Session1",
                RoleArn = roleArnToAssume
            };

            var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);

            // Now create a new client based on the credentials of the caller
 assuming the role.
            var client2 = new AmazonSecurityTokenServiceClient(credentials:
 assumeRoleRes.Credentials);

            // Get and display information about the caller that has assumed the
 defined role.
            var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"AssumedRole Caller: {caller2.Arn}");
        }
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 AssumeRole을 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function sts_assume_role
#
# This function assumes a role in the AWS account and returns the temporary
#  credentials.
#
# Parameters:
#       -n role_session_name -- The name of the session.
#       -r role_arn -- The ARN of the role to assume.
#
# Returns:
```

```
#       [access_key_id, secret_access_key, session_token]
#     And:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function sts_assume_role() {
  local role_session_name role_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function sts_assume_role"
    echo "Assumes a role in the AWS account and returns the temporary
 credentials:"
    echo "  -n role_session_name -- The name of the session."
    echo "  -r role_arn -- The ARN of the role to assume."
    echo ""
  }

  while getopts n:r:h option; do
    case "${option}" in
      n) role_session_name=${OPTARG} ;;
      r) role_arn=${OPTARG} ;;
      h)
        usage
        return 0
        ;;
      \?)
        ech o"Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  response=$(aws sts assume-role \
    --role-session-name "$role_session_name" \
    --role-arn "$role_arn" \
    --output text \
    --query "Credentials.[AccessKeyId, SecretAccessKey, SessionToken]")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
```

```
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API 세부 정보는 AWS CLI명령 참조의 AssumeRole을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
                             const Aws::String &roleSessionName,
                             const Aws::String &externalId,
                             Aws::Auth::AWSCredentials &credentials,
                             const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
 result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 AssumeRole을 참조하십시오.

CLI

AWS CLI

역할 위임

다음 assume-role 명령은 IAM 역할 s3-access-example에 대한 일련의 단기 보안 인증 정보를 검색합니다.

```
aws sts assume-role \
    --role-arn arn:aws:iam::123456789012:role/xaccounts3access \
    --role-session-name s3-access-example
```

출력:

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:s3-access-example",
        "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-
access-example"
    },
    "Credentials": {
```

```
        "SecretAccessKey": "9drTJvcXLB89EXAMPLELB8923FB892xMFI",
        "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/
qwjzP2iEXAMPLEbw/m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8B
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=",
        "Expiration": "2016-03-15T00:05:07Z",
        "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
    }
}
```

명령의 출력에는 AWS 인증에 사용할 수 있는 액세스 키, 비밀 키 및 세션 토큰이 포함됩니다.

AWS CLI를 사용하는 경우 역할과 연결된 이름이 지정된 프로파일을 설정할 수 있습니다. 프로파일을 사용하면 AWS CLI에서 assume-role을 호출하고 대신 보안 인증 정보를 관리합니다. 자세한 내용은 AWS CLI 사용 설명서의 AWS CLI에서 IAM 역할 사용을 참조하세요.

- API 세부 정보는 AWS CLI명령 참조의 AssumeRole을 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;
```

```
/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 * "Version": "2012-10-17",
 * "Statement": [
 * {
 * "Effect": "Allow",
 * "Principal": {
 * "AWS": "<Specify the ARN of your IAM user you are using in this code
 * example>"
 * },
 * "Action": "sts:AssumeRole"
 * }
 * ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleArn> <roleSessionName>\s

                Where:
                    roleArn - The Amazon Resource Name (ARN) of the role to
 assume (for example, rn:aws:iam::000008047983:role/s3role).\s
                    roleSessionName - An identifier for the assumed role session
 (for example, mysession).\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
                .region(region)
                .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn,
String roleSessionName) {
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();

            // Display the time when the temp creds expire.
            Instant exTime = myCreds.expiration();
            String tokenInfo = myCreds.sessionToken();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(exTime);
            System.out.println("The token " + tokenInfo + "  expires on " +
exTime);

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AssumeRole을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

클라이언트를 생성합니다.

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an AWS STS service client object.
export const client = new STSClient({ region: REGION });
```

IAM 역할을 수임합니다.

```
import { AssumeRoleCommand } from "@aws-sdk/client-sts";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Returns a set of temporary security credentials that you can use to
    // access Amazon Web Services resources that you might not normally
    // have access to.
    const command = new AssumeRoleCommand({
      // The Amazon Resource Name (ARN) of the role to assume.
      RoleArn: "ROLE_ARN",
      // An identifier for the assumed role session.
```

```
      RoleSessionName: "session1",
      // The duration, in seconds, of the role session. The value specified
      // can range from 900 seconds (15 minutes) up to the maximum session
      // duration set for the role.
      DurationSeconds: 900,
    });
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 AssumeRole을 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
const AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

var roleToAssume = {
  RoleArn: "arn:aws:iam::123456789012:role/RoleName",
  RoleSessionName: "session1",
  DurationSeconds: 900,
};
var roleCreds;

// Create the STS service object
var sts = new AWS.STS({ apiVersion: "2011-06-15" });

//Assume Role
sts.assumeRole(roleToAssume, function (err, data) {
  if (err) console.log(err, err.stack);
  else {
```

```
    roleCreds = {
      accessKeyId: data.Credentials.AccessKeyId,
      secretAccessKey: data.Credentials.SecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    stsGetCallerIdentity(roleCreds);
  }
});

//Get Arn of current identity
function stsGetCallerIdentity(creds) {
  var stsParams = { credentials: creds };
  // Create STS service object
  var sts = new AWS.STS(stsParams);

  sts.getCallerIdentity({}, function (err, data) {
    if (err) {
      console.log(err, err.stack);
    } else {
      console.log(data.Arn);
    }
  });
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 AssumeRole을 참조하십시오.

PowerShell

PowerShell용 도구

요청하는 사용자가 일반적으로 액세스할 수 없는 AWS 리소스에 액세스하는 데 1시간 동안 사용할 수 있는 임시 자격 증명(액세스 키, 비밀 키 및 세션 토큰) 세트를 반환합니다. 반환된 자격 증명에는 수임 중인 역할의 액세스 정책과 제공된 정책에 의해 허용되는 권한이 있습니다. 제공된 정책을 사용하여 수임 중인 역할의 액세스 정책에 의해 정의된 권한을 초과하는 권한을 부여할 수 없습니다.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -Policy "...JSON policy..." -DurationInSeconds 3600
```

예제 2: 수임된 역할의 액세스 정책에 정의된 것과 동일한 권한을 갖고 1시간 동안 유효한 임시 자격 증명 세트를 반환합니다.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600
```

예제 3: cmdlet을 실행하는 데 사용되는 사용자 자격 증명과 연결된 MFA에서 생성된 토큰과 일련 번호를 제공하는 임시 자격 증명 세트를 반환합니다.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

예제 4: 고객 계정에 정의된 역할을 수임한 임시 자격 증명 세트를 반환합니다. 타사에서 수임할 수 있는 각 역할에 대해 고객 계정은 역할이 수임될 때마다 -ExternalID 파라미터로 전달되는 식별자를 사용하여 역할을 생성해야 합니다.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -ExternalId "ABC123"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 AssumeRole을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

MFA 토큰이 필요한 IAM 역할을 수임하고 임시 자격 증명을 사용하여 계정에 대한 Amazon S3 버킷을 나열합니다.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
```

```
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 AssumeRole를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: "create-use-assume-role-scenario"
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 AssumeRole을 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn assume_role(config: &SdkConfig, role_name: String, session_name:
 Option<String>) {
    let provider = aws_config::sts::AssumeRoleProvider::builder(role_name)
        .session_name(session_name.unwrap_or("rust_sdk_example_session".into()))
        .configure(config)
        .build()
        .await;

    let local_config = aws_config::from_env()
        .credentials_provider(provider)
        .load()
        .await;
    let client = Client::new(&local_config);
    let req = client.get_caller_identity();
    let resp = req.send().await;
    match resp {
        Ok(e) => {
            println!("UserID :               {}",
 e.user_id().unwrap_or_default());
            println!("Account:               {}",
 e.account().unwrap_or_default());
            println!("Arn    :               {}", e.arn().unwrap_or_default());
        }
        Err(e) => println!("{:?}", e),
    }
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 AssumeRole을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public func assumeRole(role: IAMClientTypes.Role, sessionName: String)
                async throws -> STSClientTypes.Credentials {
    let input = AssumeRoleInput(
        roleArn: role.arn,
        roleSessionName: sessionName
    )
    do {
        let output = try await stsClient.assumeRole(input: input)

        guard let credentials = output.credentials else {
            throw ServiceHandlerError.authError
        }

        return credentials
    } catch {
        throw error
    }
}
```

- API 세부 정보는 AWS SDK for Swift API 참조의 AssumeRole을 참조하십시오.

미사용 AWS 자격 증명 찾기

AWS 계정의 보안을 강화하려면 불필요한 IAM 사용자 보안 인증(암호와 액세스 키)을 제거합니다. 예를 들어, 사용자가 조직을 떠나거나 AWS 액세스가 더 이상 필요하지 않은 경우 해당 자격 증명을 찾아서 더 이상 작동하지 않도록 해야 합니다. 더 이상 필요 없는 자격 증명을 삭제하는 것이 가장 좋습니다. 나중에 필요한 경우가 생기면 언제든지 다시 생성할 수 있습니다. 적어도 암호를 변경하거나 액세스 키를 비활성화하여 이전 사용자가 더 이상 액세스할 수 없게 해야 합니다.

미사용은 이와는 다른 것으로 보통 특정 기간 동안 사용되지 않은 자격 증명을 뜻합니다.

미사용 암호 찾기

AWS Management Console을 사용하여 사용자의 암호 사용 정보를 볼 수 있습니다. 사용자 수가 많을 경우 콘솔을 사용하여 각 사용자가 자신의 콘솔 암호를 사용한 최종 시각에 대한 정보가 담긴 자격 증명 보고서를 다운로드할 수 있습니다. AWS CLI 또는 IAM API에서 해당 정보에 액세스할 수도 있습니다.

미사용 암호를 확인하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 필요할 경우 사용자 테이블에 Console last sign-in(콘솔 마지막 로그인) 열을 추가합니다.
   a. 테이블 위 맨 오른쪽에서 설정 아이콘
      ( )
      을 선택합니다.
   b. 표시되는 열 선택에서 콘솔 마지막 로그인을 선택합니다.
   c. 확인을 선택하여 사용자 목록으로 돌아갑니다.
4. [Console last sign-in(콘솔 마지막 로그인)] 열에 사용자가 콘솔을 통해 마지막으로 AWS에 로그인한 날짜가 표시됩니다. 이 정보를 통해 지정된 기간 이상 동안 암호를 사용하여 로그인하지 않은 사용자를 확인할 수 있습니다. 암호 사용자 중 로그인한 적이 없는 사용자는 이 열에 없음이라고 표시됩니다. 없음은 암호가 없는 사용자를 나타냅니다. 최근에 사용된 적이 없는 암호는 삭제해야 할 자격 증명을 식별하기 위한 좋은 기준이 될 수 있습니다.

> **Important**
>
> 서비스 문제로 인해 암호가 마지막으로 사용된 데이터에 2018년 5월 3일 22:50 PDT ~ 2018년 5월 23일 14:08 PDT 사이의 암호 사용이 포함되어 있지 않습니다. 이는 IAM 콘솔에 표시되는 마지막 로그인 날짜, IAM 자격 증명 보고서의 암호가 마지막으로 사용된 날짜와 GetUser API 연산에 의해 반환되는 암호가 마지막으로 사용된 날짜에 영향을 줍니다. 사용자가 해당 기간에 로그인한 경우 반환되는 암호가 마지막으로 사용된 날짜는 사용자가 2018년 5월 3일 이전에 마지막으로 로그인한 날짜입니다. 사용자가 2018년 5월 23일 14:08 PDT 이후에 로그인한 경우 반환되는 암호가 마지막으로 사용된 날짜는 정확합니다.
>
> 마지막으로 사용된 암호 정보를 사용하여 사용되지 않는 자격 증명을 식별하고 삭제하는 경우(예: 지난 90일 동안 AWS에 로그인하지 않은 사용자 삭제) 2018년 5월 23일 이후의 날짜를 포함하도록 평가 기간을 조정하는 것이 좋습니다. 또는 사용자가 액세스 키를 사용하여 AWS에 프로그래밍 방식으로 액세스하는 경우 액세스 키가 마지막으로 사용된 정보가 모든 날짜에 대해 정확하므로 해당 정보를 참조할 수 있습니다.

자격 증명 보고서를 다운로드하여 미사용 암호를 찾으려면(콘솔)

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 자격 증명 보고서를 선택합니다.
3. 보고서 다운로드를 선택하여 `status_reports_<date>T<time>.csv`라는 쉼표 구분 값(CSV) 파일을 다운로드합니다. 5번째 열에는 날짜 또는 다음 중 하나가 있는 `password_last_used` 열이 포함됩니다.
   - 해당 없음 - 할당된 암호가 전혀 없는 사용자
   - no_information - IAM이 2014년 10월 20일에 암호 수명을 추적하기 시작한 이후 암호를 사용하지 않은 사용자

미사용 암호를 찾으려면(AWS CLI)

미사용 암호를 찾으려면 다음 명령을 실행합니다.

- `aws iam list-users`는 각자 `PasswordLastUsed` 값이 있는 사용자 목록을 반환합니다. 값이 비어 있는 경우 사용자에게 암호가 없거나 2014년 10월 20일 IAM이 암호 수명을 추적하기 시작한 이후 암호가 사용되지 않은 것입니다.

미사용 암호를 찾으려면(AWS API)

미사용 암호를 찾으려면 다음 연산을 호출합니다.

- `ListUsers`는 각각 <PasswordLastUsed> 값이 있는 사용자의 집합을 반환합니다. 값이 비어 있는 경우 사용자에게 암호가 없거나 2014년 10월 20일 IAM이 암호 수명을 추적하기 시작한 이후 암호가 사용되지 않은 것입니다.

자격 증명 보고서를 다운로드하기 위한 명령어에 대한 자세한 내용은 자격 증명 보고서 가져오기(AWS CLI)섹션을 참조하세요.

미사용 액세스 키 찾기

AWS Management Console을 사용하여 사용자의 액세스 키 사용 정보를 볼 수 있습니다. 사용자 수가 많을 경우 콘솔을 사용하여 자격 증명 보고서를 다운로드하여 각 사용자가 자신의 액세스 키를 마지막으로 사용한 때를 알 수 있습니다. AWS CLI 또는 IAM API에서 해당 정보에 액세스할 수도 있습니다.

미사용 액세스 키를 확인하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 필요할 경우 사용자 테이블에 Access key last used(마지막으로 사용한 액세스 키) 열을 추가합니다.
   a. 테이블 위 맨 오른쪽에서 설정 아이콘 ( ) 을 선택합니다.
   b. 표시되는 열 선택에서 마지막으로 사용한 액세스 키를 선택합니다.
   c. 확인을 선택하여 사용자 목록으로 돌아갑니다.
4. Access key last used(마지막으로 사용한 액세스 키) 열에는 사용자가 프로그래밍 방식으로 AWS에 마지막으로 액세스한 때부터 경과한 일수가 표시됩니다. 이 정보를 통해 지정된 기간 이상 동안 액세스 키를 사용하지 않은 사용자를 확인할 수 있습니다. 액세스 키가 없는 사용자는 이 열에 –가

표시됩니다. 최근에 사용된 적이 없는 액세스 키는 삭제해야 할 자격 증명을 식별하기 위한 좋은 기준이 될 수 있습니다.

자격 증명 보고서를 다운로드하여 미사용 액세스 키를 찾으려면(콘솔)

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Credential Report(자격 증명 보고서)를 선택합니다.
3. 보고서 다운로드를 선택하여 `status_reports_<date>T<time>.csv`라는 쉼표 구분 값(CSV) 파일을 다운로드합니다. 열 11부터 13에는 액세스 키 1의 마지막 사용 날짜, 리전 및 서비스 정보가 표시됩니다. 열 16부터 18에는 액세스 키 2에 대해 동일한 정보가 표시됩니다. 값이 해당 없음으로 되어 있는 것은 사용자에게 액세스 키가 없거나 2015년 4월 22일 IAM이 액세스 키 수명을 추적하기 시작한 이후 사용자가 액세스 키를 사용하지 않았다는 것입니다.

미사용 액세스 키를 확인하려면(AWS CLI)

미사용 액세스 키를 찾으려면 다음 명령을 실행합니다.

- `aws iam list-access-keys`는 AccessKeyID를 포함해 사용자의 액세스 키에 대한 정보를 반환합니다.
- `aws iam get-access-key-last-used`는 액세스 키 ID를 받아들여 LastUsedDate 액세스 키의 마지막 사용 및 Region 마지막으로 요청된 서비스의 ServiceName을 포함하는 출력을 반환합니다. LastUsedDate가 없는 경우 2015년 4월 22일 IAM이 액세스 키 수명을 추적하기 시작한 이후 액세스 키가 사용되지 않은 것입니다.

미사용 액세스 키를 확인하려면(AWS API)

미사용 액세스 키를 찾으려면 다음 연산을 호출합니다.

- `ListAccessKeys`는 지정된 사용자와 연결된 액세스 키에 대한 AccessKeyID 값의 목록을 반환합니다.
- `GetAccessKeyLastUsed`는 액세스 키 ID를 받아들여 값의 집합을 반환합니다. LastUsedDate, 액세스 키가 마지막으로 사용된 Region 및 마지막으로 요청된 서비스의 ServiceName이 포함되어 있습니다. 값이 비어 있는 경우 사용자에게 액세스 키가 없거나 2015년 4월 22일 IAM이 액세스 키 수명을 추적하기 시작한 이후 액세스 키가 사용되지 않은 것입니다.

자격 증명 보고서를 다운로드하기 위한 명령어에 대한 자세한 내용은 자격 증명 보고서 가져오기(AWS CLI)섹션을 참조하세요.

AWS 계정의 자격 증명 보고서 생성

계정의 모든 사용자와 암호, 액세스 키, MFA 디바이스 등 이들의 자격 증명 상태를 나열하는 자격 증명 보고서를 생성하고 다운로드할 수 있습니다. AWS Management Console, AWS SDK 및 명령줄 도구 또는 IAM API에서 자격 증명 보고서를 가져올 수 있습니다.

자격 증명 보고서를 사용하면 감사 및 규정 준수에 도움이 됩니다. 이 보고서를 통해 암호, 액세스 키 업데이트 등 보안 인증 수명 주기 요구 사항이 어떤 영향을 주는지 감사할 수 있습니다. 외부 감사자에게 이 보고서를 제공하거나 보고서를 직접 다운로드할 권한을 감사자에게 부여할 수 있습니다.

최소 네 시간에 한 번씩 자격 증명 보고서를 생성할 수 있습니다. 보고서를 요청하면 IAM은 먼저 해당 AWS 계정의 보고서가 4시간 이내에 생성되었는지 여부를 확인합니다. 네 시간 이내에 생성된 경우 최신 보고서를 다운로드하고, 계정의 최신 보고서가 생성된지 네 시간이 넘었거나 해당 계정에 대한 이전 보고서가 없는 경우 IAM이 새 보고서를 생성하고 다운로드합니다.

주제

- 필수 권한
- 보고서 형식 이해
- 자격 증명 보고서 가져오기(콘솔)
- 자격 증명 보고서 가져오기(AWS CLI)
- 자격 증명 보고서 가져오기(AWS API)

필수 권한

보고서를 생성하고 다운로드하려면 다음 권한이 필요합니다.

- 자격 증명 보고서를 생성하려면 `iam:GenerateCredentialReport`
- 보고서를 다운로드하려면 `iam:GetCredentialReport`

보고서 형식 이해

자격 증명 보고서는 CSV(쉼표로 구분된 값) 파일 형식으로 되어 있습니다. 공통 스프레드시트 소프트웨어로 CSV 파일을 열어 분석을 수행하거나 CSV 파일을 프로그래밍 방식으로 사용하고 사용자 지정 분석을 수행하는 애플리케이션을 구축할 수 있습니다.

CSV 파일에는 다음 열이 포함되어 있습니다:

사용자

사용자의 표시 이름입니다.

arn

사용자의 Amazon 리소스 이름(ARN)입니다. ARN에 대한 자세한 내용은 IAM ARN 섹션을 참조하세요.

user_creation_time

사용자가 생성된 날짜 및 시간(ISO 8601 날짜-시간 형식)입니다.

password_enabled

사용자에게 암호가 있는 경우 이 값은 TRUE입니다. 그렇지 않으면 FALSE입니다. AWS 계정 루트 사용자 값은 항상 not_supported입니다.

password_last_used

AWS 웹 사이트에 로그인하는 데 AWS 계정 루트 사용자 또는 사용자의 암호가 마지막으로 사용된 날짜 및 시간(ISO 8601 날짜-시간 형식)입니다. 사용자의 마지막 로그인 시간을 캡처하는 AWS 웹 사이트는 AWS Management Console, AWS 토론 포럼, AWS Marketplace입니다. 암호가 5분 내에 두 번 이상 사용된 경우, 첫 번째 사용만 이 필드에 기록됩니다.

- 다음과 같은 경우 이 필드의 값은 no_information입니다.
  - 사용자의 암호가 사용된 적이 없는 경우.
  - 암호와 관련된 로그인 데이터가 없는 경우(예: IAM에서 2014년 10월 20일에 이 정보를 추적하기 시작한 이후로 사용자의 암호가 사용되지 않은 경우)
- 사용자에게 암호가 없는 경우, 이 필드의 값은 N/A(해당 사항 없음)입니다.

Important

서비스 문제로 인해 암호가 마지막으로 사용된 데이터에 2018년 5월 3일 22:50 PDT ~ 2018년 5월 23일 14:08 PDT 사이의 암호 사용이 포함되어 있지 않습니다. 이는 IAM 콘솔에 표시되는 마지막 로그인 날짜, IAM 자격 증명 보고서의 암호가 마지막으로 사용된 날짜와 GetUser API 연산에 의해 반환되는 암호가 마지막으로 사용된 날짜에 영향을 줍니다. 사용자가 해당 기간에 로그인한 경우 반환되는 암호가 마지막으로 사용된 날짜는 사용자가 2018년 5월 3일 이전에 마지막으로 로그인한 날짜입니다. 사용자가 2018년 5월 23일 14:08 PDT 이후에 로그인한 경우 반환되는 암호가 마지막으로 사용된 날짜는 정확합니다.

마지막으로 사용된 암호 정보를 사용하여 사용되지 않는 자격 증명을 식별하고 삭제하는 경우(예: 지난 90일 동안 AWS에 로그인하지 않은 사용자 삭제) 2018년 5월 23일 이후의 날짜를 포함하도록 평가 기간을 조정하는 것이 좋습니다. 또는 사용자가 액세스 키를 사용하여 AWS에 프로그래밍 방식으로 액세스하는 경우 액세스 키가 마지막으로 사용된 정보가 모든 날짜에 대해 정확하므로 해당 정보를 참조할 수 있습니다.

password_last_changed

사용자의 암호가 마지막으로 설정된 날짜 및 시간(ISO 8601 날짜-시간 형식)입니다. 사용자에게 암호가 없는 경우, 이 필드의 값은 N/A(해당 사항 없음)입니다. AWS 계정(루트)의 값은 항상 `not_supported`입니다.

password_next_rotation

계정에 암호 교체를 요구하는 암호 정책이 있는 경우, 사용자가 새 암호를 설정해야 할 때 이 필드에 날짜 및 시간(ISO 8601 날짜-시간 형식)이 포함됩니다. AWS 계정(루트)의 값은 항상 `not_supported`입니다.

mfa_active

사용자에 대해 멀티 팩터 인증(MFA) 디바이스를 사용하도록 설정된 경우, 이 값은 `TRUE`입니다. 그렇지 않은 경우 이 값은 `FALSE`입니다.

access_key_1_active

사용자에게 액세스 키가 있고 액세스 키의 상태가 `Active`이면, 이 값은 `TRUE`입니다. 그렇지 않은 경우 이 값은 `FALSE`입니다. 계정 루트 사용자와 IAM 사용자 모두에게 적용됩니다.

access_key_1_last_rotated

사용자의 액세스 키가 생성되었거나 마지막으로 변경된 날짜 및 시간(ISO 8601 날짜-시간 형식)입니다. 사용자에게 활성 상태의 액세스 키가 없는 경우, 이 필드의 값은 N/A(해당 사항 없음)입니다. 계정 루트 사용자와 IAM 사용자 모두에게 적용됩니다.

access_key_1_last_used_date

사용자의 액세스 키를 AWS API 요청 서명에 마지막으로 사용한 날짜 및 시간(ISO 8601 날짜-시간 형식)입니다. 액세스 키가 15분 내에 두 번 이상 사용된 경우, 첫 번째 사용만 이 필드에 기록됩니다. 계정 루트 사용자와 IAM 사용자 모두에게 적용됩니다.

다음과 같은 경우 이 필드의 값은 N/A(해당 사항 없음)입니다.

- 사용자에게 액세스 키가 없는 경우.

- 액세스 키가 사용된 적이 없는 경우.
- IAM에서 2015년 4월 22일에 이 정보를 추적하기 시작한 이후로 액세스 키가 사용되지 않은 경우

access_key_1_last_used_region

액세스 키가 마지막으로 사용된 AWS 리전입니다. 액세스 키가 15분 내에 두 번 이상 사용된 경우, 첫 번째 사용만 이 필드에 기록됩니다. 계정 루트 사용자와 IAM 사용자 모두에게 적용됩니다.

다음과 같은 경우 이 필드의 값은 N/A(해당 사항 없음)입니다.

- 사용자에게 액세스 키가 없는 경우.
- 액세스 키가 사용된 적이 없는 경우.
- IAM에서 2015년 4월 22일에 이 정보의 추적을 시작하기 전에 액세스 키가 마지막으로 사용된 경우
- 마지막으로 사용한 서비스가 리전 전용이 아닌 경우(예: Amazon S3)

access_key_1_last_used_service

액세스 키로 가장 최근에 액세스한 AWS 제품입니다. 이 필드의 값은 서비스의 네임스페이스를 사용합니다. 예를 들어 Amazon S3의 경우 s3, Amazon EC2의 경우 ec2를 사용합니다. 액세스 키가 15분 내에 두 번 이상 사용된 경우, 첫 번째 사용만 이 필드에 기록됩니다. 계정 루트 사용자와 IAM 사용자 모두에게 적용됩니다.

다음과 같은 경우 이 필드의 값은 N/A(해당 사항 없음)입니다.

- 사용자에게 액세스 키가 없는 경우.
- 액세스 키가 사용된 적이 없는 경우.
- IAM에서 2015년 4월 22일에 이 정보의 추적을 시작하기 전에 액세스 키가 마지막으로 사용된 경우

access_key_2_active

사용자에게 두 번째 액세스 키가 있고 두 번째 키의 상태가 Active이면, 이 값은 TRUE입니다. 그렇지 않은 경우 이 값은 FALSE입니다. 계정 루트 사용자와 IAM 사용자 모두에게 적용됩니다.

> **Note**
>
> 사용자는 쉽게 교체할 수 있도록 최대 2개의 액세스 키를 보유할 수 있습니다. 이 경우 키를 먼저 업데이트한 다음 이전 키를 삭제하면 됩니다. 액세스 키 업데이트에 대한 자세한 내용은 액세스 키 업데이트 섹션을 참조하세요.

access_key_2_last_rotated

사용자의 두 번째 액세스 키가 생성되었거나 마지막으로 변경된 날짜 및 시간(ISO 8601 날짜-시간 형식)입니다. 사용자에게 활성 상태의 두 번째 액세스 키가 있는 경우, 이 필드의 값은 N/A(해당 사항 없음)입니다. 계정 루트 사용자와 IAM 사용자 모두에게 적용됩니다.

access_key_2_last_used_date

AWS API 요청에 서명하는 데 사용자의 두 번째 액세스 키가 마지막으로 사용된 날짜 및 시간(ISO 8601 날짜-시간 형식)입니다. 액세스 키가 15분 내에 두 번 이상 사용된 경우, 첫 번째 사용만 이 필드에 기록됩니다. 계정 루트 사용자와 IAM 사용자 모두에게 적용됩니다.

다음과 같은 경우 이 필드의 값은 N/A(해당 사항 없음)입니다.

- 사용자에게 두 번째 액세스 키가 없는 경우.
- 사용자의 두 번째 액세스 키가 사용된 적이 없는 경우.
- IAM에서 2015년 4월 22일에 이 정보의 추적을 시작하기 전에 사용자의 두 번째 액세스 키가 마지막으로 사용된 경우

access_key_2_last_used_region

사용자의 두 번째 액세스 키가 마지막으로 사용된 AWS 리전입니다. 액세스 키가 15분 내에 두 번 이상 사용된 경우, 첫 번째 사용만 이 필드에 기록됩니다. 계정 루트 사용자와 IAM 사용자 모두에게 적용됩니다. 다음과 같은 경우 이 필드의 값은 N/A(해당 사항 없음)입니다.

- 사용자에게 두 번째 액세스 키가 없는 경우.
- 사용자의 두 번째 액세스 키가 사용된 적이 없는 경우.
- IAM에서 2015년 4월 22일에 이 정보의 추적을 시작하기 전에 사용자의 두 번째 액세스 키가 마지막으로 사용된 경우
- 마지막으로 사용한 서비스가 리전 전용이 아닌 경우(예: Amazon S3)

access_key_2_last_used_service

사용자의 두 번째 액세스 키로 가장 최근에 액세스한 AWS 서비스입니다. 이 필드의 값은 서비스의 네임스페이스를 사용합니다. 예를 들어 Amazon S3의 경우 s3, Amazon EC2의 경우 ec2를 사용합니다. 액세스 키가 15분 내에 두 번 이상 사용된 경우, 첫 번째 사용만 이 필드에 기록됩니다. 계정 루트 사용자와 IAM 사용자 모두에게 적용됩니다. 다음과 같은 경우 이 필드의 값은 N/A(해당 사항 없음)입니다.

- 사용자에게 두 번째 액세스 키가 없는 경우.
- 사용자의 두 번째 액세스 키가 사용된 적이 없는 경우.

- IAM에서 2015년 4월 22일에 이 정보의 추적을 시작하기 전에 사용자의 두 번째 액세스 키가 마지막으로 사용된 경우

cert_1_active

사용자에게 X.509 서명 인증서가 있고 해당 인증서의 상태가 `Active`인 경우, 이 값은 `TRUE`입니다. 그렇지 않은 경우 이 값은 `FALSE`입니다.

cert_1_last_rotated

사용자의 서명 인증서가 생성되었거나 마지막으로 변경된 날짜 및 시간(ISO 8601 날짜-시간 형식)입니다. 사용자에게 활성 상태의 서명 인증서가 있는 경우, 이 필드의 값은 N/A(해당 사항 없음)입니다.

cert_2_active

사용자에게 두 번째 X.509 서명 인증서가 있고 해당 인증서의 상태가 `Active`인 경우, 이 값은 `TRUE`입니다. 그렇지 않은 경우 이 값은 `FALSE`입니다.

> **Note**
>
> 사용자는 인증서 교체가 쉽도록 최대 두 개의 X.509 서명 인증서를 보유할 수 있습니다.

cert_2_last_rotated

사용자의 두 번째 서명 인증서가 생성되었거나 마지막으로 변경된 날짜 및 시간(ISO 8601 날짜-시간 형식)입니다. 사용자에게 활성 상태의 두 번째 서명 인증서가 있는 경우, 이 필드의 값은 N/A(해당 사항 없음)입니다.

자격 증명 보고서 가져오기(콘솔)

AWS Management Console을 사용하여 자격 증명 보고서를 CSV(쉼표로 구분된 값) 파일로 다운로드할 수 있습니다.

자격 증명 보고서를 다운로드하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 자격 증명 보고서를 선택합니다.
3. 보고서 다운로드를 선택합니다.

자격 증명 보고서 가져오기(AWS CLI)

자격 증명 보고서를 다운로드하려면(AWS CLI)

1. 자격 증명 보고서를 생성합니다. AWS에는 단일 보고서가 저장됩니다. 보고서가 있는 경우 자격 증명 보고서를 생성하면 이전 보고서를 덮어씁니다. `aws iam generate-credential-report`
2. 마지막으로 생성된 보고서를 봅니다. `aws iam get-credential-report`

자격 증명 보고서 가져오기(AWS API)

자격 증명 보고서를 다운로드하려면(AWS API)

1. 자격 증명 보고서를 생성합니다. AWS에는 단일 보고서가 저장됩니다. 보고서가 있는 경우 자격 증명 보고서를 생성하면 이전 보고서를 덮어씁니다. `GenerateCredentialReport`
2. 마지막으로 생성된 보고서를 봅니다. `GetCredentialReport`

CodeCommit용 IAM 자격 증명: Git 자격 증명, SSH 키 및 AWS 액세스 키

CodeCommit은 AWS 클라우드에서 프라이빗 Git 리포지토리를 호스팅하는 관리형 버전 관리 서비스입니다. CodeCommit을 사용하려면 CodeCommit 리포지토리와 통신하도록 Git 클라이언트를 구성합니다. 이 구성의 일환으로 CodeCommit에서 사용자 인증에 사용할 수 있는 IAM 자격 증명을 제공합니다. IAM에서는 세 가지 유형의 자격 증명으로 CodeCommit을 지원합니다.

- Git 자격 증명: HTTPS를 통해 CodeCommit 리포지토리와 통신하는 데 사용할 수 있는 IAM 생성 사용자 이름 및 암호 페어입니다.
- SSH 키: SSH를 통해 CodeCommit 리포지토리와 통신하기 위해 IAM 사용자와 연결할 수 있는 로컬로 생성된 퍼블릭-프라이빗 키 페어입니다.
- AWS 액세스 키: HTTPS를 통해 CodeCommit 리포지토리와 통신하기 위해 AWS CLI에 포함된 자격 증명 헬퍼와 함께 사용할 수 있습니다.

> **Note**
>
> 또 다른 AWS 계정에서 리포지토리에 액세스하기 위해 SSH 키나 Git 자격 증명을 사용할 수 없습니다. 다른 AWS 계정의 IAM 사용자 및 그룹에 대해 CodeCommit 리포지토리 액세스

를 구성하는 방법을 알아보려면 AWS CodeCommit 사용 설명서에서 역할을 사용하여 AWS CodeCommit 리포지토리에 대한 크로스 계정 액세스 구성을 참조하세요.

각 옵션에 대한 자세한 내용은 다음 섹션을 참조하세요.

CodeCommit에 Git 자격 증명 및 HTTPS 사용(권장)

Git 자격 증명을 사용하여 IAM 사용자에 대한 정적 사용자 이름 및 암호 페어를 생성한 다음 HTTPS 연결에 이러한 자격 증명을 사용합니다. 정적 Git 자격 증명을 지원하는 타사 도구 또는 IDE(통합 개발 환경)에서도 이러한 자격 증명을 사용할 수 있습니다.

이러한 자격 증명은 모든 지원되는 운영 체제에 공통적이고 대부분의 자격 증명 관리 시스템, 개발 환경 및 기타 소프트웨어 개발 도구와 호환되므로 이는 권장되는 방법입니다. 언제든지 Git 자격 증명에 대한 암호를 재설정할 수 있습니다. 또한 자격 증명이 더 이상 필요하지 않은 경우 자격 증명을 비활성화하거나 삭제할 수 있습니다.

> **Note**
>
> Git 보안 인증 정보에 대해 본인의 사용자 이름 또는 암호를 선택할 수 없습니다. IAM에서는 사용자가 AWS에 대한 보안 표준을 준수하고 CodeCommit에서 리포지토리를 보호하도록 돕기 위해 이러한 자격 증명을 생성합니다. 자격 증명은 생성될 때 한 번만 다운로드할 수 있습니다. 따라서 자격 증명을 안전한 장소에 보관하세요. 필요한 경우 언제든지 암호를 재설정할 수 있지만, 그러면 이전 암호를 사용하여 구성된 연결은 무효화됩니다. 새 암호를 사용하여 연결하려면 연결을 다시 구성해야 합니다.

자세한 내용은 다음 주제 섹션을 참조하세요.

- IAM 사용자를 생성하려면 AWS 계정에서 IAM 사용자 생성 섹션을 참조하세요
- CodeCommit에서 Git 자격 증명을 생성하고 사용하려면 AWS CodeCommit 사용 설명서의 Git 자격 증명을 사용하는 HTTPS 사용자의 경우를 참조하세요.

> **Note**
>
> Git 자격 증명을 생성한 이후에 IAM 사용자의 이름을 변경해도 Git 자격 증명의 사용자 이름은 변경되지 않습니다. 사용자 이름과 암호는 동일하게 유지되고 계속 유효합니다.

서비스별 보안 인증을 업데이트하려면

1. 현재 사용 중인 서비스별 자격 증명 세트 이외에 두 번째 세트를 만듭니다.
2. 새 자격 증명 세트를 사용하도록 모든 애플리케이션을 업데이트하고 애플리케이션이 작동하는지 확인합니다.
3. 원래 자격 증명의 상태를 "Inactive"로 변경합니다.
4. 모든 애플리케이션이 계속 작동하는지 확인합니다.
5. 비활성 서버별 자격 증명을 삭제합니다.

CodeCommit에 SSH 키 및 SSH 사용

SSH 연결을 사용하여 Git 및 CodeCommit에서 SSH 인증에 사용하는 퍼블릭 및 프라이빗 키 파일을 로컬 시스템에서 생성합니다. 퍼블릭 키를 IAM 사용자와 연결하고 프라이빗 키를 로컬 시스템에 저장합니다. 자세한 내용은 다음 주제 섹션을 참조하세요.

- IAM 사용자를 생성하려면 AWS 계정에서 IAM 사용자 생성 섹션을 참조하세요
- SSH 퍼블릭 키를 생성하여 IAM 사용자와 연결하려면 AWS CodeCommit 사용 설명서에서 Linux, macOS 또는 Unix에서 SSH 연결의 경우 또는 Windows에서 SSH 연결의 경우를 참조하세요.

> **Note**
>
> 퍼블릭 키는 ssh-rsa 형식 또는 PEM 형식으로 인코딩해야 합니다. 퍼블릭 키의 최소 비트 길이는 2048비트이고 최대 길이는 16384비트입니다. 이것은 업로드하는 파일의 크기와는 별개입니다. 예를 들어 2048비트 키를 생성할 수 있으며 결과 PEM 파일의 길이는 1679바이트입니다. 퍼블릭 키를 다른 형식 또는 크기로 제공하면 키 형식이 잘못되었다는 오류 메시지가 표시됩니다.

AWS CLI 자격 증명 헬퍼 및 CodeCommit에 HTTPS 사용

Git 자격 증명을 사용한 HTTPS 연결의 대안으로, Git에서 CodeCommit 리포지토리와 상호 작용하기 위해 AWS에 인증해야 할 때마다 암호화 방식으로 서명된 IAM 사용자 자격 증명 또는 Amazon EC2 인스턴스 역할을 사용하도록 허용할 수 있습니다. 이는 IAM 사용자 없이 CodeCommit 리포지토리에 연결할 수 있는 유일한 방법입니다. 또한 페더레이션 액세스 및 임시 자격 증명으로 작동되는 유일한 방법입니다. 자세한 내용은 다음 주제 섹션을 참조하세요.

- 페더레이션 액세스에 대한 자세한 내용은 자격 증명 공급자 및 페더레이션 및 외부에서 인증된 사용자에 대한 액세스(ID 페더레이션) 단원을 참조하십시오.
- 임시 자격 증명에 대한 자세한 내용은 IAM의 임시 보안 자격 증명 및 CodeCommit 리포지토리에 대한 임시 액세스를 참조하세요.

AWS CLI Credential Helper는 Keychain Access, Windows Credential Management 등과 같은 다른 자격 증명 헬퍼 시스템과 호환되지 않습니다. HTTPS를 통한 자격 증명 헬퍼 연결을 구성할 때 고려해야 할 추가 사항이 있습니다. 자세한 내용은 AWS CodeCommit 사용 설명서에서 AWS CLI 자격 증명 헬퍼를 사용한 Linux, macOS 또는 Unix에서 HTTPS 연결 또는 AWS CLI 자격 증명 헬퍼를 사용하여 Windows에서 HTTPS 연결을 참조하세요.

IAM에서 서버 인증서 관리

AWS에서 웹 사이트나 애플리케이션에 대한 HTTPS 연결을 활성화하려면 SSL/TLS 서버 인증서가 필요합니다. AWS Certificate Manager(ACM)에서 지원되는 리전에서 사용되는 인증서의 경우, ACM을 사용하여 서버 인증서를 프로비저닝, 관리 및 배포하는 것이 좋습니다. 지원되지 않는 리전에서는 IAM을 인증서 관리자로 사용해야 합니다. ACM이 지원하는 리전을 알아보려면 AWS 일반 참조의 AWS Certificate Manager 엔드포인트 및 할당량을 참조하세요.

> **Important**
>
> ACM은 서버 인증서를 프로비저닝, 관리 및 배포하기 위한 기본 도구입니다. ACM을 사용하면 인증서를 요청하거나 기존 ACM 또는 외부 인증서를 AWS 리소스에 배포할 수 있습니다. ACM이 제공하는 인증서는 무료이고 자동으로 갱신됩니다. 지원되는 리전에서는 ACM을 사용하여 콘솔에서 또는 프로그래밍 방식으로 서버 인증서를 관리할 수 있습니다. ACM 사용에 대한 자세한 내용은 AWS Certificate Manager 사용 설명서를 참조하세요. ACM 인증서 요청에 대한 자세한 내용은 AWS Certificate Manager 사용 설명서의 퍼블릭 인증서 요청 또는 프라이빗 인증서 요청을 참조하세요. 서드 파티 인증서를 ACM으로 가져오는 방법에 대한 자세한 내용은 AWS Certificate Manager 사용 설명서의 인증서 가져오기를 참조하세요.

ACM에서 지원하지 않는 리전에서 HTTPS 연결을 지원해야 하는 경우에만 IAM을 인증서 관리자로 사용합니다. IAM은 프라이빗 키를 안전하게 암호화하고 암호화된 버전을 IAM SSL 인증서 스토리지에 저장합니다. IAM은 모든 리전에서 서버 인증서 배포를 지원하지만 외부 공급자로부터 AWS에서 사용할 인증서를 얻어야 합니다. ACM 인증서는 IAM에 업로드할 수 없습니다. 또한 IAM 콘솔에서 인증서를 관리할 수 없습니다.

서드 파티 인증서를 IAM에 업로드하는 방법에 대한 자세한 정보는 다음 주제를 참조하세요.

주제

서버 인증서 업로드(AWS API)

IAM에 서버 인증서를 업로드하려면 인증서와 함께 그에 딸린 프라이빗 키를 제공해야 합니다. 인증서에 자체 서명이 되어 있지 않은 경우, 인증서 체인도 제공해야 합니다. (자체 서명된 인증서를 업로드하는 경우에는 인증서 체인이 필요하지 않습니다). 인증서를 업로드하기 전에 이 모든 항목이 있는지, 있다면 각 항목이 다음 기준을 충족하는지 확인하세요.

- 인증서는 업로드 시점에 유효해야 합니다. 유효 기간이 시작되기 전(인증서의 `NotBefore` 날짜) 또는 만료된 후(인증서의 `NotAfter` 날짜)에는 인증서를 업로드할 수 없습니다.
- 프라이빗 키는 암호화되지 않은 것이어야 합니다. 패스워드나 패스프레이즈로 보호된 프라이빗 키는 업로드할 수 없습니다. 암호화된 프라이빗 키의 해독에 대한 도움말은 서버 인증서 문제 해결 섹션을 참조하세요.
- 인증서, 프라이빗 키 및 인증서 체인은 모두 PEM 인코딩되어야 합니다. 이 항목들을 PEM 형식으로 변환하는 작업에 대한 도움말은 서버 인증서 문제 해결 섹션을 참조하세요.

IAM API를 사용하여 인증서를 업로드하려면 UploadServerCertificate 요청을 전송합니다. 다음 예에서는 AWS Command Line Interface(AWS CLI)에서 이 작업을 수행하는 방법을 보여줍니다. 이 예시에서는 다음과 같이 가정합니다.

- PEM 인코딩된 인증서가 `Certificate.pem`이라는 파일에 저장되어 있다.
- PEM 인코딩된 인증서 체인이 `CertificateChain.pem`이라는 파일에 저장되어 있다.
- PEM 인코딩된 비암호화 프라이빗 키가 `PrivateKey.pem`이라는 파일에 저장되어 있다.
- (선택 사항) 키 값 페어로 서버 인증서를 태깅하는 것이 좋습니다. 예를 들어 인증서를 식별하고 구성하는 데 도움이 되도록 태그 키 `Engineering`과 태그 값 `Department`를 추가할 수 있습니다.

다음 예제 명령을 사용하려면 이 같은 파일 이름을 고유한 이름으로 바꿉니다. *`ExampleCertificate`*를 업로드한 인증서의 이름으로 바꿉니다. 인증서를 태깅하려면 *`ExampleKey`* 및 *`ExampleValue`* 태그 키 값 페어를 고유한 값으로 바꿉니다. 하나의 연속선에 명령을 입력합니다. 다음 예시에는 가독성을 높여주는 줄바꿈과 추가 공백이 포함되어 있습니다.

```
aws iam upload-server-certificate --server-certificate-name ExampleCertificate
                                  --certificate-body file://Certificate.pem
                                  --certificate-chain file://CertificateChain.pem
                                  --private-key file://PrivateKey.pem
                                  --tags '{"Key": "ExampleKey", "Value":
 "ExampleValue"}'
```

선행 명령은 성공적으로 실행되는 경우 Amazon Resource Name(ARN), 표시 이름, 식별자(ID), 만료 날짜, 태그 등 업로드된 인증서에 대한 메타데이터를 반환합니다.

> **Note**
>
> Amazon CloudFront에서 사용할 서버 인증서를 업로드하는 경우 `--path` 옵션을 사용하여 경로를 지정해야 합니다. 경로는 `/cloudfront`로 시작해야 하고 후행 슬래시를 포함해야 합니다(예: `/cloudfront/test/`).

AWS Tools for Windows PowerShell를 사용하여 인증서를 업로드하려면 Publish-IAMServerCertificate를 사용하세요.

서버 인증서를 위한 AWS API 작업

서버 인증서를 보고, 태그를 지정하고, 이름을 바꾸고, 삭제하려면 다음 명령을 사용합니다.

- 인증서를 검색하려면 GetServerCertificate를 사용합니다. 이 요청은 인증서, 인증서 체인(업로드된 경우) 및 인증서 관련 메타데이터를 반환합니다.

  > **Note**
  >
  > 업로드 후에는 IAM에서 프라이빗 키를 다운로드하거나 조회할 수 없습니다.

- 인증서를 검색하려면 Get-IAMServerCertificate를 사용합니다.
- 업로드한 서버 인증서를 나열하려면 ListServerCertificates를 사용합니다. 이 요청은 각 인증서 관련 메타데이터가 담긴 목록을 반환합니다.
- 업로드한 서버 인증서의 목록을 나열하려면 Get-IAMServerCertificates를 사용합니다.
- 기존 서버 인증서에 태그를 지정하려면 TagServerCertificate를 사용합니다.
- 서버 인증서에서 태그를 해제하려면 UntagServerCertificate를 사용합니다.

- 서버 인증서의 이름을 바꾸거나 해당 경로를 업데이트하려면 UpdateServerCertificate를 사용합니다.

  다음 예에서는 AWS CLI에서 이 작업을 수행하는 방법을 보여줍니다.

  다음 예시 명령을 사용하려면 기존 및 신규 인증서의 이름과 인증서 경로를 바꾸고 명령을 하나의 연속선에 입력해야 합니다. 다음 예시에는 가독성을 높여주는 줄바꿈과 추가 공백이 포함되어 있습니다.

  ```
  aws iam update-server-certificate --server-certificate-name ExampleCertificate
                                    --new-server-certificate-
  name CloudFrontCertificate
                                    --new-path /cloudfront/
  ```

  AWS Tools for Windows PowerShell을(를) 사용하여 서버 인증서의 이름을 변경하거나 경로를 업데이트하려면 Update-IAMServerCertificate을 사용하세요.
- 서버 인증서를 삭제하려면 DeleteServerCertificate를 사용합니다.

  AWS Tools for Windows PowerShell을 사용하여 서버 인증서를 삭제하려면 Remove-IAMServerCertificate을 사용하세요.

서버 인증서 문제 해결

IAM에 인증서를 업로드하려면 인증서, 프라이빗 키 및 인증서 체인이 모두 PEM으로 인코딩되어 있어야 합니다. 또한 프라이빗 키가 암호화되지 않은 것이어야 합니다. 다음 예시를 참조하세요.

Example PEM으로 인코딩된 인증서 예시

```
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
```

Example PEM 인코딩된 비암호화 프라이빗 키 예시

```
-----BEGIN RSA PRIVATE KEY-----
Base64-encoded private key
-----END RSA PRIVATE KEY-----
```

Example PEM 인코딩된 인증서 체인 예시

인증서 체인에는 한 개 이상의 인증서가 포함되어 있습니다. 텍스트 편집기, Windows의 copy 명령 또는 Linux의 cat 명령을 사용하여 여러 인증서 파일을 하나의 체인으로 연결할 수 있습니다. 여러 개의 인증서를 포함하는 경우 각 인증서가 앞에 지정된 인증서를 인증해야 합니다. 루트 CA 인증서를 포함하여 인증서를 연결함으로써 이를 달성합니다.

다음 예시의 경우에는 세 개의 인증서가 포함되어 있지만, 사용자에 따라 인증서 체인에 포함된 인증서가 그보다 많거나 적을 수 있습니다.

```
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
```

이 항목들이 IAM에 업로드하기에 적합한 형식이 아닌 경우, OpenSSL을 사용하여 적합한 형식으로 변환할 수 있습니다.

인증서 또는 인증서 체인을 DER에서 PEM으로 변환하려면

다음 예시와 같이 OpenSSL x509 명령을 사용합니다. 다음 예시 명령에서 *Certificate.der*을 DER 인코딩된 인증서가 포함된 파일의 이름으로 대체합니다. *Certificate.pem*을, PEM 인코딩된 인증서를 포함할 출력 파일에 지정하려는 이름으로 바꿉니다.

```
openssl x509 -inform DER -in Certificate.der -outform PEM -out Certificate.pem
```

프라이빗 키를 DER에서 PEM으로 변환하려면

다음 예시와 같이 OpenSSL rsa 명령을 사용합니다. 다음 예시 명령에서 *PrivateKey.der*을 DER 인코딩된 프라이빗 키가 포함된 파일의 이름으로 대체해야 합니다. *PrivateKey.pem*을, PEM 인코딩된 프라이빗 키를 포함할 출력 파일에 지정하려는 이름으로 바꿉니다.

```
openssl rsa -inform DER -in PrivateKey.der -outform PEM -out PrivateKey.pem
```

암호화된 프라이빗 키를 해독하려면(패스워드나 패스프레이즈 제거)

다음 예시와 같이 OpenSSL rsa 명령을 사용합니다. 다음 예시 명령을 사용하려면 *EncryptedPrivateKey.pem*을 암호화된 프라이빗 키가 포함된 파일의 이름으로 대체해야 합니다. *PrivateKey.pem*을, PEM 인코딩된 비암호화 프라이빗 키를 포함할 출력 파일에 지정하려는 이름으로 바꿉니다.

```
openssl rsa -in EncryptedPrivateKey.pem -out PrivateKey.pem
```

인증서 번들을 PKCS#12(PFX)에서 PEM으로 변환하려면

다음 예시와 같이 OpenSSL pkcs12 명령을 사용합니다. 다음 예시 명령에서 *CertificateBundle.p12*를 PKCS#12 인코딩된 인증서 번들이 포함된 파일의 이름으로 대체합니다. *CertificateBundle.pem*을, PEM 인코딩된 인증서 번들을 포함할 출력 파일에 지정하려는 이름으로 대체합니다.

```
openssl pkcs12 -in CertificateBundle.p12 -out CertificateBundle.pem -nodes
```

인증서 번들을 PKCS#7에서 PEM으로 변환하려면

다음 예시와 같이 OpenSSL pkcs7 명령을 사용합니다. 다음 예시 명령에서 *CertificateBundle.p7b*를 PKCS#7 인코딩된 인증서 번들이 포함된 파일의 이름으로 대체합니다. *CertificateBundle.pem*을, PEM 인코딩된 인증서 번들을 포함할 출력 파일에 지정하려는 이름으로 대체합니다.

```
openssl pkcs7 -in CertificateBundle.p7b -print_certs -out CertificateBundle.pem
```

IAM 사용자 그룹

IAM 사용자 그룹은 IAM 사용자의 집합입니다. 사용자 그룹을 활용하면 다수의 사용자들에 대한 권한을 지정함으로써 해당 사용자들에 대한 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, Admins라는 사용자 그룹이 있고 해당 사용자 그룹에 일반 관리자 권한을 부여할 수 있습니다. 해당 사용자 그룹의 모든 사용자는 자동으로 Admins 그룹 권한을 갖습니다. 관리자 권한을 필요로 하는 새로운 사용자가 조

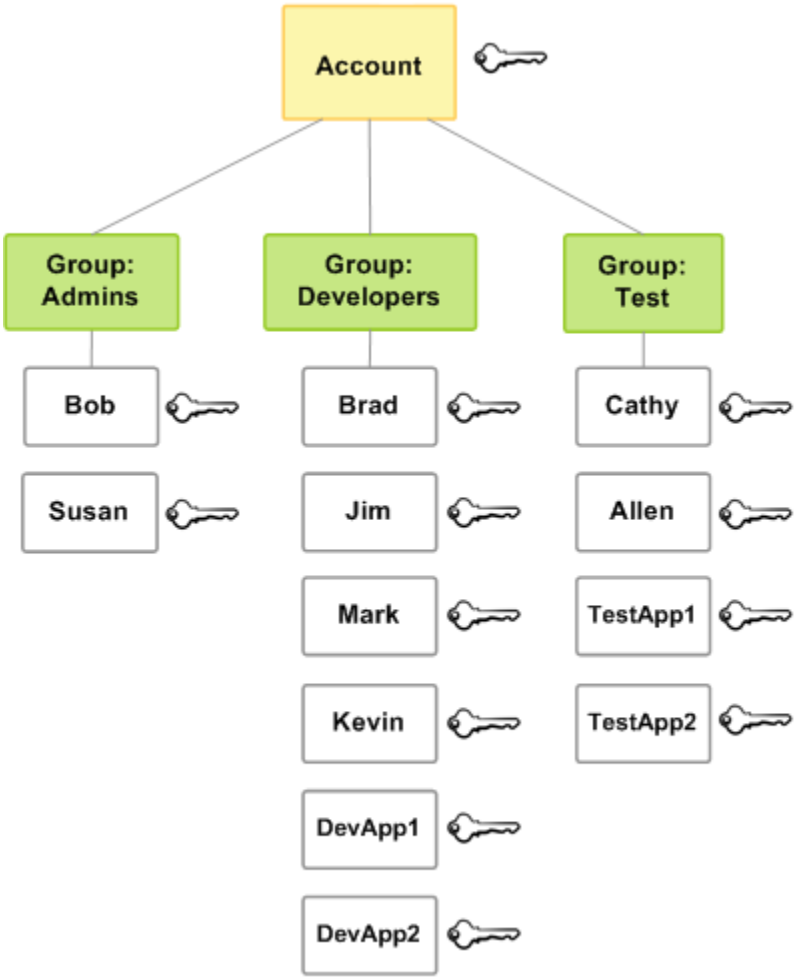
직에 들어올 경우 해당 사용자를 Admins 사용자 그룹에 추가하여 적절한 권한을 할당할 수 있습니다. 조직에서 직원의 업무가 바뀌면 해당 사용자의 권한을 편집하는 대신 이전 사용자 그룹에서 해당 사용자를 제거한 후 적절한 새 사용자 그룹에 추가하면 됩니다.

사용자 그룹의 모든 사용자가 정책의 권한을 받도록 아이덴티티 기반 정책을 사용자 그룹에 연결할 수 있습니다. 그룹은 인증이 아니라 권한과 관련이 있고 보안 주체는 인증된 IAM 엔터티이기 때문에 정책(예: 리소스 기반 정책)에서 사용자 그룹을 `Principal`로 식별할 수 없습니다. 정책 유형에 대한 자세한 정보는 자격 증명 기반 정책 및 리소스 기반 정책 섹션을 참조하세요.

다음은 사용자 그룹이 갖는 몇 가지 중요한 특징입니다.

- 한 사용자 그룹에 여러 사용자가 포함될 수 있으며 한 사용자가 다중 사용자 그룹에 속할 수 있습니다.
- 사용자 그룹은 중첩될 수 없습니다. 즉, 사용자 그룹은 사용자만 포함할 수 있으며 다른 그룹은 포함할 수 없습니다.
- AWS 계정의 모든 사용자를 자동으로 포함하는 기본 사용자 그룹은 없습니다. 이러한 사용자 그룹이 필요한 경우 하나 만들어 새로운 사용자를 각각 해당 그룹에 할당해야 합니다.
- 그룹 수, 사용자가 멤버가 될 수 있는 그룹 수와 같이 AWS 계정에 있는 IAM 리소스의 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 단원을 참조하십시오.

다음 다이어그램에서는 어느 작은 회사의 간단한 예제를 보여줍니다. 회사 소유주는 `Admins` 사용자 그룹을 생성해 회사가 성장함에 따라 사용자들이 다른 사용자들을 생성하고 관리하도록 합니다. `Admins` 사용자 그룹은 `Developers` 사용자 그룹 및 `Test` 사용자 그룹을 생성합니다. 이러한 각 사용자 그룹은 AWS와 상호 작용하는 사용자(사람 및 애플리케이션: Jim, Brad, DevApp1 등)들로 구성됩니다. 사용자마다 개별적인 보안 자격 증명 세트가 있습니다. 이 예제에서는 각 사용자가 단일 사용자 그룹에 속합니다. 하지만 사용자는 다중 사용자 그룹에 속할 수 있습니다.



IAM 사용자 그룹 생성

> **Note**
>
> 가장 좋은 방법은 인간 사용자가 ID 제공업체와의 페더레이션을 사용하여 임시 보안 인증으로 AWS에 액세스하도록 하는 것입니다. 이 방법을 따르는 경우 사용자가 IAM 사용자 및 그룹을 관리하지 않습니다. 대신 사용자와 그룹은 AWS 외부에서 관리되며 페더레이션형 ID로 AWS 리소스에 액세스할 수 있습니다. AWS 페더레이션형 ID는 엔터프라이즈 사용자 디렉터리, 웹 ID 제공업체, Directory Service, Identity Center 디렉터리의 사용자 또는 ID 소스를 통해 제공된 보안 인증을 사용하여 AWS 서비스에 액세스하는 모든 사용자입니다. 페더레이션형 ID는 ID 제공업체에서 정의한 그룹을 사용합니다. AWS IAM Identity Center를 사용하는 경우 IAM Identity Center에서 사용자 및 그룹 생성에 대한 자세한 내용은 AWS IAM Identity Center User

Guide(SSO 사용 설명서)의 Manage identities in IAM Identity Center(IAM Identity Center에서 ID 관리)를 참조하세요.

사용자 그룹을 설정하려면 그룹을 생성해야 합니다. 그런 다음 그룹 내 사용자가 할 수 있는 작업 유형에 따라 권한을 부여해야 합니다. 마지막으로 그룹에 사용자를 추가합니다.

사용자 그룹을 생성하기 위해 필요한 권한에 대한 자세한 내용은 IAM 리소스에 액세스하는 데 필요한 권한 섹션을 참조하세요.

IAM 사용자 그룹을 만들어 정책을 연결하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자 그룹(User groups)을 선택한 다음, 그룹 생성(Create group)을 선택합니다.
3. 사용자 그룹 이름(User group name)에 그룹의 이름을 입력합니다.

   > **Note**
   >
   > AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 단원을 참조하십시오. 그룹 이름에는 최대 128개의 문자, 숫자 및 더하기(+), 등호(=), 쉼표(,), 마침표(.), 앳(@), 밑줄(_) 및 하이픈(-) 조합을 사용할 수 있습니다. 이름은 계정 내에서 고유해야 합니다. 대소문자는 구별하지 않습니다. 예를 들어 이름이 **`ADMINS`** 및 **`admins`**, 두 가지로 지정된 그룹을 만들 수는 없습니다.

4. 사용자 목록에서 그룹에 추가하고자 하는 각 사용자의 확인란을 선택합니다.
5. 정책 목록에서 그룹 멤버 전체에 적용하고자 하는 정책 이름마다 확인란을 선택합니다.
6. 그룹 생성을 선택합니다.

IAM 사용자 그룹을 만들려면(AWS CLI또는 APIAWS)

다음 중 하나를 사용하세요.

- AWS CLI: aws iam create-group
- AWS API: CreateGroup

사용자 그룹 보기

계정의 모든 사용자 그룹, 사용자 그룹에 속한 사용자 및 사용자가 속한 사용자 그룹의 목록을 조회할 수 있습니다. AWS CLI 또는 AWS API를 사용하는 경우 특정 경로 접두사를 사용해 전체 사용자 그룹의 목록을 조회할 수 있습니다.

계정에 속한 모든 사용자 그룹을 표시하려면

다음을 수행하세요.

- AWS Management Console: 탐색 창에서 사용자 그룹(User groups)을 선택합니다.
- AWS CLI: aws iam list-groups
- AWS API: ListGroups

특정 사용자 그룹에 속한 사용자를 표시하려면

다음을 수행하세요.

- AWS Management Console: 탐색 창에서 사용자 그룹(User groups)을 선택하고, 그룹 이름을 선택한 후 사용자(Users) 탭을 선택합니다.
- AWS CLI: aws iam get-group
- AWS API: GetGroup

사용자가 속한 모든 사용자 그룹을 표시하려면

다음을 수행하세요.

- AWS Management Console: 탐색 창에서 사용자를 선택하고, 사용자 이름을 선택한 후 그룹 탭을 선택합니다.
- AWS CLI: aws iam list-groups-for-user
- AWS API: ListGroupsForUser

IAM 사용자 그룹의 사용자 편집

사용자 그룹을 사용하면 한 번에 여러 사용자에게 동일한 권한 정책을 적용할 수 있습니다. 그런 다음 IAM 사용자 그룹에서 사용자를 추가하거나 제거할 수 있습니다. 이 기능은 사람들이 조직에 들어오거나 조직을 떠날 때 유용합니다.

정책 액세스 보기

정책에 대한 권한을 변경하기 전에 최근 서비스 수준 활동을 검토해야 합니다. 이 기능은 사용 중인 보안 주체(사람 또는 애플리케이션)의 액세스 권한을 제거하지 않으려는 경우 중요합니다. 마지막으로 액세스한 정보 보기에 대한 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 섹션을 참조하세요.

사용자 그룹에서 사용자 추가 또는 제거(콘솔)

AWS Management Console을 사용하여 사용자 그룹에서 사용자를 추가 또는 제거할 수 있습니다.

IAM 사용자 그룹에 사용자를 추가하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자 그룹(User groups)을 선택한 다음 그룹 이름을 선택합니다.
3. 사용자(Users) 탭을 선택한 후 사용자 추가(Add Users)를 선택합니다. 추가할 사용자 옆에 있는 확인란을 선택합니다.
4. 사용자 추가(Add users)를 선택합니다.

IAM 그룹에서 사용자를 제거하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자 그룹(User groups)을 선택한 다음 그룹 이름을 선택합니다.
3. 사용자 탭을 선택합니다. 제거할 사용자 옆에 있는 확인란을 선택한 다음 사용자 제거(Remove users)를 선택합니다.

사용자 그룹에서 사용자 추가 또는 제거(AWS CLI)

AWS CLI를 사용하여 사용자 그룹에서 사용자를 추가 또는 제거할 수 있습니다.

IAM 사용자 그룹에 사용자를 추가하려면(AWS CLI)

- 다음 명령을 사용합니다.
  - aws iam add-user-to-group

IAM 사용자 그룹에서 사용자를 제거하려면(AWS CLI)

- 다음 명령을 사용합니다.
  - aws iam remove-user-from-group

사용자 그룹에서 사용자 추가 또는 제거(AWS API)

AWS API를 사용하여 사용자 그룹에서 사용자를 추가 또는 제거할 수 있습니다.

IAM 그룹에 사용자를 추가하려면(AWS API)

- 다음 작업을 완료합니다.
  - AddUserToGroup

IAM 사용자 그룹에서 사용자를 제거하려면(AWS API)

- 다음 작업을 완료합니다.
  - RemoveUserFromGroup

IAM 사용자 그룹에 정책 연결

다음 단계에 설명된 대로 사용자 그룹에 AWS 관리형 정책, 즉 AWS가 제공하는 미리 작성된 정책을 연결할 수 있습니다. 고객 관리형 정책, 즉 사용자가 생성한 사용자 지정 권한이 있는 정책을 연결하려면 먼저 정책을 생성해야 합니다. 고객 관리형 정책 만들기에 대한 자세한 내용은 고객 관리형 정책으로 사용자 지정 IAM 권한 정의 섹션을 참조하세요.

권한 및 정책에 대한 자세한 내용은 AWS리소스에 대한 액세스 관리을 참조하세요.

사용자 그룹에 정책을 연결하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자 그룹(User groups)을 선택한 다음 그룹 이름을 선택합니다.
3. 권한 탭을 선택합니다.
4. 권한 추가를 선택한 다음, 정책 연결을 선택합니다.

5. 사용자 그룹에 연결된 현재 정책이 현재 권한 정책(Current permissions policies) 목록에 표시됩니다. 기타 권한 정책(Other permissions policies)의 목록에서 연결할 정책 이름 옆의 확인란을 선택합니다. 검색 상자를 사용하여 정책 목록을 유형 및 정책 이름별로 필터링할 수 있습니다.
6. IAM 사용자 그룹에 연결하려는 정책을 선택하고 정책 연결을 선택합니다.

사용자 그룹에 정책을 연결하려면(AWS CLI 또는 AWS API)

다음 중 하나를 수행하세요.

- AWS CLI: aws iam attach-group-policy
- AWS API: AttachGroupPolicy

IAM 사용자 그룹 이름 변경

사용자 그룹의 이름 또는 경로를 변경하면 다음과 같이 진행됩니다.

- 사용자 그룹에 연결된 정책은 새 이름의 그룹에 그대로 유지됩니다.
- 사용자 그룹의 모든 사용자도 새로운 이름의 그룹에 그대로 유지됩니다.
- 사용자 그룹의 고유 ID는 변동 없이 유지됩니다. 고유 ID에 대한 자세한 내용은 고유 식별자 섹션을 참조하세요.

해당 사용자 그룹을 리소스로 참조하는 정책은 새 이름을 사용하도록 IAM에서 자동으로 업데이트하지 않습니다. 따라서 사용자 그룹 이름을 변경할 때는 주의해야 합니다. 사용자 그룹의 이름을 바꾸기 전에 모든 정책을 수동으로 확인해 해당 사용자 그룹이 이름으로 언급된 모든 정책을 찾아야 합니다. Bob이라는 직원이 회사의 테스트 부서 관리자인 경우를 예로 들어 보겠습니다. Bob의 IAM 사용자 엔터티에는 정책이 연결되어 있고 이 정책에 따라 Bob은 테스트 사용자 그룹의 사용자를 추가하거나 제거할 수 있습니다. 관리자가 사용자 그룹의 이름이나 그룹 경로를 변경하는 경우, 관리자는 Bob에게 연결된 정책도 업데이트하여 새 이름이나 경로를 사용하도록 해야 합니다. 그러지 않으면 Bob은 사용자 그룹에 사용자를 추가할 수도 그룹에서 사용자를 제거할 수도 없습니다.

사용자 그룹을 리소스로 참조하는 정책을 찾으려면

1. IAM 콘솔의 탐색 창에서 정책(Policies)을 선택합니다.
2. 유형(Type) 열을 기준으로 정렬하여 고객 관리형(Customer managed) 사용자 지정 정책을 찾습니다.
3. 편집할 정책의 정책 이름을 선택합니다.

4. 권한 탭을 선택한 후 요약을 선택합니다.
5. 서비스 목록에서 IAM이 있으면 선택합니다.
6. 리소스(Resource) 열에서 사용자 그룹의 이름을 찾습니다.
7. 편집을 선택하여 정책에서 사용자 그룹 이름을 변경합니다.

IAM 사용자 그룹의 이름을 변경하려면

다음을 수행하세요.

- AWS Management Console: 탐색 창에서 사용자 그룹(User groups)을 선택한 다음 그룹 이름을 선택합니다. 편집을 선택합니다. 새 사용자 그룹 이름을 입력한 다음 변경 사항 저장(Save changes)을 선택합니다.
- AWS CLI: aws iam update-group
- AWS API: UpdateGroup

IAM 사용자 그룹 삭제

AWS Management Console에서 사용자 그룹을 삭제하면 콘솔은 모든 그룹 구성원을 자동으로 제거하고 연결된 모든 관리형 정책을 분리하며, 모든 인라인 정책을 삭제합니다. 그러나 IAM은 이러한 사용자 그룹을 리소스로 참조하는 정책을 자동으로 삭제하지 않기 때문에 사용자 그룹을 삭제할 때 주의해야 합니다. 사용자 그룹을 삭제하기 전에 모든 정책을 수동으로 확인하여 해당 그룹이 이름으로 언급된 모든 정책을 찾아야 합니다. 예를 들어, 테스트 팀 관리자인 John은 IAM 사용자 엔터티에 연결된 정책을 갖고 있어 테스트 사용자 그룹에서 사용자를 추가하고 제거할 수 있습니다. 관리자는 그룹을 삭제할 경우 John에게 연결된 정책도 삭제해야 합니다. 그렇지 않고 관리자가 삭제된 그룹을 다시 생성하고 동일한 이름을 지정하면 John이 테스트 팀을 떠나더라도 존의 권한은 그대로 유지됩니다.

사용자 그룹을 리소스로 참조하는 정책을 찾으려면

1. IAM 콘솔의 탐색 창에서 정책(Policies)을 선택합니다.
2. 유형(Type) 열을 기준으로 정렬하여 고객 관리형(Customer managed) 사용자 지정 정책을 찾습니다.
3. 삭제할 정책의 정책 이름을 선택합니다.
4. 권한 탭을 선택한 후 요약을 선택합니다.
5. 서비스 목록에서 IAM이 있으면 선택합니다.
6. 리소스(Resource) 열에서 사용자 그룹의 이름을 찾습니다.

7. 삭제를 선택하여 정책을 삭제합니다.
8. 정책 이름을 입력하여 정책 삭제를 확인하고 삭제를 선택합니다.

반면에 AWS CLI, Tools for Windows PowerShell 또는 AWS API를 사용하여 사용자 그룹을 삭제할 경우에는 먼저 그룹의 사용자를 제거해야 합니다. 그런 다음 이 사용자 그룹에 포함된 인라인 정책을 삭제합니다. 그런 다음 그룹에 연결된 관리형 정책을 모두 분리합니다. 이렇게 해야만 사용자 그룹을 삭제할 수 있습니다.

IAM 사용자 그룹 삭제(콘솔)

AWS Management Console에서 IAM 사용자 그룹을 삭제할 수 있습니다.

IAM 사용자 그룹을 삭제하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자 그룹(User groups)을 선택합니다.
3. 사용자 그룹 목록에서 삭제할 사용자 그룹 이름 옆의 확인란을 선택합니다. 검색 상자를 사용하여 유형, 권한 및 사용자 그룹 이름별로 사용자 그룹 목록을 필터링할 수 있습니다.
4. Delete(삭제)를 선택합니다.
5. 확인 상자에서 단일 사용자 그룹을 삭제하려면 해당 사용자 그룹 이름을 입력하고 삭제(Delete)를 선택합니다. 여러 사용자 그룹을 삭제하려면 삭제할 사용자 그룹 수와 **`user groups`**를 차례로 입력하고 삭제(Delete)를 선택합니다. 예를 들어 세 개의 사용자 그룹을 삭제하는 경우 **`3 user groups`**를 입력합니다.

IAM 사용자 그룹 삭제(AWS CLI)

AWS CLI에서 IAM 사용자 그룹을 삭제할 수 있습니다.

IAM 사용자 그룹을 삭제하려면(AWS CLI)

1. 사용자 그룹에서 모든 사용자를 제거합니다.
   - aws iam get-group(사용자 그룹의 사용자 목록 확인) 및 aws iam remove-user-from-group(사용자 그룹에서 사용자 제거)
2. 사용자 그룹에 포함된 인라인 정책을 모두 삭제합니다.

- aws iam list-group-policies(사용자 그룹의 인라인 정책 목록 확인) 및 aws iam delete-group-policy(사용자 그룹의 인라인 정책 삭제)

3. 사용자 그룹에 연결된 관리형 정책을 모두 분리합니다.

   - aws iam list-attached-group-policies(사용자 그룹에 연결된 관리형 정책 목록 확인) 및 aws iam detach-group-policy(사용자 그룹에서 관리형 정책 분리)

4. 사용자 그룹을 삭제합니다.

   - aws iam delete-group

IAM 사용자 그룹 삭제(AWS API)

AWS API를 사용하여 IAM 사용자 그룹을 삭제할 수 있습니다.

IAM 사용자 그룹을 삭제하려면(AWS API)

1. 사용자 그룹에서 모든 사용자를 제거합니다.

   - GetGroup(사용자 그룹의 사용자 목록 확인) 및 RemoveUserFromGroup(사용자 그룹에서 사용자 제거)

2. 사용자 그룹에 포함된 인라인 정책을 모두 삭제합니다.

   - ListGroupPolicies(사용자 그룹의 인라인 정책 목록 확인) 및 DeleteGroupPolicy(사용자 그룹의 인라인 정책 삭제)

3. 사용자 그룹에 연결된 관리형 정책을 모두 분리합니다.

   - ListAttachedGroupPolicies(사용자 그룹에 연결된 관리형 정책의 목록 확인) 및 DetachGroupPolicy(사용자 그룹에서 관리형 정책 분리)

4. 사용자 그룹을 삭제합니다.

   - DeleteGroup

IAM 역할

IAM 역할은 계정에 생성할 수 있는, 특정 권한을 지닌 IAM 자격 증명입니다. AWS에서 자격 증명이 할 수 있는 것과 없는 것을 결정하는 권한 정책을 갖춘 AWS 자격 증명이라는 점에서 IAM 역할은 IAM 사용자와 유사합니다. 그러나 역할은 한 사람하고만 연관되지 않고 해당 역할이 필요한 사람이라면 누구

든지 맡을 수 있어야 합니다. 또한 역할에는 그와 연관된 암호 또는 액세스 키와 같은 표준 장기 자격 증명이 없습니다. 대신에 역할을 맡은 사람에게는 해당 역할 세션을 위한 임시 보안 자격 증명이 제공됩니다.

역할을 사용하여 일반적으로 AWS 리소스에 액세스할 수 없는 사용자, 애플리케이션 또는 서비스에 액세스 권한을 위임할 수 있습니다. 예를 들어 AWS 계정의 사용자에게 이들이 대개 권한이 없는 리소스에 대한 액세스 권한을 부여하거나 한 AWS 계정의 사용자에게 다른 계정의 리소스에 대한 액세스 권한을 부여해야 할 경우가 있습니다. 또는 모바일 앱에서 AWS 리소스를 사용할 수 있도록 허용하되 앱에 AWS 키를 내장하길 원치 않는 경우(업데이트하기 어렵고 사용자가 추출할 가능성이 있음)도 있습니다. 때로는 기업 디렉터리에서처럼 AWS 외부에 정의된 자격 증명을 이미 보유하고 있는 사용자에게 AWS 액세스 권한을 부여해야 하는 경우도 있습니다. 또는 타사에 계정에 대한 액세스 권한을 부여하여 리소스에 대한 감사를 수행할 수 있도록 해야 할 경우도 있을 수 있습니다.

이러한 경우 IAM 역할을 사용하여 AWS 리소스에 대한 액세스 권한을 위임할 수 있습니다. 이 단원에서는 역할 및 역할을 사용할 수 있는 여러 가지 방법, 다양한 접근 방식을 선택하는 경우와 방법, 역할을 생성, 관리, 전환(또는 수임) 및 삭제하는 방법을 소개합니다.

> **Note**
>
> AWS 계정을 처음 만들 때는 기본적으로 역할이 생성되지 않습니다. 계정에 서비스를 추가하면 서비스 연결 역할을 추가하여 사용 사례를 지원할 수 있습니다.
> 서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.
> 서비스 연결 역할을 삭제하려면 먼저 관련 리소스를 삭제해야 합니다. 이렇게 하면 리소스에 대한 액세스 권한을 부주의로 삭제할 수 없기 때문에 리소스가 보호됩니다.
> 서비스 연결 역할의 사용을 지원하는 서비스에 대한 자세한 정보는 AWS IAM으로 작업하는 서비스를 참조하고 서비스 연결 역할 열에 예가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

역할 용어 및 개념

아래는 역할을 시작하는 데 도움이 되는 몇 가지 기본 용어들입니다.

역할

계정에서 생성할 수 있는 특정 권한을 가진 IAM 자격 증명입니다. IAM 역할은 IAM 사용자와 몇 가지 점에서 유사합니다. 역할과 사용자 모두 AWS에서 자격 증명으로 할 수 있는 것과 할 수 없는 것을 결정하는 권한 정책을 포함하는 AWS 자격 증명입니다. 그러나 역할은 한 사람하고만 연관되지 않고 해당 역할이 필요한 사람이라면 누구든지 맡을 수 있어야 합니다. 또한 역할에는 그와 연관된 암호 또는 액세스 키와 같은 표준 장기 자격 증명이 없습니다. 대신에 역할을 맡은 사람에게는 해당 역할 세션을 위한 임시 보안 자격 증명이 제공됩니다.

역할은 다음의 주체들이 수임할 수 있습니다.

- 동일한 AWS 계정 또는 다른 AWS 계정의 IAM 사용자
- 동일한 계정의 IAM 역할
- 다음과 같은 AWS 서비스 및 기능과 함께 사용할 수 있는 서비스 보안 주체:
  - Amazon EC2 또는 Lamda와 같은 컴퓨팅 서비스에서 코드를 실행할 수 있게 해주는 서비스
  - Amazon S3 객체 복제와 같이 사용자를 대신하여 리소스에 작업을 수행하는 기능
  - IAM Roles Anywhere 또는 Amazon ECS Anywhere와 같이 AWS 외부에서 실행되는 애플리케이션에 임시 보안 자격 증명을 제공하는 서비스
- SAML 2.0, OpenID Connect와 호환되는 ID 제공업체(idP) 서비스에 의해 인증된 외부 사용자

AWS 서비스 역할

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

AWS 서비스 연결 역할

서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

> **Note**
>
> 서비스 연결 역할 지원을 시작할 때 이미 서비스를 사용하는 중이라면 계정의 새 역할에 대해 알려주는 이메일을 받게 될 수 있습니다. 이 경우 서비스에서 계정에 서비스 연결 역할을 자동으로 생성합니다. 이 역할을 지원하기 위해 어떤 작업도 수행할 필요가 없으며, 이 역할

을 수동으로 삭제할 수 없습니다. 자세한 내용은 내 AWS 계정에 표시되는 새 역할 단원을 참조하십시오.

서비스 연결 역할의 사용을 지원하는 서비스에 대한 자세한 정보는 AWS IAM으로 작업하는 서비스를 참조하고 서비스 연결 역할 열에 예가 있는 서비스를 찾으세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다. 자세한 내용은 서비스 연결 역할 생성 단원을 참조하십시오.

역할 함께 묶기

역할 체인은 역할을 사용하여 AWS CLI 또는 API를 통해 두 번째 역할을 수임하는 경우입니다. 예를 들어, `RoleA`에 `RoleB`를 수임할 권한이 있습니다. AssumeRole API 작업에서 User1의 장기 사용자 자격 증명을 사용하여 `RoleA`를 수임할 수 있습니다. 그러면 `RoleA` 단기 자격 증명이 반환됩니다. 역할 체인을 사용하면 `RoleA`의 단기 자격 증명을 사용하여 User1이 `RoleB`를 수임하도록 할 수 있습니다.

역할을 맡을 때 세션 태그를 전달하고 태그를 전이적으로 설정할 수 있습니다. 전이적 세션 태그는 역할 체인의 모든 후속 세션에 전달됩니다. 세션 태그에 대한 자세한 내용은 AWS STS에서 세션 태그 전달 섹션을 참조하세요.

역할 체인을 사용하면 AWS CLI 또는 AWS API 역할 세션이 최대 1시간으로 제한됩니다. AssumeRole API 작업을 사용하여 역할을 수임할 때 `DurationSeconds` 파라미터를 사용하여 역할 세션 길이를 지정할 수 있습니다. 역할에 대한 최대 세션 기간 설정에 따라 파라미터 값을 최대 43200초(12시간)까지 지정할 수 있습니다. 그러나 역할 함께 묶기를 사용해 역할을 수임하고 1시간보다 큰 `DurationSeconds` 파라미터 값을 지정하면 작업이 실패합니다.

AWS에서는 역할을 사용하여 EC2 인스턴스에서 실행되는 애플리케이션에 권한을 부여하는 것을 역할 함께 묶기로 간주하지 않습니다.

위임

제어하는 리소스에 대한 액세스를 허용하는 권한을 누군가에게 부여하는 것입니다. 위임은 두 계정 간에 신뢰를 설정하는 것을 포함합니다. 첫 번째는 리소스를 소유한 계정입니다(신뢰하는 계정). 두 번째는 리소스에 액세스해야 하는 사용자가 포함된 계정입니다(신뢰되는 계정). 신뢰받는 계정과 신뢰하는 계정은 다음 중 하나가 될 수 있습니다.

- 동일 계정
- 조직에서 통제하는 별도의 계정
- 서로 다른 조직이 소유한 2개의 계정

리소스에 액세스할 수 있는 권한을 위임하려면 두 개의 정책이 연결된 신뢰하는 계정에서 IAM 역할을 생성합니다. 권한 정책은 역할 사용자에게 리소스에 대해 의도한 작업을 수행하는 데 필요한 권한을 부여합니다. 신뢰 정책은 역할을 위임하도록 허용된 신뢰할 수 있는 계정 멤버를 지정합니다.

트러스트 정책을 생성할 때 와일드카드(*)를 주요 요소의 일부로 지정하고 ARN을 지정할 수 없습니다. 신뢰 정책은 신뢰하는 계정의 역할에 연결되어 있고 권한의 절반에 해당합니다. 나머지 절반은 사용자에게 역할 전환 또는 위임을 허용하는 신뢰받는 계정의 사용자에게 연결된 권한 정책입니다. 임시로 역할을 위임하는 사용자는 자신의 고유 권한을 포기하고 대신 해당 역할의 권한을 위임합니다. 사용자가 역할을 끝내거나 역할 사용을 중지하면 원래 사용자 권한이 자동으로 회복됩니다. 외부 ID라 불리는 부가적인 파라미터는 동일한 조직에 의해 제어되지 않는 계정 사이에서 역할을 안전하게 사용하도록 하는 데 도움이 됩니다.

신뢰 정책

역할을 수임하도록 신뢰하는 보안 주체를 정의하는 JSON 정책 문서입니다. 역할 신뢰 정책은 IAM의 역할에 연결된 필수 리소스 기반 정책입니다. 신뢰 정책에서 지정할 수 있는 보안 주체에는 사용자, 역할, 계정 및 서비스가 포함됩니다.

크로스 계정 액세스를 위한 역할

한 계정의 리소스에 대한 액세스 권한을 다른 계정의 신뢰할 수 있는 보안 주체에 부여하는 역할. 역할은 크로스 계정 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 제품을 사용하면 (역할을 프록시로 사용하는 대신) 리소스에 직접 정책을 연결할 수 있습니다. 이를 리소스 기반 정책이라고 하며, 이 정책을 사용하여 다른 AWS 계정의 보안 주체에게 리소스에 대한 액세스 권한을 부여할 수 있습니다. 이러한 리소스에는 Amazon Simple Storage Service(S3) 버킷, S3 Glacier 볼트, Amazon Simple Notification Service(SNS) 주제 및 Amazon Simple Queue Service(SQS) 대기열이 포함됩니다. 리소스 기반 정책을 지원하는 서비스에 대한 자세한 내용은 AWS IAM으로 작업하는 서비스 섹션을 참조하세요. 리소스 기반 정책에 대한 자세한 내용은 IAM의 크로스 계정 리소스 액세스 섹션을 참조하세요.

추가 리소스

다음 리소스는 IAM 역할과 관련된 IAM 용어를 자세히 알아보는 데 도움이 됩니다.

- 보안 주체란 작업을 수행하고 리소스에 액세스할 수 있는 AWS의 엔터티입니다. 보안 주체는 AWS 계정 루트 사용자, IAM 사용자 또는 역할입니다. AWS 서비스의 ID를 나타내는 보안 주체는 서비스 보안 주체입니다. 역할 신뢰 정책에서 보안 주체 요소를 사용하여 역할을 수임하도록 신뢰하는 보안 주체를 정의합니다.

역할을 수임하도록 허용할 수 있는 보안 주체에 대한 자세한 내용 및 예시는 AWS JSON 정책 요소: Principal 단원을 참조하세요.

- ID 제공업체는 외부 ID 제공업체와 AWS 사이에 신뢰 관계를 생성합니다. 기존 OpenID Connect(OIDC) 또는 Security Assertion Markup Language(SAML) 2.0 제공업체를 사용하여 AWS 리소스에 액세스할 수 있는 사용자를 관리할 수 있습니다. OIDC 및 SAML 2.0을 사용해 이러한 외부 ID 제공업체와 AWS 사이에 신뢰 관계를 구성하면 사용자가 IAM 역할에 할당됩니다. 사용자는 임시 보안 자격 증명을 부여받아 AWS 리소스에 대한 액세스가 가능합니다.

  페더레이션 사용자에 대한 자세한 내용은 자격 증명 공급자 및 페더레이션 섹션을 참조하세요.
- 페더레이션 사용자는 AWS Directory Service, 기업 사용자 디렉터리 또는 OIDC 제공업체의 기존 ID입니다. 이 사용자를 페더레이션 사용자라고 합니다. AWS에서는 자격 증명 공급자를 통해 액세스가 요청되면 페더레이션 사용자에게 역할을 할당합니다.

  페더레이션 사용자에 대한 자세한 내용은 연동 사용자 및 역할 섹션을 참조하세요.
- 권한 정책은 역할이 사용할 수 있는 작업 및 리소스를 정의하는 ID 기반 정책입니다. 이 문서는 IAM 정책 언어의 규칙에 따라 작성됩니다.

  자세한 내용은 IAM JSON 정책 참조 단원을 참조하십시오.
- 권한 경계는 ID 기반 정책이 역할에 부여할 수 있는 최대 권한을 제한하는 정책을 사용하는 고급 기능입니다. 서비스 연결 역할에 권한 경계를 적용할 수 없습니다.

  자세한 내용은 IAM 엔터티의 권한 범위 단원을 참조하십시오.

혼동된 대리자 문제

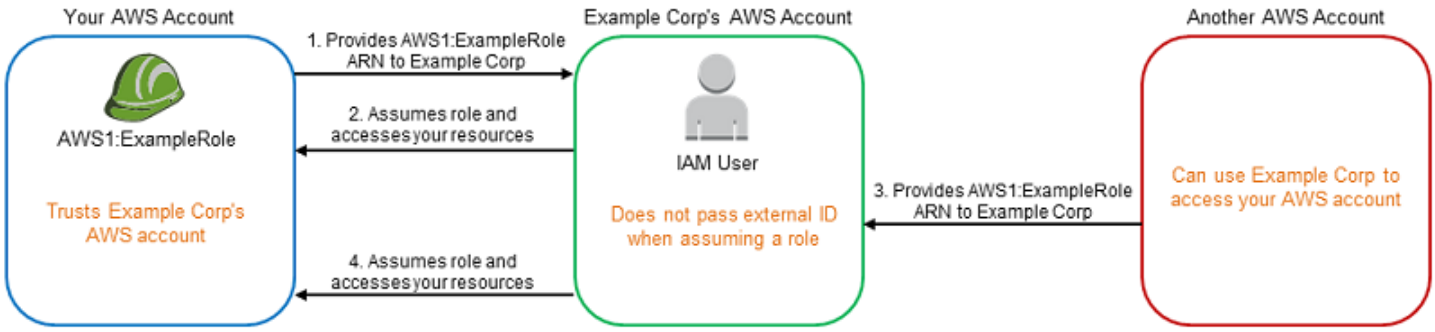
혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 이를 방지하기 위해 AWS은(는) 계정의 리소스에 타사(크로스 계정) 또는 기타 AWS 서비스(교차 서비스) 액세스를 제공할 경우 계정을 보호하는 데 도움이 되는 도구를 제공합니다.

이따금 AWS 리소스에 대한 액세스를 타사에 부여해야 할 때가 있습니다(액세스 위임). 예를 들어 Example Corp이라는 타사를 고용해 AWS 계정을 모니터링하고 비용을 최적화하기로 했다고 가정해 봅시다. 일일 경비를 추적하기 위해 Example Corp은 AWS 리소스에 접근해야 합니다. Example Corp 역시 다른 고객을 위해 다른 많은 AWS 계정을 모니터링합니다. IAM 역할을 사용하여 AWS 계정과 Example Corp 계정 사이에 신뢰 관계를 설정할 수 있습니다. 이 시나리오의 한 가지 중요한 부분은 IAM 역할 신뢰 정책에서 역할 수임자를 지정하는 데 사용할 수 있는 옵션 정보인 외부 ID입니다. 외부 ID의 주된 기능은 혼동된 대리자 문제를 해결하고 방지하는 것입니다.

AWS에서는 교차 서비스 가장(cross-service impersonation)으로 인해 혼동된 대리자 문제가 발생할 수 있습니다. 교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 직접적으로 호출할 때 발생할 수 있습니다. 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다.

크로스 계정 혼동된 대리자 예방

다음 다이어그램은 크로스 계정 혼동된 대리자 문제를 보여줍니다.



이 시나리오에서는 다음과 같이 가정합니다.

- AWS1은 사용자의 AWS 계정입니다.
- AWS1:ExampleRole은 계정의 역할입니다. 이 역할의 신뢰 정책은 Example Corp의 AWS 계정의 역할을 위임할 수 있는 것으로 지정함으로써 Example Corp을 신뢰합니다.

다음은 무슨 일이 일어나는지에 대한 것입니다.

1. Example Corp 서비스 사용을 시작할 때 Example Corp에 AWS1:ExampleRole의 ARN을 제공합니다.
2. Example Corp은 그 ARN을 사용해 임시 보안 인증을 얻어 AWS 계정의 리소스에 액세스합니다. 이러한 방식으로 Example Corp을 대신 행위할 수 있는 "대리자"로 신뢰합니다.
3. 또 다른 AWS 고객도 Example Corp의 서비스를 사용하기 시작하고, 이 고객 역시 Example Corp가 사용할 AWS1:ExampleRole의 ARN을 제공합니다. 아마도 그 다른 고객은 비밀이 아닌 AWS1:ExampleRole을 알거나 짐작했을 것입니다.
4. 다른 고객이 Example Corp에게(자신의 것이라고 주장하는) 계정의 AWS 리소스에 액세스할 수 있는 권한을 요청하면, Example Corp는 AWS1:ExampleRole을 사용해 계정의 리소스에 액세스합니다.

이것이 바로 다른 고객이 리소스에 무단으로 액세스하는 과정입니다. 이 고객은 Example Corp이 자신도 모르게 리소스에 대한 작업을 하도록 속일 수 있었기 때문에 Example Corp은 이제 "혼동된 대리자"가 되었습니다.

Example Corp는 사용자가 역할의 신뢰 정책에 `ExternalId` 조건 확인을 포함하도록 요구하여 혼동된 대리자 문제를 해결합니다. Example Corp는 각 고객에 대해 고유 `ExternalId` 값을 생성하며 요청에 해당 값을 사용하여 역할을 수임합니다. `ExternalId` 값은 Example Corp의 고객 사이에서 고유해야 하며 고객이 아닌 Example Corp에 의해 제어되어야 합니다. 이것이 바로 Example Corp에서 외부 ID를 얻고 그것을 스스로 찾아내지 않는 이유입니다. 이로 인해 Example Corp이 혼동된 대리자가 되는 것을 예방하고 다른 계정의 AWS 리소스에 대한 액세스 권한을 부여하는 것을 방지합니다.

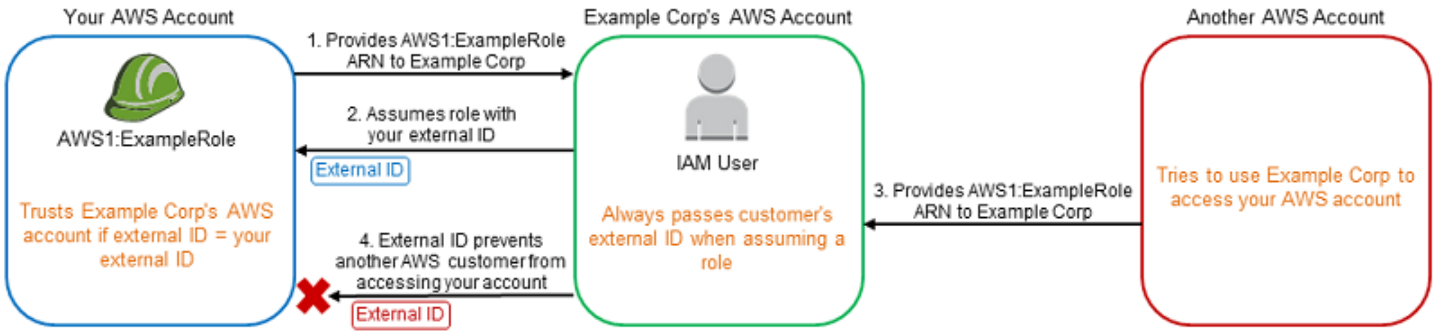
이 시나리오에서 Example Corp의 고유 식별자가 12345이고, 다른 고객에 대해서는 그 식별자가 67890이라고 가정합시다. 이러한 식별자는 이 시나리오를 위해 단순화된 것입니다. 일반적으로 이러한 식별자는 GUID입니다. 이 식별자가 Example Corp의 고객 사이에서 고유한 것이라고 가정할 때, 외부 ID를 위해 사용하기에 합리적인 값들입니다.

Example Corp은 12345라는 외부 ID 값을 부여합니다. 그런 다음 `Condition` 값이 12345가 되어야 한다고 요구하는 역할의 신뢰 정책에 `sts:ExternalId` 요소를 다음과 같이 추가해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "AWS": "Example Corp's AWS Account ID"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "sts:ExternalId": "12345"
      }
    }
  }
}
```

이 정책의 조건 요소는 AssumeRole API 호출에 12345라는 외부 ID 값이 포함될 때만 Example Corp이 역할을 수임하도록 허용합니다. Example Corp은 고객을 대신해 역할을 위임할 때마다 항상 AssumeRole 호출에 해당 고객의 외부 ID 값을 포함하도록 보장합니다. 다른 고객이 Example Corp에게 ARN을 공급한다 하더라도 Example Corp이 AWS에 대한 요청 시 포함하는 외부 ID를 제어할 수 없습니다. 이는 권한을 부여받지 않은 고객이 리소스에 액세스하지 못하도록 방지하는 데 도움이 됩니다.

다음 다이어그램에서 이 방법을 보여줍니다.



1. 전과 같이 Example Corp 서비스 사용을 시작할 때 Example Corp에 AWS1:ExampleRole의 ARN을 제공합니다.
2. Example Corp가 그 ARN을 사용해 AWS1:ExampleRole 역할을 위임하는 경우 Example Corp는 AssumeRole API 호출에 외부 ID(12345)를 포함시킵니다. 외부 ID는 역할의 신뢰 정책과 일치하므로 AssumeRole API 호출은 성공하고 Example Corp는 임시 보안 인증을 획득해 AWS 계정의 리소스에 액세스합니다.
3. 또 다른 AWS 고객도 Example Corp의 서비스를 사용하기 시작하고, 전과 같이 이 고객 역시 Example Corp가 사용할 AWS1:ExampleRole의 ARN을 제공합니다.
4. 그러나 이번에는 Example Corp가 AWS1:ExampleRole이라는 역할을 위임하려 할 때 다른 고객과 연결된 외부 ID(67890)를 제공하므로 해당 고객은 이를 바꿀 방법이 없습니다. Example Corp이 이렇게 하는 이유는 역할을 사용하겠다는 요청이 다른 고객에게서 왔으므로, 67890은 Example Corp이 작용하고 있는 상황을 나타내기 때문입니다. AWS1:ExampleRole의 신뢰 정책에 자신의 외부 ID(12345)가 있는 조건을 추가했기 때문에 AssumeRole API 호출은 실패하고 다른 고객이 계정 리소스에 무단으로 액세스하는 것을 막을 수 있습니다(다이어그램의 빨간색 "X" 참조).

외부 ID는 다른 고객이 Example Corp을 속여 자신도 모르게 리소스에 액세스하지 못하도록 방지합니다.

교차 서비스 혼동된 대리인 방지

서비스 권한을 특정 리소스로 제한하기 위해 리소스 기반 정책의 `aws:SourceArn`, `aws:SourceAccount`, `aws:SourceOrgID`, 또는 `aws:SourceOrgPaths` 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다. `aws:SourceArn`을 사용하면 하나의 리소스만 교차 서비스 액세스 권한과 연결됩니다. `aws:SourceAccount`를 사용하면 해당 계정의 모든 리소스가 교차 서비스 사용 권한과 연결됩니다. `aws:SourceOrgID`를 사용하면 조직 내 모든 계정의 모든 리소스가 교차 서비스 사용 권한과 연결될 수 있습니다. `aws:SourceOrgPaths`를 사용하면 AWS Organizations 경로 내 계정

의 모든 리소스가 교차 서비스 사용 권한과 연결됩니다. 경로 사용 및 이해에 대한 자세한 내용은 AWS Organizations 엔터티 경로 이해하기 섹션을 참조하세요.

혼동된 대리자 문제로부터 보호하는 가장 좋은 방법은 리소스 기반 정책에서 리소스의 전체 ARN이 포함된 `aws:SourceArn` 전역 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모를 경우 또는 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드(*)를 포함한 `aws:SourceArn` 전역 조건 컨텍스트 키를 사용합니다. 예: `arn:aws:servicename:*:123456789012:*`

만약 `aws:SourceArn` 값에 Amazon S3 버킷 ARN과 같은 계정 ID가 포함되어 있지 않은 경우, 권한을 제한하려면 두 `aws:SourceAccount` 및 `aws:SourceArn`를 모두 사용해야 합니다.

혼동된 대리자 문제로부터 보호하려면 리소스 기반 정책에서 리소스의 조직 ID 또는 조직 경로와 함께 `aws:SourceOrgID` 또는 `aws:SourceOrgPaths` 전역 조건 컨텍스트 키를 사용하세요. `aws:SourceOrgID` 또는 `aws:SourceOrgPaths` 키가 포함된 정책에는 올바른 계정이 자동으로 포함되며 조직에서 계정을 추가, 제거 또는 이동할 때 정책을 수동으로 업데이트할 필요가 없습니다.

서비스에 연결되지 않은 역할 트러스트 정책의 경우, 트러스트 정책의 모든 서비스는 역할이 호출하는 서비스와 동일한 계정에 있는지 확인하는 `iam:PassRole` 작업을 수행했습니다. 따라서 트러스트 정책과 `aws:SourceAccount`, `aws:SourceOrgID`, 또는 `aws:SourceOrgPaths` 은(는) 함께 사용하지 않아도 됩니다. 트러스트 정책에서 `aws:SourceArn`을(를) 사용하면 Lambda 함수 ARN과 같이 역할을 대신 맡을 수 있는 리소스를 지정할 수 있습니다. 일부 AWS 서비스는 새로 만든 역할을 위한 트러스트 정책 `aws:SourceAccount` 및 `aws:SourceArn`을(를) 사용하지만 계정의 기존 역할에 대해서는 키를 사용하지 않아도 됩니다.

> **Note**
>
> KMS 키 부여 사용과 AWS Key Management Service을 통합하는 AWS 서비스은 `aws:SourceArn`, `aws:SourceAccount`, `aws:SourceOrgID`, 또는 `aws:SourceOrgPaths` 조건 키를 지원하지 않습니다. KMS 키 정책에서 이러한 조건 키를 사용하면 KMS 키 부여를 통해서 AWS 서비스가 키를 사용하는 경우 예상치 못한 동작이 발생할 수 있습니다.

AWS Security Token Service에 대한 교차 서비스 혼동된 대리자 예방

AWS 서비스는 역할을 사용하여 서비스가 사용자를 대신하여 다른 서비스의 리소스로 액세스할 수 있어야 합니다. 서비스가 사용자를 대신하여 작업을 수행하기 위해 수임한 역할을 서비스 역할이라고 합니다. 역할에는 두 가지 정책이 필요합니다. 즉, 역할을 수임할 수 있는 보안 주체를 지정하는 역할 신뢰

정책과 역할로 수행할 수 있는 작업을 지정하는 권한 정책이 필요합니다. 역할 신뢰 정책은 IAM의 유일한 리소스 기반 정책 유형입니다. 기타 AWS 서비스에는 Amazon S3 버킷 정책과 같은 리소스 기반 정책이 있습니다.

서비스가 사용자를 대신하여 역할을 맡을 경우 서비스 보안 주체는 역할 신뢰 정책의 `sts:AssumeRole` 작업을 수행하도록 허용되어야 합니다. 서비스가 `sts:AssumeRole`을(를) 호출할 때, AWS STS은(는) 서비스 보안 주체가 역할의 권한 정책에 허용되는 리소스에 액세스하는 데 사용하는 일련의 임시 보안 자격 증명을 반환합니다. 서비스가 계정에서 역할을 맡은 경우, 역할 신뢰 정책의 `aws:SourceArn`, `aws:SourceAccount`, `aws:SourceOrgID`, 또는 `aws:SourceOrgPaths` 전역 조건 컨텍스트 키를 포함하여 역할에 대한 액세스를 예상 리소스에 의해 생성된 요청으로만 제한할 수 있습니다.

예를 들어, AWS Systems Manager Incident Manager에서는 Incident Manager가 사용자를 대신하여 Systems Manager 자동화 문서를 실행할 수 있는 역할을 선택해야 합니다. 자동화 문서에는 CloudWatch 경보 또는 EventBridge 이벤트에 의해 시작된 인시던트에 대한 자동 응답 계획이 포함될 수 있습니다. 다음 역할 신뢰 정책 예에서는 `aws:SourceArn` 조건 키를 사용하여 인시던트 레코드의 ARN을 기반으로 서비스 역할에 대한 액세스를 제한할 수 있습니다. 응답 계획 리소스 `myresponseplan`에서 생성된 인시던트 레코드만이 이 역할을 사용할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "Service": "ssm-incidents.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:ssm-incidents:*:111122223333:incident-
record/myresponseplan/*"
      }
    }
  }
}
```

> **Note**
>
> 모든 서비스가 AWS STS 지원 `aws:SourceArn`, `aws:SourceAccount`, `aws:SourceOrgID`, 또는 `aws:SourceOrgPaths` 조건 키와 통합되는 것은 아닙니다. 지원

되지 않는 통합과 함께 IAM 신뢰 정책에서 이러한 키를 사용하면 예상치 못한 동작이 발생할 수 있습니다.

IAM 역할 관련 일반 시나리오

대부분의 AWS 기능과 마찬가지로 역할 사용에는 일반적으로 두 가지 방법이 있습니다. 즉, IAM 콘솔에서 대화식으로 사용하는 것 또는 AWS CLI, Tools for Windows PowerShell이나 API에서 프로그래밍 방식으로 사용하는 것입니다.

- IAM 콘솔을 사용하는 계정의 IAM 사용자가 역할을 전환하여 콘솔에서 해당 역할의 권한을 임시로 사용할 수 있습니다. 사용자는 자신의 원래 권한을 포기하고 역할에 할당된 권한을 수임합니다. 사용자가 역할을 끝내면 원래 권한이 복원됩니다.
- AWS에서 제공하는 애플리케이션 또는 서비스(예: Amazon EC2)가 AWS에 프로그래밍 방식으로 요청할 때 사용할 역할의 임시 보안 자격 증명을 요청하여 역할을 수임할 수 있습니다. 역할을 이렇게 사용하면 리소스에 액세스해야 하는 각 엔터티를 위한 장기 보안 자격 증명을 공유하거나 유지(예: IAM 사용자 생성)할 필요가 없습니다.

> **Note**
>
> 이 안내서는 역할로 전환합니다와 역할을 수임합니다라는 표현을 서로 대치할 수 있는 동일한 의미로 사용합니다.

역할을 사용하는 가장 간단한 방법은 IAM 사용자에게 자신이나 다른 사용자의 AWS 계정에서 생성하는 역할로 전환할 권한을 부여하는 것입니다. IAM 사용자는 IAM 콘솔을 통해 역할을 쉽게 전환하여 일반적으로 부여받지 않은 권한을 사용한 다음 역할을 끝내 해당 권한을 포기할 수 있습니다. 이를 통해 중요한 리소스에 잘못 액세스하거나 이를 수정하는 일을 방지할 수 있습니다.

애플리케이션 및 서비스 또는 연동된 외부 사용자에게 액세스 권한을 부여하는 등 역할을 한층 복잡한 방식으로 사용하기 위해 `AssumeRole` API를 호출할 수 있습니다. 이 API 호출은 애플리케이션이 이후의 API 호출에 사용할 수 있는 일련의 임시 자격 증명 세트를 반환합니다. 임시 자격 증명을 사용하여 시도하는 작업에는 연결된 역할에서 부여한 권한만 있습니다. 애플리케이션에서는 콘솔에서 사용자가 하듯이 역할을 "끝낼" 필요가 없습니다. 단지 애플리케이션에서 임시 자격 증명 사용을 중지하고 원래 자격 증명으로 호출을 재개합니다.

페더레이션 사용자는 IdP(자격 증명 공급자)에서 제공하는 자격 증명을 사용하여 로그인합니다. 그 다음 AWS에서 신뢰받는 IdP에 임시 자격 증명을 제공하여 이후의 AWS 리소스 요청에 포함할 수 있도록 사용자에게 전달합니다. 그러한 자격 증명은 할당된 역할에 부여된 권한을 제공합니다.

이 섹션에서는 다음 시나리오의 개요를 제공합니다.

- 소유한 AWS 계정의 IAM 사용자에게 소유한 다른 계정의 리소스에 액세스할 수 있는 권한 제공
- AWS 워크로드 이외 워크로드에 대한 액세스 권한 제공
- 타사가 소유한 AWS 계정에 속한 IAM 사용자에 액세스 권한 제공
- AWS가 제공하는 서비스를 위해 AWS 리소스에 대한 액세스 권한을 제공하는 경우
- 외부에서 인증된 사용자에게 액세스 권한 제공(아이덴티티 페더레이션)

소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스

IAM 사용자에게 AWS 계정 내의 역할 또는 소유하고 있는 다른 AWS 계정에 정의된 역할로 전환할 수 있는 권한을 부여할 수 있습니다.

> **Note**
>
> 소유하지 않은 또는 제어하지 않는 계정에 대한 액세스 권한을 부여하고자 하는 경우, 이 주제 뒷부분의 타사가 소유한 AWS 계정에 액세스 섹션을 참조하세요.

조직에 중요한 Amazon EC2 인스턴스가 있다고 가정해 봅시다. 사용자에게 인스턴스를 종료할 수 있는 권한을 직접 부여하지 않고, 이러한 권한이 있는 역할을 만들 수 있습니다. 그런 다음 관리자는 인스턴스를 종료해야 하는 경우 해당 역할로 전환할 수 있습니다. 그러면 이러한 인스턴스에 다음과 같은 보호 계층이 추가됩니다.

- 사용자에게 역할을 수임할 권한을 명시적으로 부여해야 합니다.
- 사용자는 AWS Management Console을 사용하여 해당 역할로 능동적으로 전환하거나 AWS CLI 또는 AWS API를 사용하여 역할을 수임해야 합니다.
- 역할에 멀티 팩터 인증(MFA) 보호를 추가하여 MFA 디바이스로 로그인하는 사용자만 역할을 수임할 수 있도록 합니다. 역할을 수임한 사용자가 MFA(멀티 팩터 인증)를 사용하여 처음에 인증을 받도록 역할을 구성하는 방법을 알아보려면 MFA를 통한 보안 API 액세스 섹션을 참조하세요.

이 방법을 사용하여 최소 권한 원칙을 적용하는 것이 좋습니다. 다시 말해, 특정 작업이 필요한 경우에만 승격된 권한을 사용하도록 제한하는 것입니다. 역할을 사용하여 중요한 환경을 실수로 변경하는 일

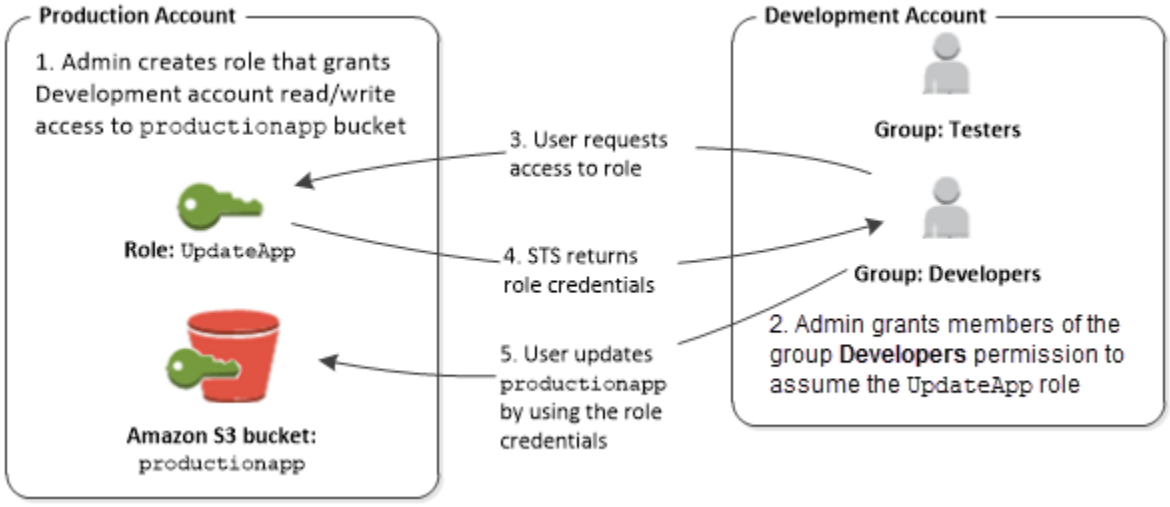
을 방지할 수 있습니다. 특히 필요할 때만 역할이 사용되는지 확인하기 위해 중요한 환경을 감사와 결합하는 경우에 그렇습니다.

이러한 목적을 위해 역할을 만들려면 해당 역할의 신뢰 정책 `Principal` 요소에서 액세스가 필요한 사용자의 ID로 계정을 지정합니다. 그러면 그러한 다른 계정의 특정 사용자에게 해당 역할로 전환할 수 있는 권한을 부여할 수 있습니다. 해당 신뢰 영역(신뢰할 수 있는 조직 또는 계정) 외의 계정 내 보안 주체가 역할을 수임하는 권한이 있는지 자세히 알고 싶다면, IAM Access Analyzer란 무엇인가요?를 참조하세요.

한 계정의 사용자는 동일한 또는 다른 계정의 역할로 전환할 수 있습니다. 사용자는 역할을 사용하는 동안 해당 작업만을 수행하고 해당 역할에서 허용한 리소스만 액세스할 수 있지만, 이들의 원래 사용자 권한은 일시 중지된 상태입니다. 사용자가 역할을 끝내면 원래 사용자 권한이 회복됩니다.

분리된 개발 및 프로덕션 계정을 사용한 예제 시나리오

프로덕션 환경에서 개발 환경을 격리하기 위해 조직이 여러 개의 AWS 계정을 갖고 있다고 가정합시다. 개발 계정의 사용자는 프로덕션 계정의 리소스에 액세스해야 하는 경우가 있습니다. 예를 들어, 개발 환경에서 프로덕션 환경으로 업데이트를 승격하려는 경우 크로스 계정 액세스 권한이 필요할 수 있습니다. 두 계정을 모두 사용하는 사용자를 위해 별도의 자격 증명(및 암호)을 생성했다 해도 여러 계정에 대한 자격 증명을 관리할 경우 자격 증명 관리가 어려워집니다. 다음 그림을 보면 모든 사용자가 개발 계정에서 관리됩니다. 그러나 일부 개발자에게는 프로덕션 계정에 대한 제한된 액세스 권한이 필요합니다. 개발 계정에는 Testers와 Developers라는 두 개의 그룹이 있으며 각 그룹에는 고유의 정책이 있습니다.



1. 프로덕션 계정에서 관리자는 IAM을 사용하여 해당 계정에 `UpdateApp` 역할을 생성합니다. 관리자는 그 역할에서 개발 계정을 `Principal`로 지정하는 신뢰 정책을 정의합니다. 이는 개발 계정의 권

한이 있는 사용자는 UpdateApp 역할을 사용할 수 있다는 것을 뜻합니다. 또한 관리자는 이 역할의 사용자가 productionapp이라는 Amazon S3 버킷에 대한 읽기 및 쓰기 권한을 보유하도록 지정하는 역할의 권한 정책을 정의합니다.

그런 다음 관리자는 적절한 정보를 이 역할을 수임해야 하는 대상과 공유합니다. 그러한 정보로는 계정 번호와 역할 이름(AWS 콘솔 사용자들에 대한) 또는 Amazon 리소스 이름(ARN)(AWS CLI 또는 AWS API 액세스용)이 있습니다. 역할 ARN은 arn:aws:iam::123456789012:role/UpdateApp과 같은 형태를 띕니다. 여기에서 역할의 이름은 UpdateApp이고, 역할이 생성된 계정 번호는 123456789012입니다.

> **Note**
>
> 관리자는 역할을 수임하는 사용자가 먼저 멀티 팩터 인증(MFA)을 사용하여 인증을 받도록 역할을 구성할 수도 있습니다. 자세한 내용은 MFA를 통한 보안 API 액세스 단원을 참조하십시오.

2. 개발 계정에서 관리자는 Developer 그룹의 구성원에게 이 역할로 전환할 수 있는 권한을 부여합니다. 이를 수행하려면 Developers 그룹에 UpdateApp 역할에 대한 AWS Security Token Service(AWS STS) AssumeRole API를 호출할 권한을 부여하면 됩니다. 이제 개발 계정의 Developers 그룹에 속한 모든 IAM 사용자는 프로덕션 계정의 UpdateApp 역할로 전환할 수 있습니다. Developer 그룹에 속하지 않은 다른 사용자는 이 역할로 전환할 수 있는 권한이 없으므로 프로덕션 계정의 S3 버킷에 액세스할 수 없습니다.
3. 사용자가 이 역할로의 전환을 요청:
   - AWS 콘솔: 탐색 표시줄에서 계정 이름을 선택하고 Switch Role(역할 전환)을 선택합니다. 계정 ID(또는 별칭) 및 역할 이름을 지정합니다. 아니면 사용자는 관리자가 이메일로 보낸 링크를 클릭해도 됩니다. 링크를 누르면 세부 정보가 이미 채워져 있는 역할 전환 페이지로 이동합니다.
   - AWS API/AWS CLI: 개발 계정의 Developers 그룹에 속한 사용자는 AssumeRole 함수를 호출하여 UpdateApp 역할에 대한 자격 증명을 가져옵니다. UpdateApp 역할의 ARN을 이 호출의 일부로 지정합니다. Testers 그룹의 사용자가 동일한 요청을 하는 경우에는 요청이 실패하는데, 이는 Testers가 AssumeRole 역할 ARN을 위해 UpdateApp을 호출할 권한이 없기 때문입니다.
4. AWS STS는 임시 자격 증명을 반환합니다.
   - AWS 콘솔: AWS STS에서 그 요청이 신뢰할 수 있는 대상(개발 계정)에서 온 것인지 확인하기 위해 그 요청에 대해 역할의 신뢰 정책을 확인합니다. 확인 후 AWS STS에서 AWS 콘솔로 임시 보안 자격 증명을 반환합니다.

- API/CLI: AWS STS에서 신뢰할 수 있는 대상(Development 계정)이 요청을 보낸 것인지 확인하기 위해 역할의 신뢰 정책에 대한 요청을 확인합니다. 확인 후 AWS STS에서 해당 애플리케이션으로 임시 보안 자격 증명을 반환합니다.

5. 임시 자격 증명은 AWS 리소스에 대한 액세스를 허용합니다.
   - AWS 콘솔: AWS 콘솔은 이후의 모든 콘솔 작업에서 사용자를 대신하여 임시 자격 증명을 사용합니다. 이 경우에 그 작업이란 `productionapp` 버킷에 대한 읽기 및 쓰기입니다. 이 콘솔은 프로덕션 계정의 다른 리소스에는 액세스할 수 없습니다. 사용자가 역할을 끝내면 사용자의 권한은 이 역할로 전환하기 전에 보유한 원래의 권한으로 돌아갑니다.
   - API/CLI: 이 애플리케이션에서는 임시 보안 자격 증명을 사용하여 `productionapp` 버킷을 업데이트합니다. 이 애플리케이션은 임시 보안 자격 증명을 통해 `productionapp` 버킷에 대한 읽기 및 쓰기만 할 수 있으며 프로덕션 계정의 다른 리소스에는 액세스할 수 없습니다. 애플리케이션은 역할을 종료하지 않아도 되지만 대신에 임시 자격 증명 사용을 중지하고 이후의 API 호출에서 다시 원래의 자격 증명을 사용합니다.

추가 리소스

자세한 내용은 다음을 참조하세요.

- 튜토리얼: IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임

비 AWS 워크로드 액세스

IAM 역할은 권한이 할당된 AWS Identity and Access Management(IAM)의 객체입니다. IAM 자격 증명 또는 AWS 외부의 자격 증명을 사용하여 역할을 맡은 경우 해당 역할 세션을 위한 임시 보안 자격 증명이 제공됩니다. AWS 리소스에 액세스해야 하는 데이터 센터 또는 AWS 외부의 기타 인프라에서 실행 중인 워크로드가 있을 수 있습니다. 장기 액세스 키를 생성, 배포 및 관리하는 대신 AWS Identity and Access Management Roles Anywhere(IAM Roles Anywhere)를 사용하여 AWS 이외 워크로드를 인증할 수 있습니다. IAM Roles Anywhere는 인증 기관(CA)의 X.509 인증서를 사용하여 자격 증명을 인증하고 IAM 역할에서 제공하는 임시 보안 인증을 사용하여 AWS 서비스에 대한 액세스 권한을 안전하게 제공합니다.

IAM Roles Anywhere 사용

1. AWS Private Certificate Authority 사용을 통해 CA를 설정하거나 자체 PKI 인프라의 CA를 사용합니다.
2. CA를 설정한 후 IAM Roles Anywhere에서 트러스트 앵커라는 객체를 생성합니다. 이 앵커는 IAM Roles Anywhere와 CA 간에 인증을 위한 트러스트를 구축합니다.

3. 그런 다음 기존 IAM 역할을 구성하거나 IAM Roles Anywhere 서비스를 신뢰하는 새 역할을 생성할 수 있습니다.
4. 트러스트 앵커를 사용하여 비 AWS 워크로드를 IAM Roles Anywhere에서 인증합니다. AWS에서는 IAM 역할에 비 AWS 워크로드 이미 자격 증명을 부여하고, 이를 통해 AWS 리소스에 액세스할 수 있습니다.

추가 리소스

다음 리소스는 비 AWS 워크로드에 대한 액세스 제공에 대해 알아보는 데 도움이 될 수 있습니다.

- IAM Roles Anywhere 구성에 대한 자세한 내용은 IAM Roles Anywhere 사용 설명서의 AWS Identity and Access Management Roles Anywhere란 무엇입니까?를 참조하세요.
- IAM Roles Anywhere에서 퍼블릭 키 인프라(PKI)를 설정하는 방법을 알아보려면 AWS 보안 블로그에서 IAM Roles Anywhere with an external certificate authority의 내용을 참조하세요.

타사가 소유한 AWS 계정에 액세스

타사가 조직의 AWS 리소스에 액세스해야 하는 경우 역할을 사용하여 해당 사용자에게 그에 대한 액세스 권한을 위임할 수 있습니다. 예를 들어, 타사가 AWS 리소스를 관리하는 서비스를 제공할 경우 IAM 역할을 사용하면 AWS 보안 자격 증명을 공유하지 않고 해당 타사에 AWS 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 대신 타사는 귀하의 AWS 계정에 만든 역할로 가장하여 귀하의 AWS 리소스에 액세스할 수 있습니다. 해당 신뢰 영역(신뢰할 수 있는 조직 또는 계정) 외의 계정 내 보안 주체가 역할을 수임하는 권한이 있는지 자세히 알고 싶다면 IAM Access Analyzer란 무엇인가요?를 참조하세요.

해당 사용자가 수임할 수 있는 역할을 생성하려면 타사가 다음 정보를 제공해야 합니다.

- 타사의 AWS 계정 ID. 역할에 대한 신뢰 정책을 정의할 때 AWS 계정 ID를 보안 주체로 지정합니다.
- 역할을 고유하게 연결하는 데 사용하는 외부 ID. 외부 ID는 여러분과 타사만이 알고 있는 임의의 식별자일 수 있습니다. 예를 들어, 여러분과 타사가 사용하는 인보이스 ID를 사용할 수 있지만 타사의 이름이나 전화번호와 같이 추측 가능한 것은 사용하지 마세요. 역할에 대한 신뢰 정책을 정의할 때 이 ID를 지정해야 합니다. 타사가 역할을 수임할 때 이 ID를 제공해야 합니다.
- AWS 리소스를 사용하기 위해 타사에 필요한 권한. 역할의 권한 정책을 정의할 때 이러한 권한을 지정해야 합니다. 이 정책은 타사에서 수행할 수 있는 작업과 액세스할 수 있는 리소스를 정의합니다.

역할을 정의한 후에는 역할의 Amazon 리소스 이름(ARN)을 타사에 제공해야 합니다. 타사가 역할을 수임하려면 해당 역할의 ARN이 필요합니다.

> **Important**
>
> 타사에 AWS 리소스에 대한 액세스 권한을 부여하는 경우 타사는 여러분이 정책에서 지정하는 모든 리소스에 액세스할 수 있습니다. 타사의 리소스 사용에 대해서는 여러분에게 과금됩니다. 타사의 리소스 사용을 적절하게 제한해야 합니다.

타사 액세스를 위한 외부 ID

외부 ID는 그 역할을 위임하고 있는 사용자가 자신이 활동하고 있는 상황을 어설션할 수 있도록 허용합니다. 또한, 계정 소유자가 특정 상황에서만 역할이 위임되도록 허용할 수 있는 방법을 제공합니다. 외부 ID의 주된 기능은 혼동된 대리자 문제를 해결하고 방지하는 것입니다.

> **Important**
>
> AWS는 외부 ID를 비밀로 취급하지 않습니다. AWS에서 액세스 키 페어 또는 암호와 같은 비밀 정보를 만든 후에는 다시 볼 수 없습니다. 역할의 외부 ID는 해당 역할을 볼 수 있는 권한을 가진 사람만 볼 수 있습니다.

언제 외부 ID를 사용해야 하나요?

다음 상황에서 외부 ID를 사용합니다.

- AWS 계정 소유자가 자신의 계정뿐 아니라 다른 AWS 계정에도 액세스하는 타사를 위한 역할을 구성했습니다. 이 경우 타사에 역할을 위임할 때 포함하는 외부 ID를 요청해야 합니다. 그런 다음 역할의 신뢰 정책에서 외부 ID를 확인합니다. 이렇게 하여 외부 사용자를 대신해서 수행하는 경우에만 역할을 맡을 수 있도록 해야 합니다.
- 이전 시나리오의 Example Corp와 같은 다른 고객을 대신하여 역할을 위임할 수 있습니다. 각 고객에게 고유한 외부 ID를 할당하고 외부 ID를 역할의 신뢰 정책에 추가하도록 지시해야 합니다. 그런 다음 역할 위임 요청에 정확한 외부 ID를 항상 포함하도록 해야 합니다.

  각 고객에 대한 고유한 식별자를 이미 갖고 있겠지만, 이 고유 ID는 외부 ID로 사용하기에 충분합니다. 외부 ID는 단지 이러한 목적을 위해 명시적으로 생성하거나 별도로 추적할 필요가 있는 특별한 값은 아닙니다.

  외부 ID는 항상 `AssumeRole` API 호출에 지정해야 합니다. 이 밖에도 고객이 역할 ARN을 부여할 때 정확한 외부 ID가 있든 없든 그 역할을 위임할 수 있는지 확인하세요. 정확한 외부 ID 없이 역할을 위임할 수 있는 경우 시스템에 고객의 역할 ARN을 저장하지 마세요. 고객이 정확한 외부 ID를 요구하

도록 역할 신뢰 정책을 업데이트할 때까지 기다립니다. 이러한 방식으로 고객이 올바른 일을 할 수 있도록 돕고, 이는 양자 모두 혼동된 대리자 문제에서 보호받는 데 도움이 됩니다.

외부 ID를 사용하는 예제 시나리오

예를 들어 Example Corp이라는 타사를 고용해 AWS 계정을 모니터링하고 비용을 최적화하기로 했다고 가정해봅시다. 일일 경비를 추적하기 위해 Example Corp은 AWS 리소스에 접근해야 합니다. Example Corp 역시 다른 고객을 위해 다른 많은 AWS 계정을 모니터링합니다.

AWS 계정의 IAM 사용자 및 해당 장기 자격 증명에 대한 액세스 권한을 Example Corp에게 제공하지 마세요. 대신 IAM 역할과 임시 보안 자격 증명을 사용합니다. IAM 역할은 장기 보안 인증(IAM 사용자 액세스 키 등)을 공유하지 않고도 AWS 리소스에 액세스할 수 있도록 허용하는 메커니즘을 타사에 제공합니다.

IAM 역할을 사용하여 AWS 계정과 Example Corp 계정 사이에 신뢰 관계를 설정할 수 있습니다. 이 관계가 설정된 후 Example Corp 계정의 멤버는 AWS Security Token Service AssumeRole API를 호출하여 임시 보안 자격 증명을 얻을 수 있습니다. Example Corp 멤버는 자격 증명을 사용하여 계정의 AWS 리소스에 액세스할 수 있습니다.

> **Note**
>
> 임시 보안 자격 증명을 얻기 위해 호출할 수 있는 AssumeRole 및 다른 AWS API 작업에 대한 자세한 내용은 다음(임시 보안 자격 증명 요청)을 참조하세요.

이 시나리오에 대한 더 자세한 분석은 다음과 같습니다.

1. Example Corp을 고용해 고유한 사용자 지정 식별자를 생성하도록 합니다. 이 고유 고객 ID와 AWS 계정 번호를 제공합니다. 이 정보는 다음 단계에서 IAM 역할을 생성하는 데 필요합니다.

   > **Note**
   >
   > 이 식별자가 Example Corp의 각 고객에게 고유한 것이라면 Example Corp는 ExternalId에 대해 그들이 원하는 어떤 문자열 값이라도 사용할 수 있습니다. 두 고객이 같은 값을 갖지 않는 한, 고객 계정 번호 또는 임의 문자열이 될 수 있습니다. 이는 '보안 유지'를 위한 것은 아닙니다. Example Corp은 각 고객에게 ExternalId 값을 제공해야 합니다. 가장 중요한 것은 각 외부 ID가 고유하도록, 고객이 아닌 Example Corp이 이 값을 생성해야 한다는 것입니다.

2. AWS에 로그인해 Example Corp에 리소스에 대한 액세스 권한을 부여하는 IAM 역할을 생성합니다. IAM 역할과 마찬가지로 해당 역할에도 권한 정책과 신뢰 정책이라는 두 가지 정책이 있습니다. 그 역할의 신뢰 정책은 역할을 위임할 사용자를 지정합니다. 이 예시 시나리오에서 정책은 Example Corp의 AWS 계정 번호를 `Principal`로 지정합니다. 이렇게 하면 계정의 자격 증명이 그 역할을 수임하도록 허용합니다. 또한, `Condition` 요소를 신뢰 정책에 추가합니다. 이 `Condition`은 Example Corp의 고유 고객 ID와 일치하는지 확인하기 위해 `ExternalId` 컨텍스트 키를 테스트합니다.

```
   "Principal": {"AWS": "Example Corp's AWS ## ID"},
   "Condition": {"StringEquals": {"sts:ExternalId": "Unique ID Assigned by Example
Corp"}}
```

3. 역할에 대한 권한 정책은 해당 역할이 누군가가 수행하도록 허용할 수 있는 작업을 지정합니다. 예를 들어 그 역할은 누군가에게 IAM 사용자나 그룹이 아닌 Amazon EC2 또는 Amazon RDS 리소스만을 관리할 수 있게 허용하도록 지정할 수 있습니다. 이 예시 시나리오에서는 권한 정책을 사용하여 Example Corp에게 계정의 리소스 전체에 대한 읽기 전용 액세스 권한을 부여합니다.
4. 역할을 정의한 후에는 역할의 Amazon 리소스 이름(ARN)을 Example Corp에 제공합니다.
5. Example Corp가 AWS 리소스에 액세스해야 할 때는 그 회사의 누군가가 AWS `sts:AssumeRole` API를 호출합니다. 이 호출에는 수임할 역할의 ARN과 사용자 지정 ID에 해당하는 ExternalId 파라미터가 포함되어 있습니다.

Example Corp의 AWS 계정을 사용하는 사람이 요청하는 경우와 역할 ARN 및 외부 ID가 올바른 경우에 요청이 성공합니다. 그 경우 요청은 역할이 허용하는 AWS 리소스에 액세스하기 위해 Example Corp이 사용할 수 있는 임시 보안 자격 증명을 제공합니다.

다시 말해서 역할 정책에 외부 ID가 포함된다면 그 역할을 수임하고자 하는 사용자는 누구든지 그 역할에서 보안 주체로 지정되어야 하고 정확한 외부 ID를 포함해야 합니다.

외부 ID의 요점

- 서로 다른 AWS 계정을 가진 여러 고객을 지원하는 다중 테넌트 환경에서는 AWS 계정당 하나의 외부 ID를 사용하는 것이 좋습니다. 이 ID는 타사에서 생성한 임의의 문자열이어야 합니다.
- 역할을 수임할 때 타사에서 외부 ID를 제공하도록 요구하려면 해당 역할의 신뢰 정책을 선택한 외부 ID로 업데이트합니다.
- 역할을 수임할 때 외부 ID를 제공하려면 AWS CLI 또는 AWS API를 사용하여 해당 역할을 수임합니다. 자세한 내용은 STS AssumeRole API 작업 또는 STS assume-role CLI 작업을 참조하세요.

추가 리소스

다음 리소스는 타사 소유의 AWS 계정에 대한 액세스 제공에 대해 알아보는 데 도움이 될 수 있습니다.

- 다른 사용자가 AWS 계정에서 작업을 수행하도록 허용하는 방법을 알아보려면 사용자 지정 트러스트 정책을 사용하여 역할 생성 의 내용을 참조하세요.
- 역할로 전환할 수 있는 권한을 부여하는 방법을 알아보려면 사용자에게 역할을 전환할 권한 부여의 내용을 참조하세요.
- 신뢰할 수 있는 사용자를 생성하고 임시 보안 자격 증명을 제공하는 방법을 알아보려면 사용자 임시 보안 자격 증명에 대한 권한 제어의 내용을 참조하세요.

AWS 서비스 액세스

많은 AWS 서비스에서는 역할을 사용하여 해당 서비스가 액세스할 수 있는 대상을 제어해야 합니다. 서비스가 사용자를 대신하여 작업을 수행하기 위해 수임한 역할을 서비스 역할이라고 합니다. 역할이 서비스에 대해 특수한 목적을 수행하는 경우 서비스 연결 역할로 분류될 수 있습니다. 서비스에서 역할을 사용하는지 여부와 서비스에서 사용할 역할을 할당하는 방법을 알아보려면 서비스별 AWS 문서를 참조하세요.

역할을 생성해 AWS가 제공하는 서비스에 액세스 권한을 위임하는 것에 대한 자세한 내용은 AWS 서비스에 대한 권한을 위임할 역할 생성 섹션을 참조하세요.

외부에서 인증된 사용자에 대한 액세스(ID 페더레이션)

사용자는 이미 회사 디렉터리 등 AWS 외부에 자격 증명을 보유할 수 있습니다. 그러한 사용자가 AWS 리소스를 사용해야 하는 경우(또는 해당 리소스에 액세스하는 애플리케이션을 사용해야 하는 경우), 해당 사용자에게도 AWS 보안 자격 증명이 필요합니다. 자격 증명이 내 조직 또는 서드 파티 자격 증명 공급자(IdP)에서 페더레이션되는 사용자의 권한을 IAM 역할을 사용하여 지정할 수 있습니다.

> **Note**
>
> 보안 모범 사례로서, IAM 사용자를 생성하지 말고, 그 대신 아이덴티티 페더레이션을 사용하여 IAM Identity Center에서 사용자 액세스를 관리하는 것이 좋습니다. IAM 사용자가 필요한 특정 상황에 대한 자세한 내용은 IAM 사용자(역할이 아님)를 생성해야 하는 경우를 참조하세요.

Amazon Cognito를 사용하여 모바일 또는 웹 기반 앱 사용자 페더레이션

AWS 리소스에 액세스하는 모바일 또는 웹 기반 앱을 생성하는 경우, 이 앱에는 AWS에 프로그래밍 방식으로 요청하기 위해 보안 자격 증명이 필요합니다. 대부분의 모바일 애플리케이션 시나리오의 경우 Amazon Cognito 사용을 권장합니다. 이 서비스와 함께 AWS Mobile SDK for iOS 및 AWS Mobile SDK for Android and Fire OS를 사용하면 사용자의 고유 자격 증명을 생성하고 사용자를 인증하여 AWS 리소스에 안전하게 액세스하도록 할 수 있습니다. Amazon Cognito는 다음 섹션에 나열된 자격 증명 공급자를 지원할 뿐만 아니라 개발자 인증 자격 증명과 인증되지 않은(게스트) 액세스까지 지원합니다. 또한, Amazon Cognito는 사용자가 디바이스 간에 이동할 때 데이터가 보존되도록 사용자 데이터를 동기화하기 위한 API 작업도 제공합니다. 자세한 내용은 모바일 앱용 Amazon Cognito 단원을 참조하십시오.

퍼블릭 자격 증명 서비스 공급자 또는 OpenID Connect를 사용하여 사용자 페더레이션

가능하면 항상 모바일 및 웹 기반 애플리케이션 시나리오에 Amazon Cognito를 사용합니다. Amazon Cognito는 퍼블릭 자격 증명 공급자 서비스를 통해 대부분의 백그라운드 작업을 수행합니다. 동일한 타사 서비스를 사용하며 익명 로그인을 지원하기도 합니다. 그러나 고급 시나리오의 경우에는 Login with Amazon, Facebook, Google, 또는 OpenID Connect(OIDC)와 호환되는 모든 IdP로 직접 작업할 수 있습니다. 이들 서비스 중 한 가지를 이용한 OIDC 페더레이션에 대한 자세한 내용은 OIDC 페더레이션 항목을 참조하세요.

SAML 2.0으로 사용자 연동하기

조직에서 SAML 2.0(Security Assertion Markup Language 2.0)을 지원하는 자격 증명 공급자 소프트웨어 패키지를 이미 사용하는 경우 IdP(자격 증명 공급자)인 조직과 서비스 공급자인 AWS 간에 신뢰를 형성할 수 있습니다. 그러면 SAML을 사용하여 사용자에게 AWS Management Console에 대한 연동 SSO(Single-Sing On) 또는 AWS API 작업을 호출하기 위한 페더레이션 액세스를 제공할 수 있습니다. 예를 들어 회사가 Microsoft Active Directory와 Active Directory Federation Services를 이용한다면, SAML 2.0을 사용해 연동할 수 있습니다. SAML 2.0를 이용한 사용자 연동에 대한 세부 정보는 SAML 2.0 연동을 참조하세요.

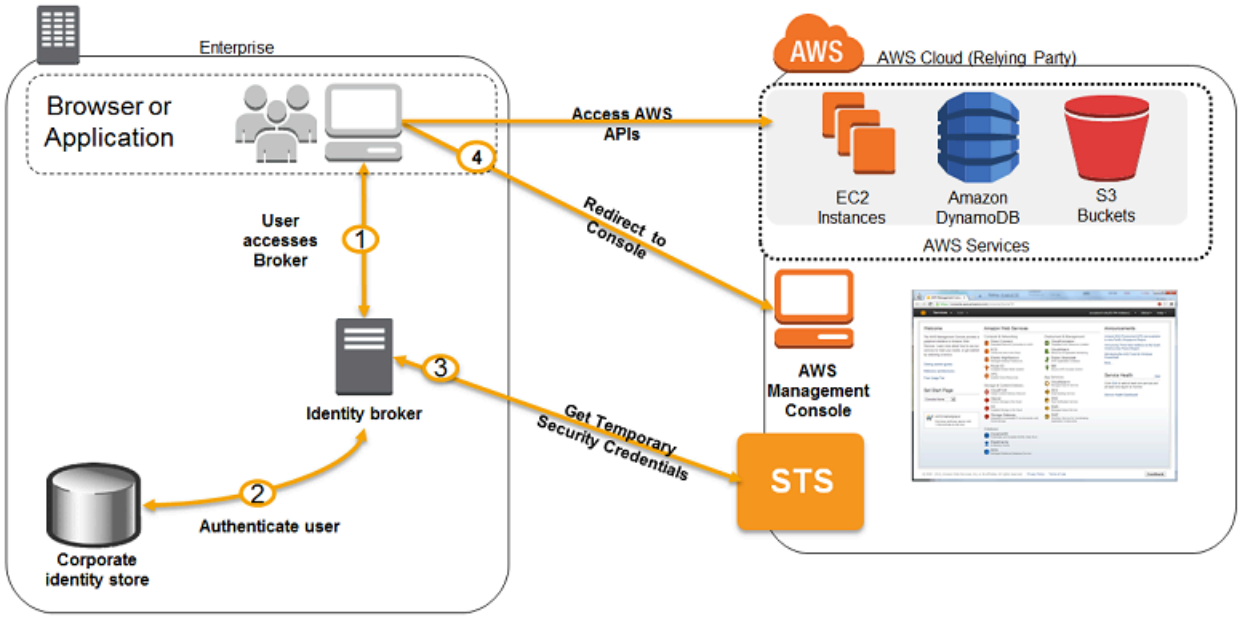
사용자 지정 자격 증명 브로커 애플리케이션 생성에 의한 사용자 연동

자격 증명 스토어가 SAML 2.0과 호환되지 않는다면, 사용자 지정 자격 증명 브로커 애플리케이션을 구축해 비슷한 기능을 수행할 수 있습니다. 브로커 애플리케이션이 사용자를 인증하고, AWS에게 사용자를 위한 임시 자격 증명을 요청한 다음, 이를 사용자에게 제공해 AWS 리소스에 액세스하도록 합니다.

예를 들어 Example Corp.에 회사의 AWS 리소스에 액세스하는 내부 애플리케이션을 실행해야 하는 직원들이 많다고 합시다. 직원들은 이미 회사 자격 증명 및 인증 시스템에 자격 증명이 있어서 Example Corp는 각 직원의 IAM 사용자를 별도로 생성하길 원하지 않습니다.

Bob은 Example Corp의 개발자입니다. Example Corp 내부 애플리케이션에서 회사의 AWS 리소스에 액세스하도록 하기 위해 Bob은 사용자 지정 자격 증명 브로커 애플리케이션을 개발합니다. 그 애플리케이션은 직원들이 기존 Example Corp. 자격 증명 및 인증 시스템에 로그인된 상태인지 확인하는데, 그 시스템은 LDAP, Active Directory, 또는 다른 시스템을 사용할 수 있습니다. 그 다음에 자격 증명 브로커 애플리케이션은 직원들에 대한 임시 보안 자격 증명을 획득합니다. 이 시나리오는 이전 시나리오(사용자 지정 인증 시스템을 사용하는 모바일 앱)과 유사합니다. 단지 AWS 리소스에 액세스해야 하는 애플리케이션이 모두 회사 네트워크 내에서 실행되고 회사에 기존 인증 시스템이 있다는 점만 다릅니다.

임시 보안 자격 증명을 얻기 위해 자격 증명 브로커 애플리케이션은 밥(Bob)이 사용자들에 대한 정책을 어떻게 관리하고자 하는지, 그리고 임시 자격 증명이 언제 만료되는지에 따라 `AssumeRole` 또는 `GetFederationToken`을 호출해 임시 보안 자격 증명을 획득합니다. (이러한 API 작업 간의 차이점을 보려면 IAM의 임시 보안 자격 증명 및 사용자 임시 보안 자격 증명에 대한 권한 제어를 참조하세요.) 호출은 AWS 액세스 키 ID, 보안 액세스 키, 세션 토큰으로 구성된 임시 보안 자격 증명을 반환합니다. 자격 증명 브로커 애플리케이션은 이 임시 보안 자격 증명을 내부 회사 애플리케이션에서도 사용할 수 있게 해줍니다. 그 앱은 그 임시 자격 증명을 사용해 AWS를 직접 호출할 수 있습니다. 그 앱은 자격 증명이 만료될 때까지 캐싱한 다음, 새로운 일련의 임시 자격 증명을 요청합니다. 다음은 이 시나리오를 설명한 그림입니다.



이 시나리오에는 다음과 같은 속성이 있습니다.

- 자격 증명 브로커 애플리케이션은 IAM의 보안 토큰 서비스(STS) API에 액세스하여 임시 보안 자격 증명을 생성할 수 있는 권한이 있습니다.

- 신원 증명 브로커 애플리케이션을 통해 기존 인증 시스템 내에서 직원이 인증되었는지 확인할 수 있습니다.
- 사용자는 AWS 관리 콘솔에 대한 액세스 권한을 제공하는 임시 URL(Single Sign-On이라고 함)을 가져올 수 있습니다.

임시 보안 자격 증명 생성에 대한 자세한 내용은 임시 보안 자격 증명 요청를 참조하세요 . AWS 관리 콘솔에 대한 액세스 권한을 얻는 페더레이션 사용자에 대한 자세한 내용은 SAML 2.0 페더레이션 사용자가 AWS Management Console에 액세스할 수 있게 하기 섹션을 참조하세요.

IAM 역할 생성

역할을 생성하려면 AWS Management Console, AWS CLI, Tools for Windows PowerShell 또는 IAM API를 사용할 수 있습니다.

AWS Management Console을 사용하는 경우 마법사가 역할 생성 절차를 단계별로 안내합니다. 마법사의 진행 단계는 생성하는 역할 대상이 AWS 서비스일 때, AWS 계정일 때, 혹은 페더레이션 사용자일 때에 따라 약간 다릅니다.

IAM 사용자 역할

이 역할을 생성하여 AWS 계정 내에 또는 소유하고 있는 다른 AWS 계정에 정의된 역할에 권한을 위임합니다. 한 계정의 사용자는 동일한 또는 다른 계정의 역할로 전환할 수 있습니다. 사용자는 역할을 사용하는 동안 해당 작업만을 수행하고 해당 역할에서 허용한 리소스만 액세스할 수 있지만, 이들의 원래 사용자 권한은 일시 중지된 상태입니다. 사용자가 역할을 끝내면 원래 사용자 권한이 회복됩니다.

자세한 내용은 IAM 사용자에게 권한을 위임할 역할 생성 단원을 참조하십시오.

크로스 계정 액세스 역할 생성에 대한 자세한 내용은 사용자 지정 트러스트 정책을 사용하여 역할 생성의 내용을 참조하세요.

AWS 서비스 역할

이 역할을 생성하여 사용자 대신 작업을 수행할 권한을 부여하는 역할을 서비스에 위임합니다. 서비스에 전달하는 서비스 역할에는 서비스가 해당 서비스와 연결된 작업을 수행할 수 있도록 허용하는 권한이 포함된 IAM 정책이 있어야 합니다. 각 AWS 서비스마다 서로 다른 권한이 필요합니다.

서비스 역할 생성에 대한 자세한 내용은 AWS 서비스에 대한 권한을 위임할 역할 생성의 내용을 참조하세요.

서비스 연결 역할 생성에 대한 자세한 내용은 서비스 연결 역할 생성의 내용을 참조하세요.

ID 페더레이션 역할

이 역할을 생성하여 AWS 외부에 이미 ID가 있는 사용자에게 권한을 위임합니다. 자격 증명 공급자를 사용하면 사용자 지정 로그인 코드를 생성할 필요도, 그리고 자신의 사용자 자격 증명을 관리할 필요도 없습니다. 외부 사용자는 IdP를 통해 로그인하고, 이러한 외부 자격 증명에 계정의 AWS 리소스를 사용할 권한을 제공할 수 있습니다. ID 제공업체를 사용하면 애플리케이션에서 사용자 액세스 키 같은 장기 보안 자격 증명을 배포하거나 포함할 필요가 없으므로 AWS 계정을 안전하게 보호할 수 있습니다.

자세한 내용은 타사 ID 제공업체의 역할 생성(페더레이션) 단원을 참조하십시오.

IAM 사용자에게 권한을 위임할 역할 생성

IAM 역할을 사용해 AWS 리소스에 대한 액세스 권한을 위임할 수 있습니다. IAM 역할을 사용해 신뢰하는 계정과 다른 AWS 신뢰받는 계정 간에 신뢰 관계를 설정할 수 있습니다. 신뢰하는 계정은 액세스되는 리소스를 소유하고 신뢰받는 계정은 리소스에 대한 액세스가 필요한 사용자를 저장합니다. 그러나, 다른 계정이 해당 계정의 리소스를 소유할 수 있는 가능성이 있습니다. 예를 들어, 신뢰받는 계정은 신뢰 계정이 Amazon S3 버킷의 새로운 객체를 생성하는 것처럼 새로운 리소스를 생성하도록 허용할 수 있습니다. 이러한 경우, 리소스를 생성하는 계정은 리소스를 소유하고 누구에게 리소스에 대한 액세스를 부여할지 제어합니다.

신뢰 관계를 생성한 후 신뢰받는 계정의 IAM 사용자 또는 애플리케이션은 AWS Security Token Service(AWS STS) `AssumeRole` API 작업을 사용할 수 있습니다. 이 작업은 계정의 AWS 리소스에 액세스할 수 있는 임시 보안 자격 증명을 제공합니다.

계정은 둘 다 직접 제어할 수 있거나 사용자가 속한 계정의 경우 타사가 제어할 수 있습니다. 사용자가 있는 다른 계정이 귀하가 제어하지 않는 AWS 계정에 있는 경우 `externalId` 속성을 사용할 수 있습니다. 외부 ID는 나와 타사 계정의 관리자 간에 합의한 숫자 또는 단어가 될 수 있습니다. 이 옵션은 요청에 올바른 `sts:ExternalID`가 포함된 경우에만 사용자가 역할을 맡을 수 있도록 허용하는 조건을 신뢰 정책에 자동으로 추가합니다. 자세한 내용은 타사가 소유한 AWS 계정에 액세스 단원을 참조하십시오.

역할을 사용해 권한을 위임하는 방법에 대한 자세한 내용은 역할 용어 및 개념 섹션을 참조하세요. 서비스 연결을 사용하여 서비스가 해당 계정의 리소스에 액세스할 수 있도록 허용하는 방법은 AWS 서비스에 대한 권한을 위임할 역할 생성 섹션을 참조하세요.

IAM 역할 생성(콘솔)

AWS Management Console을 사용하여 IAM 사용자가 수임할 수 있는 역할을 생성할 수 있습니다. 예를 들면 프로덕션 환경에서 개발 환경을 격리하기 위해 조직이 여러 개의 AWS 계정을 갖고 있다고 가

정합시다. 개발 계정의 사용자가 프로덕션 계정의 리소스에 액세스할 수 있도록 하는 역할 생성에 대한 개괄적 정보는 분리된 개발 및 프로덕션 계정을 사용한 예제 시나리오 섹션을 참조하세요.

역할을 만들려면(콘솔 사용)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 콘솔의 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
3. AWS 계정 역할 유형을 선택합니다.
4. 계정에 대한 역할을 생성하려면 이 계정(This account)을 선택합니다. 다른 계정에 대한 역할을 생성하려면 다른 AWS 계정(Another AWS 계정)을 선택하고 리소스에 대한 액세스 권한을 부여할 계정 ID(Account ID)를 입력합니다.

   지정된 계정의 관리자는 해당 계정의 IAM 사용자에게 이 역할을 수임할 수 있는 권한을 부여할 수 있습니다. 이를 위해 관리자는 `sts:AssumeRole` 작업에 대한 권한을 부여하는 정책을 사용자나 그룹에 연결합니다. 이 정책은 역할의 ARN을 `Resource`로 지정해야 합니다.
5. 통제권이 없는 계정의 사용자에게 권한을 부여하려면 사용자는 이 역할을 프로그래밍 방식으로 가정하고 외부 ID 필요(Require external ID)를 선택합니다. 외부 ID는 나와 서드 파티 계정의 관리자 간에 합의한 숫자 또는 단어가 될 수 있습니다. 이 옵션은 요청에 올바른 `sts:ExternalID`가 포함된 경우에만 사용자가 역할을 맡을 수 있도록 허용하는 조건을 신뢰 정책에 자동으로 추가합니다. 자세한 내용은 타사가 소유한 AWS 계정에 액세스 단원을 참조하십시오.

   > **Important**
   >
   > 이 옵션을 선택하면 역할로의 액세스가 AWS CLI, Tools for Windows PowerShell 또는 AWS API를 통해서만 가능하도록 제한됩니다. 이는 AWS 콘솔을 사용해 해당 신뢰 정책에 `externalId` 조건이 있는 역할로 전환할 수 없기 때문입니다. 하지만 관련 SDK를 통해 스크립트나 애플리케이션을 작성하여 프로그래밍 방식으로 이러한 종류의 액세스를 만들 수 있습니다. 자세한 내용 및 샘플 스크립트는 AWS 보안 블로그의 AWS Management Console에 대한 크로스 계정 액세스를 가능하게 하는 방법을 참조하세요.
6. 멀티 팩터 인증(MFA)으로 로그인하는 사용자로 역할을 제한하려면, Require MFA(MFA 필요)를 선택합니다. 이렇게 하면 MFA 로그인을 확인하는 역할의 신뢰 정책에 조건이 추가됩니다. 역할을 맡으려는 사용자는 구성된 MFA 디바이스에서 임시 일회용 암호로 로그인해야 합니다. MFA 인증을 사용하지 않는 사용자는 역할을 맡을 수 없습니다. MFA에 대한 자세한 내용은 IAM의 AWS 다중 인증 섹션을 참조하세요.
7. Next(다음)를 선택합니다.

8. IAM은 계정의 AWS 관리형 또는 고객 관리형 정책 목록을 포함합니다. 권한 정책을 사용하기 위한 정책을 선택하거나 정책 생성을 선택하여 새 브라우저 탭을 열고 완전히 새로운 정책을 생성합니다. 자세한 내용은 IAM 정책 생성 단원을 참조하십시오. 정책을 생성하면 탭을 닫고 원래 탭으로 돌아갑니다. 누구든지 역할에게 위임하려는 권한 정책 옆의 확인란을 선택합니다. 원할 경우, 여기서 정책을 선택하지 않고 나중에 정책을 만들어서 역할에 연결할 수 있습니다. 기본적으로 역할은 권한이 없습니다.
9. (선택 사항) 권한 경계를 선택합니다. 이는 고급 기능입니다.

   Set permissions boundary(권한 경계 설정) 섹션을 열고 Use a permissions boundary to control the maximum role permissions(최대 역할 권한을 관리하기 위한 권한 경계 사용)을 선택합니다. 정책을 선택하여 권한 경계를 사용하세요.
10. Next(다음)를 선택합니다.
11. 역할 이름에 역할의 이름을 입력합니다. 역할 이름은 AWS 계정 내에서 고유해야 합니다. 역할 이름이 정책에서 또는 ARN의 일부로 사용되는 경우 역할 이름은 대소문자를 구분합니다. 콘솔에서 고객에게 역할 이름이 표시되는 경우(예: 로그인 프로세스 중) 역할 이름은 대소문자를 구분하지 않습니다. 다양한 엔터티가 역할을 참조할 수 있기 때문에 역할이 생성된 후에는 역할 이름을 편집할 수 없습니다.
12. (선택 사항)설명에 새 역할에 대한 설명을 입력합니다.
13. 1단계: 신뢰할 수 있는 엔터티 선택(Step 1: Select trusted entities) 또는 2단계: 권한 추가(Step 2: Add permissions) 섹션에서 편집(Edit)을 선택하여 역할에 대한 사용 사례와 권한을 편집합니다. 편집을 위해 이전 페이지로 돌아갑니다.
14. (선택 사항) 태그를 키 값 페어로 연결하여 메타데이터를 역할에 추가합니다. IAM에서의 태그 사용에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요.
15. 역할을 검토한 다음 Create role을 선택합니다.

> **Important**
>
> 필요한 구성의 절반이 끝났습니다. 이제 신뢰할 수 있는 계정의 개별 사용자에게 콘솔의 역할로 전환하거나 역할을 프로그래밍 방식으로 위임할 수 있는 권한을 부여해야 합니다. 이 단계에 대한 자세한 내용은 사용자에게 역할을 전환할 권한 부여 섹션을 참조하세요.

IAM 역할 생성(AWS CLI)

AWS CLI에서 역할을 만들려면 여러 단계를 거쳐야 합니다. 콘솔을 사용하여 역할을 만들 때는 많은 단계가 자동으로 수행되지만 AWS CLI를 사용하면 각 단계를 직접 명시적으로 수행해야 합니다. 역할을 만든 다음 권한 정책을 역할에 할당해야 합니다. 선택적으로 역할에 대한 권한 경계를 설정할 수 있습니다.

크로스 계정 액세스에 대한 역할을 만들려면(AWS CLI)

1. 역할 생성: aws iam create-role
2. 역할에 관리형 권한 정책 연결: aws iam attach-role-policy

   또는

   역할을 위한 인라인 권한 정책 생성: aws iam put-role-policy
3. (선택 사항) 태그를 연결하여 사용자 지정 속성을 역할에 추가: aws iam tag-role

   자세한 내용은 IAM 역할의 태그 관리(AWS CLI 또는 AWS API) 단원을 참조하십시오.
4. (선택 사항) 역할(aws iam put-role-permissions-boundary)에 대한 권한 경계를 설정합니다.

   이 권한 경계는 역할이 가질 수 있는 최대 권한을 관리합니다. 권한 경계는 고급 AWS 기능입니다.

다음 예는 단순한 환경에서 크로스 계정 역할을 생성하는 가장 일반적인 단계 중 첫 두 단계를 보여줍니다. 이 예제는 `123456789012` 계정에 있는 모든 사용자가 역할을 수임하고 `example_bucket` Amazon S3 버킷을 볼 수 있도록 허용합니다. 이 예제에서도 Windows가 구동되는 클라이언트 컴퓨터를 사용 중이며 명령줄 인터페이스를 계정 자격 증명 및 리전으로 이미 구성했다고 가정합니다. 자세한 내용은 AWS 명령줄 인터페이스 구성을 참조하세요.

이 예제는 역할을 생성할 경우 첫 번째 명령의 다음 신뢰 정책을 포함합니다. 이 신뢰 정책은 `123456789012` 계정에서 사용자가 `AssumeRole` 작업을 사용하여 역할을 가정할 수 있도록 허용합니다. 단, 사용자가 `SerialNumber` 및 `TokenCode` 파라미터를 사용하는 MFA 인증을 제공하는 경우에만 허용합니다. MFA에 대한 자세한 내용은 IAM의 AWS 다중 인증 섹션을 참조하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
```

```
            "Action": "sts:AssumeRole",
            "Condition": { "Bool": { "aws:MultiFactorAuthPresent": "true" } }
        }
    ]
}
```

> **Important**
>
> `Principal` 요소에 특정 IAM 역할 또는 사용자에 대한 ARN이 포함되어 있으면, 정책을 저장할 때 해당 ARN이 고유 보안 주체 ID로 변환됩니다. 그러면 누군가가 해당 역할 또는 사용자를 제거하고 다시 만들어 본인의 권한을 에스컬레이션할 위험을 완화할 수 있습니다. 일반적으로 콘솔에서는 이 ID가 보이지 않습니다. 신뢰 정책이 표시될 때 해당 ARN으로 다시 역변환되기 때문입니다. 그러나 역할 또는 사용자를 삭제할 경우, 보안 주체 ID가 콘솔에 표시됩니다. AWS에서 더 이상 이를 ARN에 다시 매핑할 수 없기 때문입니다. 따라서 신뢰 정책의 `Principal` 요소에서 참조된 사용자 또는 역할을 삭제하고 다시 생성하는 경우, ARN을 바꾸도록 역할을 편집해야 합니다.

두 번째 명령을 사용할 경우, 기존 관리형 정책을 역할에 연결해야 합니다. 다음 권한 정책에서는 역할을 수임하는 사용자가 `example_bucket` Amazon S3 버킷에서 `ListBucket` 작업만 수행하도록 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": "s3:ListBucket",
          "Resource": "arn:aws:s3:::example_bucket"
      }
  ]
}
```

이 `Test-UserAccess-Role` 역할을 생성하기 위해서는 이전 신뢰 정책을 `trustpolicyforacct123456789012.json` 이름으로 로컬 `policies` 드라이브의 `C:` 폴더에 먼저 저장해야 합니다. 그런 다음 이전 권한 정책을 고객 관리형 정책으로서 `PolicyForRole` 이름으로 AWS 계정에 저장합니다. 그러고 나면 다음 명령을 사용하여 역할을 만들고 관리형 정책을 연결합니다.

```
# Create the role and attach the trust policy file that allows users in the specified
 account to assume the role.
$ aws iam create-role --role-name Test-UserAccess-Role --assume-role-policy-document
 file://C:\policies\trustpolicyforacct123456789012.json

# Attach the permissions policy (in this example a managed policy) to the role to
 specify what it is allowed to do.
$ aws iam attach-role-policy --role-name Test-UserAccess-Role --policy-arn
 arn:aws:iam::123456789012:policy/PolicyForRole
```

> **Important**
>
> 필요한 구성의 절반이 끝났습니다. 이제 신뢰할 수 있는 계정의 개별 사용자에게 역할로 전환할 수 있는 권한을 부여해야 합니다. 이 단계에 대한 자세한 내용은 사용자에게 역할을 전환할 권한 부여 섹션을 참조하세요.

역할을 만든 다음 AWS 작업을 수행하거나 AWS 리소스에 액세스할 수 있는 권한을 부여해야 123456789012 계정의 사용자가 역할을 위임할 수 있습니다. 자세한 내용은 IAM 역할로 전환(AWS CLI) 단원을 참조하십시오.

IAM 역할 생성(AWS API)

AWS API에서 역할을 만들려면 여러 단계를 거쳐야 합니다. 콘솔을 사용하여 역할을 만들 때는 많은 단계가 자동으로 수행되지만 API를 사용하면 각 단계를 직접 명시적으로 수행해야 합니다. 역할을 만든 다음 권한 정책을 역할에 할당해야 합니다. 선택적으로 역할에 대한 권한 경계를 설정할 수 있습니다.

코드로 역할을 만들려면(AWS API)

1. 역할 만들기: CreateRole

   역할의 신뢰 정책에 대해 파일 위치를 지정할 수 있습니다.
2. 역할에 관리형 권한 정책 연결: AttachRolePolicy

   또는

   역할을 위한 인라인 권한 정책 생성: PutRolePolicy

> **Important**
>
> 필요한 구성의 절반이 끝났습니다. 이제 신뢰할 수 있는 계정의 개별 사용자에게 역할로 전환할 수 있는 권한을 부여해야 합니다. 이 단계에 대한 자세한 내용은 사용자에게 역할을 전환할 권한 부여 섹션을 참조하세요.

3. (선택 사항) 태그를 연결하여 사용자 지정 속성을 사용자에게 추가: TagRole

   자세한 내용은 IAM 사용자의 태그 관리(AWS CLI 또는 AWS API) 단원을 참조하십시오.

4. (선택 사항) 역할(PutRolePermissionsBoundary)에 대한 권한 경계를 설정합니다.

   이 권한 경계는 역할이 가질 수 있는 최대 권한을 관리합니다. 권한 경계는 고급 AWS 기능입니다.

역할을 만든 다음 AWS 작업을 수행하거나 AWS 리소스에 액세스할 수 있는 권한을 부여해야 계정의 사용자에게 권한을 부여하여 역할을 위임할 수 있습니다. 역할 위임하기에 대한 자세한 내용은 IAM 역할로 전환(AWS API) 섹션을 참조하세요.

IAM 역할 생성(AWS CloudFormation)

AWS CloudFormation에서 IAM 역할을 생성하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 리소스 및 속성 참조와 예제를 참조하세요.

AWS CloudFormation의 IAM 템플릿에 관한 자세한 정보는 AWS CloudFormation 사용 설명서의 AWS Identity and Access Management 템플릿 코드 조각을 참조하세요.

AWS 서비스에 대한 권한을 위임할 역할 생성

AWS 서비스는 역할을 사용하여 서비스가 사용자를 대신하여 다른 서비스의 리소스로 액세스할 수 있어야 합니다. 서비스가 사용자를 대신하여 작업을 수행하기 위해 수임한 역할을 서비스 역할이라고 합니다. 역할이 서비스에 대해 특수한 목적을 수행하는 경우 서비스 연결 역할로 분류됩니다. 서비스 연결 역할을 사용하여 지원되는 서비스 또는 서비스가 임시 자격 증명의 형식을 지원하는지 여부를 확인하려면 AWS IAM으로 작업하는 서비스 섹션을 참조하세요. 개별 서비스가 역할을 사용하는 방법을 알아보려면 테이블에서 서비스 이름을 선택하여 해당 서비스의 설명서를 확인합니다.

`PassRole` 권한을 설정할 때 사용자에게 부여하려는 역할보다 더 많은 권한이 있는 역할을 사용자가 넘기지 않도록 해야 합니다. 예를 들어 Alice가 Amazon S3 작업을 수행하는 것이 허용되지 않을 수 있습니다. Alice가 Amazon S3 작업을 허용하는 서비스에 역할을 넘길 수 있으면 해당 서비스에서 작업 실행 시 Alice를 대신하여 Amazon S3 작업을 수행할 수 있습니다.

역할을 통해 권한을 위임하는 방법에 대한 자세한 내용은 역할 용어 및 개념 섹션을 참조하세요.

서비스 역할 권한

IAM 엔터티(사용자 또는 역할)가 서비스 연결 역할을 작성하거나 편집할 수 있도록 권한을 구성해야 합니다.

> **Note**
>
> 서비스 연결 역할의 ARN에는 다음 정책에 *SERVICE-NAME*.amazonaws.com으로 표시된 서비스 보안 주체가 포함됩니다. 서비스 보안 주체는 대소문자를 구분하고 AWS 서비스마다 형식이 다를 수 있으므로 추측하지 마세요. 서비스의 보안 주체를 보려면 해당 서비스 링크된 역할 설명서 섹션을 참조하세요.

IAM 엔터티가 특정 서비스 역할을 생성하도록 허용하려면

서비스 역할을 생성해야 하는 IAM 엔터티에 다음 정책을 추가합니다. 이 정책으로 특정 서비스에 대하여 구체적인 이름이 있는 서비스 역할을 만들 수 있습니다. 그런 다음 관리형 또는 인라인 정책을 해당 역할에 연결할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreateRole",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
        }
    ]
}
```

IAM 엔터티가 모든 서비스 역할을 생성하도록 허용하려면

AWS에서는 관리 사용자만 서비스 역할을 생성할 수 있도록 허용하는 것이 좋습니다. 역할을 생성하고 정책을 연결할 수 있는 권한이 있는 사람은 자신의 권한을 에스컬레이션할 수 있습니다. 대신 필요한 역할만 생성하거나 관리자가 대신 서비스 역할을 생성하도록 허용하는 정책을 만듭니다.

관리자가 사용자 전체 AWS 계정에 액세스할 수 있도록 허용하는 정책을 연결하려면 AdministratorAccess AWS 관리형 정책을 사용합니다.

IAM 엔터티가 서비스 역할을 편집할 수 있도록 허용하려면

서비스 역할을 편집해야 하는 IAM 엔터티에 다음 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EditSpecificServiceRole",
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:DeleteRolePolicy",
                "iam:DetachRolePolicy",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:ListAttachedRolePolicies",
                "iam:ListRolePolicies",
                "iam:PutRolePolicy",
                "iam:UpdateRole",
                "iam:UpdateRoleDescription"
            ],
            "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
        },
        {
            "Sid": "ViewRolesAndPolicies",
            "Effect": "Allow",
            "Action": [
                "iam:GetPolicy",
                "iam:ListRoles"
            ],
            "Resource": "*"
        }
    ]
}
```

IAM 엔터티가 특정 서비스 역할을 삭제하도록 허용하려면

특정 서비스 역할을 삭제해야 하는 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": "iam:DeleteRole",
    "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
}
```

IAM 엔터티가 모든 서비스 역할을 삭제하도록 허용하려면

AWS에서는 관리 사용자만 서비스 역할을 삭제할 수 있도록 허용하는 것이 좋습니다. 대신, 필요한 역할만 삭제하거나 관리자가 대신 서비스 역할을 삭제하도록 허용하는 정책을 만듭니다.

관리자가 사용자 전체 AWS 계정에 액세스할 수 있도록 허용하는 정책을 연결하려면 AdministratorAccess AWS 관리형 정책을 사용합니다.

AWS 서비스에 대한 역할 생성(콘솔)

AWS Management Console을 사용하여 서비스의 역할을 만들 수 있습니다. 일부 서비스는 두 개 이상의 서비스 역할을 지원하기 때문에 어떤 사용 사례를 선택할지 확인하려면 해당 서비스의 AWS 설명서 섹션을 참조하세요. 서비스에서 역할을 위임할 수 있도록 역할에 필요한 신뢰 정책과 권한 정책을 할당하는 방법을 알아볼 수 있습니다. 역할에 대한 권한을 관리할 수 절차는 서비스가 어떻게 사용 사례를 정의하느냐와 서비스 링크된 역할을 생성할 수 있는지 여부에 따라 다양할 수 있습니다.

AWS 서비스에 대한 역할 생성(IAM 콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할을 선택하고 역할 생성을 선택합니다.
3. 신뢰할 수 있는 엔터티 유형에 AWS 서비스를 선택합니다.
4. 서비스 또는 사용 사례의 경우 서비스를 선택한 다음, 사용 사례를 선택합니다. 사용 사례는 서비스에 필요한 신뢰 정책을 포함하기 위해 서비스에서 정합니다.
5. Next(다음)를 선택합니다.
6. 권한 정책의 경우 선택한 사용 사례에 따라 옵션이 달라집니다.
   - 서비스가 역할에 대한 권한을 정의하는 경우 권한 정책을 선택할 수 없습니다.
   - 제한된 권한 정책 세트에서 선택합니다.
   - 모든 권한 정책에서 선택합니다.
   - 권한 정책 없음을 선택하고 역할이 생성된 후 정책을 생성한 다음, 정책을 역할에 연결합니다.

7. (선택 사항) 권한 경계를 선택합니다. 이는 서비스 역할에서 가능한 고급 기능이며 서비스 링크된 역할은 아닙니다.

   a. 권한 경계 설정 섹션을 열고 최대 역할 권한을 관리하기 위한 권한 경계 사용을 선택합니다.

      IAM은 계정의 AWS 관리형 또는 고객 관리형 정책 목록을 포함합니다.

   b. 정책을 선택하여 권한 경계를 사용하세요.

8. Next(다음)를 선택합니다.

9. 역할 이름의 경우 옵션은 서비스에 따라 달라집니다.

   - 서비스에서 역할 이름을 정의하는 경우 이 역할 이름을 편집할 수 없습니다.
   - 서비스에서 역할 이름에 대한 접두사를 정의하는 경우 사용자가 선택적 접미사를 입력할 수 있습니다.
   - 서비스에서 역할 이름을 정의하지 않는 경우 역할 이름을 지정할 수 있습니다.

   > **Important**
   >
   > 역할 이름을 지정할 때는 다음 사항에 유의하세요.
   >
   > - 역할 이름은 AWS 계정 내에서 고유해야 하지만 대소문자를 구분하지는 않습니다.
   >
   >   예를 들어, 이름이 **`PRODROLE`**과 **`prodrole`**, 두 가지로 지정된 역할을 만들지 마십시오. 역할 이름이 정책 또는 ARN의 일부로 사용되는 경우 역할 이름은 대소문자를 구분합니다. 그러나 로그인 프로세스와 같이 콘솔에서 역할 이름이 고객에게 표시되는 경우에는 역할 이름이 대소문자를 구분하지 않습니다.
   > - 다른 엔터티가 역할을 참조할 수 있기 때문에 역할이 생성된 후에는 역할 이름을 편집할 수 없습니다.

10. (선택 사항)설명에 역할에 대한 설명을 입력합니다.

11. (선택 사항) 역할에 대한 사용 사례와 권한을 편집하려면 1단계: 신뢰할 수 있는 엔터티 선택 또는 2단계: 권한 추가 섹션에서 편집을 선택합니다.

12. (선택 사항) 태그를 키-값 페어로 연결하여 역할을 식별, 구성 또는 검색합니다. IAM에서 태그 사용에 대한 자세한 내용을 알아보려면 IAM 사용 설명서의 IAM 리소스에 태그 지정을 참조하세요.

13. 역할을 검토한 다음 역할 생성을 선택합니다.

서비스에 대한 역할 생성(AWS CLI)

AWS CLI에서 역할을 만들려면 여러 단계를 거쳐야 합니다. 콘솔을 사용하여 역할을 만들 때는 많은 단계가 자동으로 수행되지만 AWS CLI를 사용하면 각 단계를 직접 명시적으로 수행해야 합니다. 역할을 만든 다음 권한 정책을 역할에 할당해야 합니다. 작업 중인 서비스가 Amazon EC2인 경우에도 인스턴스 프로파일을 생성하여 거기에 역할을 추가해야 합니다. 선택적으로 역할에 대한 권한 경계를 설정할 수 있습니다.

AWS CLI에서 AWS 서비스에 대한 역할을 만들려면

1. 다음 `create-role` 명령은 Test-Role이라는 역할을 생성하고 해당 역할에 신뢰 정책을 연결합니다.

   ```
   aws iam create-role --role-name Test-Role --assume-role-policy-document
   file://Test-Role-Trust-Policy.json
   ```

2. 역할에 관리형 권한 정책 연결: aws iam attach-role-policy.

   예를 들어 `attach-role-policy` 명령은 `ReadOnlyAccess`라는 AWS 관리형 정책을 `ReadOnlyRole`이라는 IAM 역할에 연결합니다.

   ```
   aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
   ReadOnlyAccess --role-name ReadOnlyRole
   ```

   또는

   역할을 위한 인라인 권한 정책 생성: aws iam put-role-policy

   인라인 권한 정책을 추가하려면 다음 예제를 참조하세요.

   ```
   aws iam put-role-policy --role-name Test-Role --policy-name
   ExamplePolicy --policy-document file://AdminPolicy.json
   ```

3. (선택 사항) 태그를 연결하여 사용자 지정 속성을 역할에 추가: aws iam tag-role

   자세한 내용은 IAM 역할의 태그 관리(AWS CLI 또는 AWS API) 단원을 참조하십시오.

4. (선택 사항) 역할(aws iam put-role-permissions-boundary)에 대한 권한 경계를 설정합니다.

   이 권한 경계는 역할이 가질 수 있는 최대 권한을 관리합니다. 권한 경계는 고급 AWS 기능입니다.

Amazon EC2 또는 Amazon EC2를 사용하는 다른 AWS 서비스에 대해 역할을 사용할 경우 인스턴스 프로파일에 역할을 저장해야 합니다. 인스턴스 프로파일은 시작할 때 Amazon EC2 인스턴스에 연결할

수 있는 역할을 위한 컨테이너입니다. 하나의 인스턴스 프로파일은 하나의 역할만 포함할 수 있으며 이 제한은 늘릴 수 없습니다. AWS Management Console을 사용하여 역할을 생성한 경우 역할과 동일한 이름을 지닌 인스턴스 프로파일이 자동으로 생성됩니다. 인스턴스 프로파일에 대한 자세한 내용은 인스턴스 프로파일 사용 섹션을 참조하세요. 역할을 사용하여 EC2 인스턴스를 시작하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 리소스에 대한 액세스 제어를 참조하세요.

인스턴스 프로파일을 만들고 여기에 역할을 저장하려면(AWS CLI)

1. 인스턴스 프로파일 생성: aws iam create-instance-profile
2. 인스턴스 프로파일에 역할 추가: aws iam add-role-to-instance-profile

아래 AWS CLI 예제 명령 집합은 역할을 생성하고 권한을 연결하는 첫 두 단계를 보여줍니다. 인스턴스 프로파일을 생성하고 프로필에 역할을 추가하는 두 단계를 보여주기도 합니다. 이 예제 신뢰 정책은 Amazon EC2 서비스가 역할을 수임하고 `example_bucket` Amazon S3 버킷을 볼 수 있도록 허용합니다. 이 예제에서는 Windows를 실행하는 클라이언트 컴퓨터에서 실행 중이며 계정 자격 증명 및 리전으로 이미 명령줄 인터페이스를 구성했다고도 가정합니다. 자세한 내용은 AWS 명령줄 인터페이스 구성을 참조하세요.

이 예제는 역할을 생성할 경우 첫 번째 명령의 다음 신뢰 정책을 포함합니다. 이 신뢰 정책은 Amazon EC2 서비스가 역할을 수임하도록 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"Service": "ec2.amazonaws.com"},
    "Action": "sts:AssumeRole"
  }
}
```

두 번째 명령을 사용할 경우, 권한 정책을 역할에 연결해야 합니다. 다음 예제 권한 정책에서는 역할이 `example_bucket` Amazon S3 버킷에서 `ListBucket` 작업만 수행하도록 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
```

```
}
```

이 Test-Role-for-EC2 역할을 생성하기 위해서는 먼저 이전 신뢰 정책을 trustpolicyforec2.json 이름으로, 이전 권한 정책을 permissionspolicyforec2.json 이름으로 로컬 C: 드라이브의 policies 디렉터리에 저장해야 합니다. 그러고 나면 다음 명령을 사용하여 역할을 만들고 인라인 정책을 연결, 인스턴스 프로파일 생성 및 인스턴스 프로파일에 역할을 추가합니다.

```
# Create the role and attach the trust policy that allows EC2 to assume this role.
$ aws iam create-role --role-name Test-Role-for-EC2 --assume-role-policy-document
 file://C:\policies\trustpolicyforec2.json

# Embed the permissions policy (in this example an inline policy) to the role to
 specify what it is allowed to do.
$ aws iam put-role-policy --role-name Test-Role-for-EC2 --policy-name Permissions-
Policy-For-Ec2 --policy-document file://C:\policies\permissionspolicyforec2.json

# Create the instance profile required by EC2 to contain the role
$ aws iam create-instance-profile --instance-profile-name EC2-ListBucket-S3

# Finally, add the role to the instance profile
$ aws iam add-role-to-instance-profile --instance-profile-name EC2-ListBucket-S3 --
role-name Test-Role-for-EC2
```

EC2 인스턴스를 시작할 때 AWS 콘솔을 사용하는 경우 인스턴스 세부 정보 구성 페이지에 인스턴스 프로파일 이름을 지정합니다. aws ec2 run-instances CLI 명령을 사용하는 경우 --iam-instance-profile 파라미터를 지정합니다.

서비스에 대한 역할 생성(AWS API)

AWS API에서 역할을 만들려면 여러 단계를 거쳐야 합니다. 콘솔을 사용하여 역할을 만들 때는 많은 단계가 자동으로 수행되지만 API를 사용하면 각 단계를 직접 명시적으로 수행해야 합니다. 역할을 만든 다음 권한 정책을 역할에 할당해야 합니다. 작업 중인 서비스가 Amazon EC2인 경우에도 인스턴스 프로파일을 생성하여 거기에 역할을 추가해야 합니다. 선택적으로 역할에 대한 권한 경계를 설정할 수 있습니다.

AWS 서비스에 대한 역학을 생성하려면(AWS API)

1. 역할 만들기: CreateRole

   역할의 신뢰 정책에 대해 파일 위치를 지정할 수 있습니다.

2. 역할에 관리형 권한 정책 연결: AttachRolePolicy

   또는

   역할을 위한 인라인 권한 정책 생성: PutRolePolicy

3. (선택 사항) 태그를 연결하여 사용자 지정 속성을 사용자에게 추가: TagRole

   자세한 내용은 IAM 사용자의 태그 관리(AWS CLI 또는 AWS API) 단원을 참조하십시오.

4. (선택 사항) 역할(PutRolePermissionsBoundary)에 대한 권한 경계를 설정합니다.

   이 권한 경계는 역할이 가질 수 있는 최대 권한을 관리합니다. 권한 경계는 고급 AWS 기능입니다.

Amazon EC2 또는 Amazon EC2를 사용하는 다른 AWS 서비스에 대해 역할을 사용할 경우 인스턴스 프로파일에 역할을 저장해야 합니다. 인스턴스 프로파일은 역할에 대한 컨테이너입니다. 각 인스턴스 프로파일은 하나의 역할만 포함할 수 있으며 이 제한은 늘릴 수 없습니다. AWS Management Console에서 역할을 생성한 경우 역할과 동일한 이름을 지닌 인스턴스 프로파일이 자동으로 생성됩니다. 인스턴스 프로파일에 대한 자세한 내용은 인스턴스 프로파일 사용을 참조하세요. 역할을 사용하여 Amazon EC2 인스턴스를 시작하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 리소스에 대한 액세스 제어를 참조하세요.

인스턴스 프로파일을 만들고 여기에 역할을 저장하려면(AWS API)

1. 인스턴스 프로파일 생성: CreateInstanceProfile
2. 인스턴스 프로파일에 역할 추가: AddRoleToInstanceProfile

서비스 연결 역할 생성

서비스 연결 역할은 AWS 서비스에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 해당 서비스에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다. 또한 연결된 서비스는 서비스 연결 역할을 만들고 수정하며 삭제하는 방법을 정의합니다. 서비스는 역할을 자동으로 만들거나 삭제할 수 있습니다. 서비스의 프로세스나 마법사를 사용하여 사용자가 역할을 만들거나 수정하거나 삭제하도록 허용할 수도 있습니다. 또는 사용자가 IAM을 사용하여 역할을 생성하거나 삭제해야 할 수도 있습니다. 방법과 관계없이 서비스 연결 역할은 서비스 설정 프로세스를 단순화합니다. 서비스가 사용자를 대신하여 작업을 완료하도록 권한을 수동으로 추가할 필요가 없기 때문입니다.

> **Note**
>
> 서비스 역할은 서비스 연결 역할과 다르다는 점을 기억하세요. 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하십시오. 서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

연결된 서비스에서 서비스 연결 역할 권한을 정의하므로 정의되지 않은 경우에만 해당 역할로 서비스를 수행할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

역할을 삭제하려면 먼저 관련 리소스를 삭제해야 합니다. 이렇게 하면 리소스에 대한 액세스 권한을 부주의로 삭제할 수 없기 때문에 리소스가 보호됩니다.

> **Tip**
>
> 서비스 연결 역할의 사용을 지원하는 서비스에 대한 자세한 정보는 AWS IAM으로 작업하는 서비스를 참조하고 서비스 연결 역할 열에 예가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

서비스 연결 역할 권한

사용자 또는 역할이 서비스 연결 역할을 작성하거나 편집할 수 있도록 IAM 엔터티(사용자 또는 역할)의 권한을 구성해야 합니다.

> **Note**
>
> 서비스 링크된 역할에 대한 ARN은 정책에서 *SERVICE-NAME*.amazonaws.com으로 나타내지는 서비스 보안 주체를 포함합니다. 각 경우마다 다르고 AWS 서비스에 따라 형식이 다양하기 때문에 서비스 보안 주체를 알기 어렵습니다. 서비스의 보안 주체를 보려면 해당 서비스 링크된 역할 설명서 섹션을 참조하세요.

IAM 엔터티가 특정 서비스 연결 역할을 만들 수 있도록 허용하려면

서비스 연결 역할을 생성해야 하는 IAM 엔터티에 다음 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-
NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX*",
            "Condition": {"StringLike": {"iam:AWSServiceName": "SERVICE-
NAME.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-
NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX*"
        }
    ]
}
```

IAM 엔터티가 서비스 연결 역할을 생성할 수 있도록 허용하려면

서비스 연결 역할 또는 필요한 정책을 포함해야 하는 모든 서비스 역할을 생성해야 하는 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다. 이 정책 명령문은 IAM 엔터티가 역할에 정책을 연결하는 것을 허용하지 않습니다.

```
{
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

IAM 엔터티가 서비스 역할의 설명을 편집할 수 있도록 허용하려면

서비스 연결 역할 또는 서비스 역할의 설명을 편집해야 하는 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": "iam:UpdateRoleDescription",
    "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

IAM 엔터티가 특정 서비스 연결 역할을 삭제하도록 허용하려면

서비스 연결 역할을 삭제해야 하는 IAM 개엔터티의 권한 정책에 다음 명령문을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-
NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX*"
}
```

IAM 엔터티가 서비스 연결 역할을 삭제할 수 있도록 허용하려면

서비스 연결 역할만 삭제하고 서비스 역할은 삭제하지 않는 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

IAM 엔터티가 기존 역할을 서비스에 전달하도록 허용하려면

일부 AWS 서비스를 사용하면 새 서비스에 연결된 역할을 생성하지 않고, 그 대신에 서비스에 기존 역할을 전달할 수 있습니다. 이렇게 하려면 사용자에게 서비스에 역할을 전달할 수 있는 권한이 있어야 합니다. 역할을 생성해야 하는 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다. 또한 이 정책 설명에서는 엔터티가 전달할 역할을 선택할 수 있는 역할 목록을 볼 수 있도록 허용합니다. 자세한 내용은 사용자에게 AWS 서비스에 역할을 전달할 권한 부여 단원을 참조하십시오.

```
{
  "Sid": "PolicyStatementToAllowUserToListRoles",
  "Effect": "Allow",
  "Action": ["iam:ListRoles"],
  "Resource": "*"
},
{
  "Sid": "PolicyStatementToAllowUserToPassOneSpecificRole",
  "Effect": "Allow",
  "Action": [ "iam:PassRole" ],
  "Resource": "arn:aws:iam::account-id:role/my-role-for-XYZ"
}
```

간접 권한과 서비스 연결 역할을 통한 간접 권한

서비스 연결 역할에서 부여한 권한은 간접적으로 다른 사용자 및 역할에게 양도할 수 있습니다. AWS 서비스가 서비스 연결 역할을 사용하는 경우 해당 서비스 연결 역할은 자체 권한을 사용하여 다른 AWS 서비스를 호출할 수 있습니다. 즉, 서비스 연결 역할을 사용하는 서비스를 호출할 권한이 있는 사용자 및 역할은 해당 서비스 연결 역할로 액세스할 수 있는 서비스에 간접적으로 액세스할 수 있습니다.

예를 들어, Amazon RDS DB 인스턴스를 생성할 때 RDS용 서비스 연결 역할이 아직 존재하지 않는다면 자동으로 하나가 생성됩니다. 이렇게 생성된 서비스 연결 역할은 사용자가 DB 인스턴스를 편집할 때마다 RDS에게 사용자 대신 Amazon EC2, Amazon SNS, Amazon CloudWatch Logs 및 Amazon Kinesis를 호출하도록 허용합니다. 계정의 사용자 및 역할이 RDS 데이터베이스를 수정하거나 생성하도록 허용하면 RDS를 호출하여 Amazon EC2, Amazon SNS, Amazon CloudWatch Logs 로그 및 Amazon Kinesis 리소스와 간접적으로 상호 작용할 수 있습니다. RDS는 서비스 연결 역할을 사용하여 해당 리소스에 액세스하기 때문입니다.

서비스 연결 역할 만들기

서비스 연결 역할을 만드는 데 사용하는 방법은 서비스에 따라 다릅니다. 경우에 따라 서비스 연결 역할을 수동으로 만들 필요가 없습니다. 예를 들어 사용자가 서비스의 특정 작업(리소스 만들기 등)을 수행할 때 서비스가 서비스 연결 역할을 자동으로 생성할 수 있습니다. 또한, 서비스가 서비스 연결 역할 지원을 시작하기 전에 서비스를 사용한 경우에는 서비스가 자동으로 해당 계정에 역할을 생성했을 수 있습니다. 자세한 내용은 내 AWS 계정에 표시되는 새 역할 섹션을 참조하세요.

다른 경우에는 서비스에서 서비스 콘솔, API 또는 CLI를 사용하여 서비스 연결 역할을 수동으로 만들도록 허용할 수 있습니다. 서비스 연결 역할의 사용을 지원하는 서비스에 대한 자세한 정보는 AWS IAM으로 작업하는 서비스를 참조하고 서비스 연결 역할 열에 예가 있는 서비스를 찾습니다. 서비스가

서비스 연결 역할 생성을 지원하는지 여부를 알아보려면 예 링크를 선택하여 해당 서비스의 서비스 연결 역할 설명서 섹션을 참조하세요.

서비스가 역할 생성을 지원하지 않는 경우에는 IAM을 사용하여 서비스 연결 역할을 생성할 수 있습니다.

> **Important**
>
> 서비스 연결 역할은 AWS 계정의 IAM 역할 제한을 계산하는 데 포함되지만, 한도에 도달한 경우에도 계정에 서비스 연결 역할을 생성할 수 있습니다. 한도를 초과해도 생성할 수 있는 역할은 서비스 연결 역할뿐입니다.

서비스 연결 역할 만들기(콘솔)

IAM에서 서비스 연결 역할을 생성하기 전에, 연결된 서비스가 서비스 역할을 자동으로 생성하는지 확인합니다. 또한 서비스의 콘솔, API, CLI 등에서 역할을 생성할 수 있는지를 알아봅니다.

서비스 연결 역할을 만들려면(콘솔 사용)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할(Roles)을 선택합니다. 그런 다음 역할 생성을 선택합니다.
3. AWS 서비스 역할 유형을 선택합니다.
4. 서비스의 사용 사례를 선택합니다. 사용 사례는 서비스에 필요한 신뢰 정책을 포함하기 위해 서비스에서 정합니다. 그리고 다음(Next)을 선택합니다.
5. 하나 이상의 권한 정책을 선택하여 역할에 연결합니다. 선택한 사용 사례에 따라 서비스에서 다음을 수행할 수 있습니다.

   - 역할에서 사용하는 권한을 정의할 수 있습니다.
   - 제한된 권한 세트에서 선택할 수 있습니다.
   - 모든 권한 중에서 선택할 수 있습니다.
   - 여기서 정책을 선택하지 않고, 나중에 정책을 만들어 역할에 연결할 수 있도록 허용합니다.

   역할에 부여하려는 권한을 할당하는 정책 옆의 확인란을 선택한 후 Next(다음)을 선택합니다.

> **Note**
>
> 지정하는 권한은 역할을 사용하는 모든 주체가 사용할 수 있습니다. 기본적으로 역할은 권한이 없습니다.

6. 역할 이름의 경우 역할 이름 사용자 지정 수준은 서비스에서 정합니다. 서비스에서 역할 이름을 정한 경우 이 옵션을 편집할 수 없습니다. 다른 경우에는 서비스에서 역할에 대한 접두사를 정의하고 사용자가 선택적 접미사를 입력할 수 있습니다.

   가능한 경우 기본 이름에 추가할 역할 이름 접미사를 입력합니다. 이 접미사는 이 역할의 목적을 파악하는 데 도움이 됩니다. 역할 이름은 AWS 계정 내에서 고유해야 합니다. 대소문자는 구별하지 않습니다. 예를 들어, 이름이 **<service-linked-role-name>_SAMPLE**과 **<service-linked-role-name>_sample**, 두 가지로 지정된 역할을 만들 수는 없습니다. 다양한 주체가 역할을 참조할 수 있기 때문에 역할이 생성된 후에는 역할 이름을 편집할 수 없습니다.
7. (선택 사항) 설명(Description)에서 새로운 서비스 연결 역할에 대한 설명을 편집합니다.
8. 생성하는 동안에는 서비스 연결 역할에 태그를 연결할 수 없습니다. IAM에서의 태그 사용에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요.
9. 역할을 검토한 다음 Create role을 선택합니다.

서비스 연결 역할 만들기(AWS CLI)

IAM에서 서비스 연결 역할을 만들기 전에, 연결된 서비스가 서비스 역할을 자동으로 생성하는지 그리고 서비스의 CLI에서 사용자가 역할을 만들 수 있는지를 확인합니다. 서비스 CLI가 지원되지 않는 경우 IAM 명령을 사용하여 서비스가 역할을 위임하는 데 필요한 인라인 정책과 신뢰 정책을 포함하는 서비스 연결 역할을 생성할 수 있습니다.

서비스 연결 역할(AWS CLI)을 만들려면

다음 명령을 실행합니다:

```
aws iam create-service-linked-role --aws-service-name SERVICE-NAME.amazonaws.com
```

서비스 연결 역할 만들기(AWS API)

IAM에서 서비스 연결 역할을 만들기 전에, 연결된 서비스가 서비스 역할을 자동으로 생성하는지 그리고 서비스의 API에서 사용자가 역할을 만들 수 있는지를 확인합니다. 서비스 API가 지원되지 않는 경

우 AWS API를 사용하여 서비스가 역할을 위임하는 데 필요한 인라인 정책과 신뢰 정책을 포함하는 서비스 연결 역할을 만들 수 있습니다.

서비스 연결 역할(AWS API)을 만들려면

CreateServiceLinkedRole API 호출을 사용합니다. 요청 시 *SERVICE_NAME_URL*`.amazonaws.com` 서비스 이름을 지정합니다.

예를 들어 Lex 봇 서비스 연결 역할을 만들려면 `lex.amazonaws.com`을 사용합니다.

타사 ID 제공업체의 역할 생성(페더레이션)

AWS 계정에 속하는 IAM 사용자를 생성하는 대신에 자격 증명 공급자를 사용할 수 있습니다. 자격 증명 공급자(IdP)를 사용하면 AWS 외부의 사용자 자격 증명을 관리할 수 있고 이 외부 사용자 자격 증명에 계정의 AWS 리소스에 대한 사용 권한을 부여할 수 있습니다. 연동 및 자격 증명 공급자에 대한 자세한 내용은 자격 증명 공급자 및 페더레이션 섹션을 참조하세요.

페더레이션 사용자의 역할 만들기(콘솔)

페더레이션 사용자의 역할을 만드는 절차는 서드 파티 공급자들의 선택에 따라 다릅니다.

- OIDC(OpenID Connect)에 대한 내용은 OpenID Connect 페더레이션을 위한 역할 생성(콘솔) 항목을 참조하세요.
- SAML 2.0은 SAML 2.0 페더레이션을 위한 역할 생성(콘솔) 섹션을 참조하세요.

페더레이션 액세스의 역할 만들기(AWS CLI)

AWS CLI에서 지원되는 자격 증명 공급자(OIDC 또는 SAML)의 역할을 만드는 절차는 동일합니다. 차이는 필수 선행 단계에서 생성하는 신뢰 정책의 내용에 있습니다. 사용하고 있는 공급자의 유형에 대한 필수 선행 조건 섹션에 나와 있는 절차에서부터 시작하세요.

- OIDC 공급자의 경우 OIDC 역할 생성을 위한 사전 조건 섹션을 참조하세요.
- SAML 공급자의 경우 SAML 역할을 생성하기 위한 사전 조건 섹션을 참조하세요.

AWS CLI에서 역할을 만들려면 여러 단계를 거쳐야 합니다. 콘솔을 사용하여 역할을 만들 때는 많은 단계가 자동으로 수행되지만 AWS CLI를 사용하면 각 단계를 직접 명시적으로 수행해야 합니다. 역할을 만든 다음 권한 정책을 역할에 할당해야 합니다. 선택적으로 역할에 대한 권한 경계를 설정할 수 있습니다.

아이덴티티 페더레이션을 위한 역할을 생성하려면(AWS CLI)

1. 역할 생성: aws iam create-role
2. 역할에 권한 정책 연결: aws iam attach-role-policy

   또는

   역할을 위한 인라인 권한 정책 생성: aws iam put-role-policy
3. (선택 사항) 태그를 연결하여 사용자 지정 속성을 역할에 추가: aws iam tag-role

   자세한 내용은 IAM 역할의 태그 관리(AWS CLI 또는 AWS API) 단원을 참조하십시오.
4. (선택 사항) 역할(aws iam put-role-permissions-boundary)에 대한 권한 경계를 설정합니다.

   이 권한 경계는 역할이 가질 수 있는 최대 권한을 관리합니다. 권한 경계는 고급 AWS 기능입니다.

다음 예는 단순한 환경에서 자격 증명 공급자를 생성하는 가장 일반적인 단계 중 첫 두 단계를 보여 줍니다. 이 예제는 123456789012 계정에 있는 모든 사용자가 역할을 수임하고 example_bucket Amazon S3 버킷을 볼 수 있도록 허용합니다. 또한 이 예는 Windows가 구동중인 컴퓨터에서 AWS CLI를 실행하고 있으며 자격 증명으로 AWS CLI를 이미 구성했다고 가정합니다. 자세한 내용은 AWS Command Line Interface 구성 섹션을 참조하세요.

다음 예는 사용자가 Amazon Cognito를 사용하여 로그인하는 경우 모바일 앱에 대해 설계되는 신뢰 정책을 보여줍니다. 이 예제에서 *us-east:12345678-ffff-ffff-ffff-123456*은 Amazon Cognito에 의해 할당된 자격 증명 풀 ID를 나타냅니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "RoleForCognito",
        "Effect": "Allow",
        "Principal": {"Federated": "cognito-identity.amazonaws.com"},
        "Action": "sts:AssumeRoleWithWebIdentity",
        "Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud": "us-
east:12345678-ffff-ffff-ffff-123456"}}
    }
}
```

다음 권한 정책에서는 역할을 수임하는 사용자가 example_bucket Amazon S3 버킷에서 ListBucket 작업만 수행하도록 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
}
```

이 Test-Cognito-Role 역할을 생성하기 위해서는 이전 신뢰 정책을 trustpolicyforcognitofederation.json 이름으로 이전 권한 정책을 permspolicyforcognitofederation.json 이름으로 로컬 policies 드라이브의 C: 폴더에 먼저 저장해야 합니다. 그러고 나면 다음 명령을 사용하여 역할을 만들고 인라인 정책을 연결합니다.

```
# Create the role and attach the trust policy that enables users in an account to
 assume the role.
$ aws iam create-role --role-name Test-Cognito-Role --assume-role-policy-document
 file://C:\policies\trustpolicyforcognitofederation.json

# Attach the permissions policy to the role to specify what it is allowed to do.
aws iam put-role-policy --role-name Test-Cognito-Role --policy-name
 Perms-Policy-For-CognitoFederation --policy-document file://C:\policies
\permspolicyforcognitofederation.json
```

페더레이션 액세스의 역할 만들기(AWS API)

AWS CLI에서 지원되는 자격 증명 공급자(OIDC 또는 SAML)의 역할을 만드는 절차는 동일합니다. 차이는 필수 선행 단계에서 생성하는 신뢰 정책의 내용에 있습니다. 사용하고 있는 공급자의 유형에 대한 필수 선행 조건 섹션에 나와 있는 절차에서부터 시작하세요.

- OIDC 공급자의 경우 OIDC 역할 생성을 위한 사전 조건 섹션을 참조하세요.
- SAML 공급자의 경우 SAML 역할을 생성하기 위한 사전 조건 섹션을 참조하세요.

아이덴티티 페더레이션을 위한 역할을 생성하려면(AWS API)

1. 역할 만들기: CreateRole
2. 역할에 권한 정책 연결: AttachRolePolicy

   또는

역할을 위한 인라인 권한 정책 생성: PutRolePolicy

3. (선택 사항) 태그를 연결하여 사용자 지정 속성을 사용자에게 추가: TagRole

   자세한 내용은 IAM 사용자의 태그 관리(AWS CLI 또는 AWS API) 단원을 참조하십시오.

4. (선택 사항) 역할(PutRolePermissionsBoundary)에 대한 권한 경계를 설정합니다.

   이 권한 경계는 역할이 가질 수 있는 최대 권한을 관리합니다. 권한 경계는 고급 AWS 기능입니다.

OpenID Connect 페더레이션을 위한 역할 생성(콘솔)

AWS 계정에 AWS Identity and Access Management 사용자를 생성하는 대신 OpenID Connect(OIDC) 페더레이션형 ID 제공업체를 사용할 수 있습니다. 자격 증명 공급자(IdP)를 사용하면 AWS 외부의 사용자 자격 증명을 관리할 수 있고 이 외부 사용자 자격 증명에 계정의 AWS 리소스에 대한 사용 권한을 부여할 수 있습니다. 페더레이션과 IdP에 대한 자세한 내용을 알아보려면 자격 증명 공급자 및 페더레이션 섹션을 참조하세요.

OIDC 역할 생성을 위한 사전 조건

OIDC 페더레이션을 위한 역할을 만들기 전에 먼저 다음 사전 조건 단계를 완료해야 합니다.

OIDC 페더레이션을 위한 역할 생성 준비

1. 페더레이션 OIDC ID를 제공하는 하나 이상의 서비스로 가입합니다. AWS 리소스로 액세스가 필요한 앱을 생성하면 공급자 정보로 앱도 구성합니다. 이렇게 하면 제공업체는 앱의 고유한 애플리케이션 및 시청자 ID를 제공합니다. (공급자마다 이 프로세스에 대해 다른 용어를 사용합니다. 이 가이드에서는 공급자에 앱을 식별하는 프로세스를 구성이라는 용어로 설명합니다.) 각 공급자로 여러 개의 앱을 구성하거나 단일 앱을 통해 다양한 공급자를 구성할 수 있습니다. 다음과 같이 ID 제공자에 대한 정보를 봅니다.

   - Login with Amazon 개발자 센터
   - Facebook 개발자 사이트의 앱 또는 웹 사이트에 Facebook 로그인 추가하기
   - Google 개발자 사이트의 OAuth 2.0을 사용한 로그인(OpenID Connect)

2. IdP로부터 필요한 정보를 받은 후 IAM에서 IdP를 생성합니다. 자세한 내용은 IAM에서 OIDC(OpenID Connect) ID 공급자 생성 단원을 참조하세요.

> **Important**
>
> Google, Facebook 또는 Amazon Cognito의 OIDC IdP를 사용하는 경우 AWS Management Console에서 IAM IdP를 별도로 생성하지 마세요. 이러한 OIDC ID 제공자는 이미 AWS에 내장되어 있고 용도에 맞게 사용할 수 있습니다. 이 단계를 건너뛰고 다음 단계에서 IdP를 사용하여 새 역할을 생성합니다.

3. IdP를 통해 인증된 사용자가 맡을 역할에 대한 정책을 준비합니다. 다른 어떤 역할과 마찬가지로 모바일 앱을 위한 역할에는 2개의 정책이 포함됩니다. 하나는 역할을 위임할 사용자를 지정하는 신뢰 정책입니다. 다른 하나는 모바일 앱의 액세스가 허용 또는 거부되는 AWS 작업 및 리소스를 지정하는 권한 정책입니다.

   웹 IdP의 경우, Amazon Cognito를 사용하여 보안 인증을 관리하는 것이 좋습니다. 이 경우 이 예제와 비슷한 신뢰 정책을 사용합니다.

   ```
   {
       "Version": "2012-10-17",
       "Statement": {
           "Effect": "Allow",
           "Principal": {"Federated": "cognito-identity.amazonaws.com"},
           "Action": "sts:AssumeRoleWithWebIdentity",
           "Condition": {
               "StringEquals": {"cognito-identity.amazonaws.com:aud": "us-
   east-2:12345678-abcd-abcd-abcd-123456"},
               "ForAnyValue:StringLike": {"cognito-identity.amazonaws.com:amr":
    "unauthenticated"}
           }
       }
   }
   ```

   `us-east-2:12345678-abcd-abcd-abcd-123456`을 Amazon Cognito에서 할당하는 아이덴티티 풀 ID로 바꿉니다.

   신뢰 정책을 생성할 때 OIDC IdP를 수동으로 구성하려면 자체 앱만이 이 역할을 수임한다고 보장하는 세 가지 값을 사용해야 합니다.

   - `Action` 요소에 대해서는 `sts:AssumeRoleWithWebIdentity` 작업을 사용하세요.
   - `Principal` 요소에 대해서는 `{"Federated":`*`providerUrl/providerArn`*`}` 문자열을 사용하세요.

- 일부 범용 OIDC IdP의 경우, *providerUrl*이 URL입니다. 다음 예제는 일부 범용 IdP에 대해 보안 주체를 지정하는 방법을 포함합니다.

  ```
  "Principal":{"Federated":"cognito-identity.amazonaws.com"}
  ```

  ```
  "Principal":{"Federated":"www.amazon.com"}
  ```

  ```
  "Principal":{"Federated":"graph.facebook.com"}
  ```

  ```
  "Principal":{"Federated":"accounts.google.com"}
  ```

- 다른 OIDC 제공업체의 경우, 다음 예제와 같이 Step 2에서 생성한 OIDC IdP의 Amazon 리소스 이름(ARN)을 사용합니다.

  ```
  "Principal":{"Federated":"arn:aws:iam::123456789012:oidc-provider/
  server.example.com"}
  ```

- 권한을 제한하려면 Condition 요소에 StringEquals 조건을 사용합니다. 자격 증명 풀 ID(Amazon Cognito용) 또는 앱 ID(다른 공급자용)를 테스트합니다. 아이덴티티 풀 ID는 IdP를 통해 앱을 구성할 때 얻은 앱 ID와 일치해야 합니다. 이러한 ID 간의 일치는 요청이 앱에서 오는지 확인합니다.

  > **Note**
  >
  > Amazon Cognito 자격 증명 풀의 IAM 역할은 서비스 보안 주체 cognito-identity.amazonaws.com의 역할 수임을 신뢰합니다. 이 유형의 역할에는 역할을 수임할 수 있는 보안 주체를 제한하기 위한 조건 키가 하나 이상 포함되어야 합니다. 크로스 계정 IAM 역할을 수임하는 Amazon Cognito 자격 증명 풀에는 추가 고려 사항이 적용됩니다. 이러한 역할의 신뢰 정책은 cognito-identity.amazonaws.com 서비스 보안 주체를 수락해야 하며 의도한 자격 증명 풀의 사용자로 역할 수임을 제한하는 aud 조건 키를 포함해야 합니다. 이 조건 없이 Amazon Cognito 자격 증명 풀을 신뢰하는 정책은 의도하지 않은 자격 증명 풀의 사용자가 역할을 맡을 수 있는 위험을 야기합니다. 자세한 내용은 Amazon Cognito 개발자 안내서의 Trust policies for IAM roles in Basic (Classic) authentication을 참조하세요.

  사용하는 IdP에 따라 다음 예제 중 하나와 비슷한 조건 요소를 생성합니다.

  ```
  "Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud":
  "us-east:12345678-ffff-ffff-ffff-123456"}}
  ```

```
"Condition": {"StringEquals": {"www.amazon.com:app_id":
"amzn1.application-oa2-123456"}}

"Condition": {"StringEquals": {"graph.facebook.com:app_id":
"111222333444555"}}

"Condition": {"StringEquals": {"accounts.google.com:aud":
"66677788899900pro0"}}
```

OIDC 공급자의 경우 다음 예시와 같이 aud 컨텍스트 키로 OIDC IdP의 정규화된 URL을 사용합니다.

```
"Condition": {"StringEquals": {"server.example.com:aud":
"appid_from_oidc_idp"}}
```

> **Note**
>
> 역할의 신뢰 정책에서 보안 주체에 대한 값은 하나의 IdP에 고유합니다. OIDC 역할은 오직 하나의 보안 주체만을 지정할 수 있습니다. 따라서 모바일 앱이 사용자에게 1개 이상의 IdP에서 로그인할 수 있게 허용한다면 지원하고자 하는 각각의 IdP에 대한 개별 역할을 생성합니다. 각 IdP에 대한 개별 신뢰 정책을 생성합니다.

사용자가 모바일 앱을 사용하여 Login with Amazon에서 로그인하는 경우 다음 예제 신뢰 정책이 적용됩니다. 예제에서 *amzn1.application-oa2-123456*은 Login with Amazon을 이용해 앱을 구성할 때 Amazon에서 할당하는 앱 ID를 나타냅니다.

```
{
      "Version": "2012-10-17",
      "Statement": [{
          "Sid": "RoleForLoginWithAmazon",
          "Effect": "Allow",
          "Principal": {"Federated": "www.amazon.com"},
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {"StringEquals": {"www.amazon.com:app_id":
 "amzn1.application-oa2-123456"}}
      }]
  }
```

사용자가 모바일 앱을 사용하여 Facebook에서 로그인하는 경우 다음 예제 신뢰 정책이 적용됩니다. 이 예제에서 *111222333444555*는 Facebook에서 할당하는 앱 ID를 나타냅니다.

```
{
      "Version": "2012-10-17",
      "Statement": [{
          "Sid": "RoleForFacebook",
          "Effect": "Allow",
          "Principal": {"Federated": "graph.facebook.com"},
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {"StringEquals": {"graph.facebook.com:app_id":
 "111222333444555"}}
      }]
  }
```

사용자가 모바일 앱을 사용하여 Google에서 로그인하는 경우 다음 예제 신뢰 정책이 적용됩니다. 이 예제에서 *666777888999000*은 Google에서 할당하는 앱 ID를 나타냅니다.

```
{
      "Version": "2012-10-17",
      "Statement": [{
          "Sid": "RoleForGoogle",
          "Effect": "Allow",
          "Principal": {"Federated": "accounts.google.com"},
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {"StringEquals": {"accounts.google.com:aud":
 "666777888999000"}}
      }]
  }
```

사용자가 모바일 앱을 사용하여 Amazon Cognito에서 로그인하는 경우 다음 예제 신뢰 정책이 적용됩니다. 이 예제에서 *us-east:12345678-ffff-ffff-ffff-123456*은 Amazon Cognito에서 할당하는 아이덴티티 풀 ID를 나타냅니다.

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

```
          "Sid": "RoleForCognito",
          "Effect": "Allow",
          "Principal": {"Federated": "cognito-identity.amazonaws.com"},
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud": "us-
east:12345678-ffff-ffff-ffff-123456"}}
      }]
  }
```

OIDC 역할 생성

사전 요구 사항을 완료한 후에는 IAM에서 역할을 생성할 수 있습니다. 다음 절차는 AWS Management Console에서 OIDC 페더레이션 역할을 생성하는 방법을 설명합니다. AWS CLI 또는 AWS API에 역할을 만들려면 타사 ID 제공업체의 역할 생성(페더레이션)의 절차 섹션을 참조하세요.

> **Important**
>
> Amazon Cognito를 사용하는 경우 Amazon Cognito 콘솔을 사용하여 역할을 설정합니다. 그 외에는 IAM 콘솔을 사용하여 OIDC 페더레이션의 역할을 생성합니다.

OIDC 페더레이션을 위한 IAM 역할 생성

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
3. 신뢰할 수 있는 엔터티 유형으로 웹 자격 증명을 선택하고 다음을 선택합니다.
4. 자격 증명 공급자(Identity provider)에서 역할의 IdP를 선택합니다.
   - 개별 웹 IdP에 대한 역할을 생성하는 경우, Login with Amazon, Facebook 또는 Google을 선택합니다.

     > **Note**
     >
     > 지원할 각 IdP에 대해 별도의 역할을 생성해야 합니다.

   - Amazon Cognito의 고급 시나리오 역할을 생성하려는 경우 Amazon Cognito를 선택합니다.

> **Note**
>
> 고급 시나리오에서 작업하는 경우에만 Amazon Cognito에서 사용할 역할을 수동으로 생성해야 합니다. 그렇지 않은 경우 Amazon Cognito가 역할을 대신 생성할 수 있습니다. 자세한 내용은 Amazon Cognito 개발자 안내서의 자격 증명 풀(페더레이션 자격 증명) 외부 자격 증명 공급자를 참조하세요.

- GitHub 작업을 위한 역할을 만들려는 경우 먼저 GitHub OIDC 공급자를 IAM에 추가해야 합니다. IAM에 GitHub OIDC 공급자를 추가한 후 token.actions.githubusercontent.com을 선택합니다.

> **Note**
>
> GitHub의 OIDC를 페더레이션형 ID로 신뢰하도록 AWS를 구성하는 방법에 대한 자세한 내용을 알아보려면 GitHub Docs - Configuring OpenID Connect in Amazon Web Services(GitHub 문서 - Amazon Web Services에서 OpenID Connect 구성)를 참조하세요. GitHub용 IAM IdP와 관련된 역할에 대한 액세스를 제한하는 모범 사례에 대한 자세한 내용은 이 페이지의 GitHub OIDC ID 제공자의 역할 구성을 참조하세요.

5. 애플리케이션의 식별자를 입력합니다. 식별자의 레이블은 선택하는 제공업체를 기준으로 변경됩니다.

   - Login with Amazon용 역할을 생성하려는 경우 애플리케이션 ID(Application ID) 상자에 앱 ID를 입력합니다.
   - Facebook용 역할을 생성하려는 경우 애플리케이션 ID(Application ID) 상자에 앱 ID를 입력합니다.
   - Google용 역할을 생성하려는 경우 대상(Audience) 상자에 대상 사용자 이름을 입력합니다.
   - Amazon Cognito용 역할을 생성하려는 경우 Amazon Cognito 애플리케이션용으로 생성한 아이덴티티 풀의 ID를 자격 증명 풀 ID(Identity Pool ID) 상자에 입력합니다.
   - GitHub 작업을 위한 역할을 만들려는 경우 다음 세부 정보를 입력하세요.
     - 대상(Audience)에서 `sts.amazonaws.com`을 입력합니다.
     - GitHub 조직에 GitHub 조직 이름을 입력합니다. GitHub 조직 이름은 필수이며 대시(-)를 포함한 영숫자여야 합니다. GitHub 조직 이름에 와일드카드 문자(* 및?)는 사용할 수 없습니다.
     - (선택 사항) GitHub 리포지토리에 GitHub 리포지토리 이름을 입력합니다. 값을 지정하지 않을 경우 기본값은 와일드카드(*)입니다.

- (선택 사항) GitHub 브랜치에 GitHub 브랜치 이름을 입력합니다. 값을 지정하지 않을 경우 기본값은 와일드카드(*)입니다.

6. (선택 사항) 애플리케이션 사용자가 역할에서 부여한 권한을 사용하기 위해 충족해야 하는 추가 조건을 만들려면 조건(선택 사항)에서 조건 추가를 클릭합니다. 예를 들어, 특정 IAM 사용자 ID에만 AWS 리소스에 대한 액세스 권한을 부여하는 조건을 추가할 수 있습니다. 역할을 만든 후에 신뢰 정책에 조건을 추가할 수도 있습니다. 자세한 내용은 역할 트러스트 정책 업데이트 단원을 참조하십시오.
7. OIDC 정보를 검토하고 다음을 선택합니다.
8. IAM은 계정의 AWS관리형 또는 고객 관리형 정책 목록을 포함합니다. 권한 정책을 사용하기 위한 정책을 선택하거나 정책 생성(Create policy)을 선택하여 새 브라우저 탭을 열고 완전히 새로운 정책을 생성합니다. 자세한 내용은 IAM 정책 생성 단원을 참조하세요. 정책을 생성하면 탭을 닫고 원래 탭으로 돌아갑니다. OIDC 사용자에게 부여하려는 권한 정책 옆의 확인란을 선택합니다. 원할 경우, 여기서 정책을 선택하지 않고 나중에 정책을 만들어서 역할에 연결할 수 있습니다. 기본적으로 역할은 권한이 없습니다.
9. (선택 사항) 권한 경계를 선택합니다. 이는 고급 기능입니다.

   권한 경계(Permissions boundary) 섹션을 열고 최대 역할 권한을 관리하기 위한 권한 경계 사용(Use a permissions boundary to control the maximum role permissions)을 선택합니다. 정책을 선택하여 권한 경계를 사용하세요.
10. Next(다음)를 선택합니다.
11. 역할 이름(Role name)에 역할 이름을 입력합니다. 역할 이름은 AWS 계정 내에서 고유해야 합니다. 대소문자를 구분하지 않습니다. 예를 들어, 이름이 **PRODROLE**과 **prodrole**, 두 가지로 지정된 역할을 만들 수는 없습니다. 다른 AWS 리소스가 역할을 참조할 수 있기 때문에 역할을 생성한 후에는 역할 이름을 편집할 수 없습니다.
12. (선택 사항) 설명(Description)에 새 역할에 대한 설명을 입력합니다.
13. 역할에 대한 사용 사례와 권한을 편집하려면 1단계: 신뢰할 수 있는 엔터티 선택(Step 1: Select trusted entities) 또는 2단계: 권한 추가(Step 2: Add permissions) 섹션에서 편집(Edit)을 선택합니다.
14. (선택 사항) 역할에 메타데이터를 추가하려면 태그를 키-값 페어로 연결합니다. IAM에서의 태그 사용에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요.
15. 역할을 검토한 다음 Create role을 선택합니다.

GitHub OIDC ID 제공자의 역할 구성

GitHub를 OIDC ID 제공업체(idP)로 사용하는 경우 IAM IdP와 연결된 역할을 수임할 수 있는 엔터티를 제한하는 것이 좋습니다. 신뢰 정책에 조건문을 포함하면 특정 GitHub 조직, 리포지토리 또는 분기로 역할을 제한할 수 있습니다. 문자열 조건 연산자가 있는 조건 키 `token.actions.githubusercontent.com:sub`을(를) 사용하여 액세스를 제한할 수 있습니다. 특정 리포지토리 또는 GitHub 조직 내 리포지토리 또는 분기 세트로 조건을 제한하는 것이 좋습니다. GitHub의 OIDC를 페더레이션형 ID로 신뢰하도록 AWS를 구성하는 방법에 대한 자세한 내용을 알아보려면 GitHub Docs - Configuring OpenID Connect in Amazon Web Services(GitHub 문서 - Amazon Web Services에서 OpenID Connect 구성)를 참조하세요.

작업 워크플로나 OIDC 정책에서 GitHub 환경을 사용하는 경우 보안을 강화하기 위해 환경에 보호 규칙을 추가하는 것이 좋습니다. 배포 브랜치와 태그를 사용하여 환경에 배포할 수 있는 브랜치와 태그를 제한하세요. 보호 규칙을 사용하여 환경을 구성하는 방법에 대한 자세한 내용은 GitHub의 배포를 위한 환경 사용 문서의 배포 브랜치 및 태그를 참조하세요.

GitHub의 OIDC IdP가 사용자 역할의 신뢰할 수 있는 보안 주체인 경우 IAM은 역할 신뢰 정책 조건을 검사하여 조건 키 `token.actions.githubusercontent.com:sub`이(가) 맞는지, 또 그 값이 와일드카드 문자(* 및 ?)만이거나 무효는 아닌지 확인합니다. IAM은 신뢰 정책이 생성되거나 업데이트될 때 이 검사를 수행합니다. 조건 키`token.actions.githubusercontent.com:sub`이(가) 존재하지 않거나 키 값이 언급된 값 기준을 충족하지 않으면 요청이 실패하고 오류가 반환됩니다.

> **Important**
>
> 조건 키 `token.actions.githubusercontent.com:sub`을(를) 특정 조직이나 리포지토리에 한정하지 않으면 제어할 수 없는 조직이나 리포지토리의 GitHub 작업이 AWS 계정의 GitHub IAM IdP와 연결된 역할을 수임할 수 있습니다.

다음 예제 신뢰 정책은 정의된 GitHub 조직, 리포지토리 및 분기에 대한 액세스를 제한합니다. 아래 예에서 조건 키 `token.actions.githubusercontent.com:sub` 값은 GitHub에서 문서화한 기본 주제 값 형식입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Federated": "arn:aws:iam::012345678910:oidc-provider/
token.actions.githubusercontent.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
          "token.actions.githubusercontent.com:sub":
 "repo:GitHubOrg/GitHubRepo:ref:refs/heads/GitHubBranch"
        }
      }
    }
  ]
}
```

다음 예제 조건은 정의된 GitHub 조직 및 리포지토리에 대한 액세스를 제한하지만 리포지토리 내의 모든 분기에 대한 액세스 권한을 부여합니다.

```
"Condition": {
  "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
  },
  "StringLike": {
    "token.actions.githubusercontent.com:sub": "repo:GitHubOrg/GitHubRepo:*"
  }
}
```

다음 예제 조건은 정의된 GitHub 조직 내의 모든 리포지토리 또는 분기에 대한 액세스를 제한합니다. 조건 키 `token.actions.githubusercontent.com:sub`은(는) GitHub 조직 내에서 GitHub 작업에 대한 액세스를 제한하는 특정 값으로 제한하는 것이 좋습니다.

```
"Condition": {
  "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
  },
  "StringLike": {
    "token.actions.githubusercontent.com:sub": "repo:GitHubOrg/*"
  }
}
```

정책에서 조건 확인을 위해 사용 가능한 OIDC 페더레이션 키에 대한 자세한 정보는 AWS OIDC 페더레이션에서 사용 가능한 키 섹션을 참조하세요.

SAML 2.0 페더레이션을 위한 역할 생성(콘솔)

AWS 계정에 속하는 IAM 사용자를 생성하는 대신에 SAML 2.0 페더레이션을 사용할 수 있습니다. 자격 증명 공급자(IdP)를 사용하면 AWS 외부의 사용자 자격 증명을 관리할 수 있고 이 외부 사용자 자격 증명에 계정의 AWS 리소스에 대한 사용 권한을 부여할 수 있습니다. 연동 및 자격 증명 공급자에 대한 자세한 내용은 자격 증명 공급자 및 페더레이션 섹션을 참조하세요.

> **Note**
>
> 페더레이션 복원력을 개선하려면 여러 SAML 로그인 엔드포인트를 지원하도록 IdP 및 AWS 페더레이션을 구성하는 것이 좋습니다. 자세한 내용은 AWS 보안 블로그 문서 How to use regional SAML endpoints for failover(장애 조치에 리전 SAML 엔드포인트를 사용하는 방법)를 참조하세요.

SAML 역할을 생성하기 위한 사전 조건

SAML 2.0 페더레이션을 위한 역할을 만들기 전에 먼저 다음 필수 선행 단계를 완료해야 합니다.

SAML 2.0 연동을 위한 역할 생성을 준비하려면

1. SAML 기반 페더레이션을 위한 역할을 생성하기 전에 IAM에서 SAML 공급자를 만들어야 합니다. 자세한 내용은 IAM에서 SAML ID 공급자 생성 단원을 참조하십시오.
2. SAML 2.0 인증 사용자들이 수임할 역할에 대한 정책을 준비합니다. 다른 어떤 역할과 마찬가지로 SAML 연동을 위한 역할에는 2개의 정책이 포함됩니다. 하나는 역할을 맡을 수 있는 사용자를 지정하는 역할 신뢰 정책이고, 다른 하나는 페더레이션 사용자의 액세스가 허용 또는 거부되는 AWS 작업 및 리소스를 지정하는 IAM 권한 정책입니다.

   역할에 대한 신뢰 정책을 생성할 때 애플리케이션만 역할을 수임할 수 있도록 세 가지 값을 사용해야 합니다.

   - `Action` 요소에 대해서는 `sts:AssumeRoleWithSAML` 작업을 사용하세요.
   - `Principal` 요소에 대해서는 `{"Federated":`*`ARNofIdentityProvider`*`}` 문자열을 사용하세요. *`ARNofIdentityProvider`*를 Step 1에서 만든 SAML 자격 증명 공급자의 ARN으로 바꿉니다.
   - `Condition` 요소에 대해서는 `StringEquals` 조건을 사용하여 SAML 응답의 `saml:aud` 속성이 AWS에 대한 SAML 연동 엔드포인트와 일치하는지 테스트하세요.

다음 예는 SAML 페더레이션 사용자를 위해 설계된 신뢰 정책입니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
      "Effect": "Allow",
      "Action": "sts:AssumeRoleWithSAML",
      "Principal": {"Federated": "arn:aws:iam::account-id:saml-provider/PROVIDER-
NAME"},
      "Condition": {"StringEquals": {"SAML:aud": "https://signin.aws.amazon.com/
saml"}}
    }
  }
```

보안 주체 ARN을 IAM에서 만든 SAML 공급자의 실제 ARN으로 바꿉니다. ARN에는 고유의 계정 ID와 공급자 이름이 있습니다.

SAML 역할 생성

사전 조건 단계를 완료한 후에는 SAML 기반 연동을 위한 역할을 생성합니다.

SAML 기반 연동을 위한 역할을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할과 역할 생성을 차례대로 선택합니다.
3. SAML 2.0 federation(SAML 2.0 연동) 역할 유형을 선택합니다.
4. SAML 공급자 선택(Select a SAML provider)에서 역할의 제공업체를 선택합니다.
5. SAML 2.0 액세스 수준 방법을 선택합니다.
   - Allow programmatic access only(프로그래밍 방식의 액세스만 허용)을 선택하여 AWS API 또는 AWS CLI에서 프로그래밍 방식으로 위임할 수 있는 역할을 만듭니다.
   - 그런 다음 프로그래밍 방식 및 AWS Management Console 액세스 허용(Allow programmatic and AWS Management Console access)을 선택하여 AWS Management Console에서 프로그래밍 방식으로 수임할 수 있는 역할을 생성합니다.

이렇게 생성된 두 역할은 비슷하지만 콘솔에서 위임할 수도 있는 역할에는 특정 조건을 포함하는 신뢰 정책을 포함합니다. 이 조건은 SAML 대상(SAML:aud 속성)이 SAML에 대한 AWS 로그인 엔드포인트(https:///signin.aws.amazon.com/saml)로 설정되도록 명시적으로 보장합니다.

6. 프로그래밍 방식 액세스를 위한 역할을 만드는 경우, 속성 목록에서 속성을 선택합니다. 그런 다음 값(Value) 상자에 역할에 포함시킬 값을 입력합니다. 이렇게 하면 지정한 속성을 포함하는 SAML 인증 응답(어설션)을 소유한 자격 증명 공급자의 사용자로 역할 액세스가 제한됩니다. 하나 이상의 속성을 지정해야 역할이 조직의 일부 사용자 집합으로 제한됩니다.

   프로그래밍 방식 액세스 및 콘솔 액세스를 위한 역할을 만드는 경우, SAML:aud 속성이 자동으로 추가되고 AWS SAML 엔드포인트의 URL(https:////signin.aws.amazon.com/saml)로 설정됩니다.

7. 신뢰 정책에 속성 관련 조건을 더 추가하려면 조건(선택 사항)(Condition (optional))을 선택하고 추가 조건을 선택한 후 값을 지정합니다.

   > **Note**
   >
   > 이 목록에는 가장 많이 사용되는 SAML 속성이 포함되어 있습니다. IAM은 조건을 생성하는 데 사용할 수 있는 추가 속성을 지원합니다. 지원되는 속성 목록은 SAML 연동에 사용할 수 있는 키를 참조하세요. 목록에는 없지만 지원되는 SAML 속성의 조건이 필요한 경우, 해당 조건을 수동으로 추가할 수 있습니다. 이렇게 하려면 역할을 만든 후 신뢰 정책을 편집합니다.

8. SAML 2.0 신뢰 정보를 검토한 후 다음(Next)을 선택합니다.

9. IAM은 계정의 AWS관리형 또는 고객 관리형 정책 목록을 포함합니다. 권한 정책을 사용하기 위한 정책을 선택하거나 정책 생성(Create policy)을 선택하여 새 브라우저 탭을 열고 완전히 새로운 정책을 생성합니다. 자세한 내용은 IAM 정책 생성 단원을 참조하세요. 정책을 생성하면 탭을 닫고 원래 탭으로 돌아갑니다. OIDC 페더레이션 사용자에게 부여할 권한 정책 옆의 확인란을 선택합니다. 원할 경우, 여기서 정책을 선택하지 않고 나중에 정책을 만들어서 역할에 연결할 수 있습니다. 기본적으로 역할은 권한이 없습니다.

10. (선택 사항) 권한 경계를 선택합니다. 이는 고급 기능입니다.

    권한 경계(Permissions boundary) 섹션을 열고 최대 역할 권한을 관리하기 위한 권한 경계 사용(Use a permissions boundary to control the maximum role permissions)을 선택합니다. 정책을 선택하여 권한 경계를 사용하세요.

11. Next(다음)를 선택합니다.

12. 다음: 검토를 선택합니다.

13. 역할 이름(Role name)에 역할 이름을 입력합니다. 역할 이름은 AWS 계정 내에서 고유해야 합니다. 대소문자는 구별하지 않습니다. 예를 들어, 이름이 **PRODROLE**과 **prodrole**, 두 가지로 지정된 역할을 만들 수는 없습니다. 기타 AWS 리소스가 역할을 참조할 수 있기 때문에 역할이 생성된 후에는 역할 이름을 편집할 수 없습니다.
14. (선택 사항) 설명(Description)에 새 역할에 대한 설명을 입력합니다.
15. 1단계: 신뢰할 수 있는 엔터티 선택(Step 1: Select trusted entities) 또는 2단계: 권한 추가(Step 2: Add permissions) 섹션에서 편집(Edit)을 선택하여 역할에 대한 사용 사례와 권한을 편집합니다.
16. (선택 사항) 태그를 키 값 페어로 연결하여 메타데이터를 역할에 추가합니다. IAM에서의 태그 사용에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요.
17. 역할을 검토한 다음 Create role을 선택합니다.

역할을 만든 후, AWS에 대한 정보로 자격 증명 공급자 소프트웨어를 구성하여 SAML 신뢰를 완료합니다. 이 정보는 페더레이션 사용자가 사용했으면 하는 역할을 포함합니다. 이를 가리켜 IdP와 AWS 간 신뢰 당사자 신뢰 구성이라고 합니다. 자세한 내용은 신뢰 당사자 신뢰 및 클레임 추가를 통해 SAML 2.0 IdP 구성 단원을 참조하십시오.

사용자 지정 트러스트 정책을 사용하여 역할 생성

사용자 지정 신뢰 정책을 생성하여 액세스를 위임하고 다른 사람이 AWS 계정에서 작업을 수행하도록 허용할 수 있습니다. 자세한 내용은 IAM 정책 생성 단원을 참조하십시오.

역할을 사용해 권한을 위임하는 방법에 대한 자세한 내용은 역할 용어 및 개념 섹션을 참조하세요.

사용자 지정 신뢰 정책을 사용하여 IAM 역할 생성(콘솔)

AWS Management Console을 사용하여 IAM 사용자가 수임할 수 있는 역할을 생성할 수 있습니다. 예를 들면 프로덕션 환경에서 개발 환경을 격리하기 위해 조직이 여러 개의 AWS 계정을 갖고 있다고 가정합시다. 개발 계정의 사용자가 프로덕션 계정의 리소스에 액세스할 수 있도록 하는 역할 생성에 대한 개괄적 정보는 분리된 개발 및 프로덕션 계정을 사용한 예제 시나리오 섹션을 참조하세요.

사용자 지정 신뢰 정책을 사용하여 역할 생성(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 콘솔의 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
3. 사용자 지정 신뢰 정책(Custom trust policy) 역할 유형을 선택합니다.

4. 사용자 지정 신뢰 정책(Custom trust policy) 섹션에서 역할의 사용자 지정 신뢰 정책을 입력하거나 붙여 넣습니다. 자세한 내용은 IAM 정책 생성 단원을 참조하십시오.
5. 정책 검증 동안 생성된 모든 보안 경고, 오류 또는 일반 경고를 해결하고 다음을 선택합니다.
6. (선택 사항) 권한 경계를 선택합니다. 이는 서비스 역할에서 가능한 고급 기능이며 서비스 링크된 역할은 아닙니다.

   권한 경계(Permissions boundary) 섹션을 열고 최대 역할 권한을 관리하기 위한 권한 경계 사용(Use a permissions boundary to control the maximum role permissions)을 선택합니다. IAM은 계정의 AWS관리형 또는 고객 관리형 정책 목록을 포함합니다. 정책을 선택하여 권한 경계를 사용하세요.
7. Next(다음)를 선택합니다.
8. 역할 이름의 경우 역할 이름 사용자 지정 수준은 서비스에서 정합니다. 서비스에서 역할 이름을 정한 경우 이 옵션을 편집할 수 없습니다. 다른 경우에는 서비스에서 역할 이름의 접두사를 정의하고 사용자가 선택적 접미사를 입력할 수 있습니다. 일부 서비스는 역할의 전체 이름을 지정할 수 있습니다.

   가능하다면 역할 이름 또는 역할 이름 접미사를 입력합니다. 역할 이름은 AWS 계정 내에서 고유해야 합니다. 대소문자는 구별하지 않습니다. 예를 들어, 이름이 **`PRODROLE`**과 **`prodrole`**, 두 가지로 지정된 역할을 만들 수는 없습니다. 기타 AWS 리소스가 역할을 참조할 수 있기 때문에 역할이 생성된 후에는 역할 이름을 편집할 수 없습니다.
9. (선택 사항) 설명(Description)에 새 역할에 대한 설명을 입력합니다.
10. (선택 사항) 1단계: 신뢰할 수 있는 엔터티 선택 또는 2단계: 권한 추가 섹션에서 편집을 선택하여 역할의 사용자 지정 정책과 권한을 편집합니다.
11. (선택 사항) 태그를 키 값 페어로 연결하여 메타데이터를 역할에 추가합니다. IAM에서의 태그 사용에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요.
12. 역할을 검토한 다음 Create role을 선택합니다.

액세스 권한 위임을 위한 정책의 예

다음 예제는 AWS 계정의 리소스에 대한 액세스를 AWS 계정에게 허용 또는 부여하는 방법을 보여줍니다. 이러한 예제 JSON 정책 문서를 사용하여 IAM 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

주제

- 역할을 사용하여 다른 AWS 계정의 리소스에 대한 액세스 권한 위임

- 정책을 사용하여 서비스에 대한 액세스 권한 위임
- 리소스 기반의 정책을 사용하여 다른 계정의 Amazon S3 버킷에 대한 액세스 권한 위임
- 리소스 기반의 정책을 사용하여 다른 계정의 Amazon SQS 대기열에 대한 액세스 권한 위임
- 계정이 액세스 거부될 경우 액세스 권한을 위임할 수 없음

역할을 사용하여 다른 AWS 계정의 리소스에 대한 액세스 권한 위임

IAM 역할을 사용하여 한 계정의 사용자에게 다른 계정의 AWS 리소스에 대한 액세스 권한을 부여하는 방법에 대한 자세한 내용은 튜토리얼: IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임 섹션을 참조하세요.

> **Important**
>
> 역할 신뢰 정책의 `Principal` 요소에 특정 역할이나 사용자에 대한 ARN을 포함할 수 있습니다. 정책을 저장하면 AWS가 ARN을 고유한 보안 주체 ID로 변환합니다. 그러면 누군가가 해당 역할 또는 사용자를 제거하고 다시 만들어 본인의 권한을 에스컬레이션할 위험을 완화할 수 있습니다. 일반적으로 콘솔에서는 이 ID가 보이지 않습니다. 신뢰 정책이 표시될 때 해당 ARN으로 다시 역변환되기 때문입니다. 그러나 해당 역할 또는 사용자를 삭제하면 관계가 깨집니다. 사용자 또는 역할을 다시 만들더라도 해당 정책이 더 이상 적용되지 않습니다. 신뢰 정책에 저장된 보안 주체 ID와 일치하지 않기 때문입니다. 이 경우 보안 주체 ID가 콘솔에 표시됩니다. AWS에서 더 이상 이를 ARN에 다시 매핑할 수 없기 때문입니다. 결과적으로 신뢰 정책의 `Principal` 요소에서 참조된 사용자 또는 역할을 삭제하고 다시 생성하는 경우, ARN을 바꾸도록 역할을 편집해야 합니다. 그러면 정책을 저장할 때 ARN이 새 보안 주체 ID로 변환됩니다.

정책을 사용하여 서비스에 대한 액세스 권한 위임

다음 예제는 역할에 연결할 수 있는 정책을 보여줍니다. 이 정책은 Amazon EMR 서비스와 AWS Data Pipeline 서비스가 역할을 수행할 수 있도록 합니다. 그러면 서비스가 해당 역할에 할당된 권한 정책에서 부여한 모든 작업을 수행할 수 있습니다(표시되지 않음). 여러 서비스 보안 주체를 지정할 때 `Service` 요소를 두 개 지정하면 안 됩니다. 하나만 지정할 수 있습니다. 대신 여러 서비스 보안 주체의 배열을 하나의 `Service` 요소의 값으로 사용합니다.

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

```
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "elasticmapreduce.amazonaws.com",
          "datapipeline.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

리소스 기반의 정책을 사용하여 다른 계정의 Amazon S3 버킷에 대한 액세스 권한 위임

이 예에서 계정 A는 리소스 기반 정책(Amazon S3 버킷 정책)을 사용하여 계정 B에게 계정 A의 S3 버킷에 액세스할 수 있는 완전한 권한을 부여합니다. 그런 다음 계정 B는 IAM 사용자 정책을 생성하여 계정 A의 버킷에 대한 해당 액세스 권한을 계정 B의 사용자 중 하나에게 위임합니다.

계정 A의 S3 버킷 정책은 다음 정책과 같을 수 있습니다. 이 예에서 계정 A의 S3 버킷 이름은 mybucket이고, 계정 B의 계정 번호는 111122223333입니다. 계정 B에서는 개별 사용자 또는 그룹을 지정하지 않고 오직 계정 자체만 지정할 뿐입니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "AccountBAccess1",
    "Effect": "Allow",
    "Principal": {"AWS": "111122223333"},
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::mybucket",
      "arn:aws:s3:::mybucket/*"
    ]
  }
}
```

또는 계정 A가 Amazon S3 액세스 제어 목록(ACL)을 사용하여 계정 B에 S3 버킷 또는 버킷 내 단일 객체에 대한 액세스 권한을 부여할 수 있습니다. 이 경우 유일한 변경 사항은 계정 A가 계정 B에게 액세스 권한을 부여하는 방식입니다. 이 예의 다음 부분에서 설명한 것처럼 계정 B는 여전히 정책을 사용하여 계정 B의 IAM 그룹에 액세스 권한을 위임합니다. S3 버킷 및 객체에 대한 액세스를 관리하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 액세스 제어를 참조하세요.

계정 B의 관리자는 다음 정책 샘플을 생성할 수 있습니다. 이 정책은 계정 B의 그룹 또는 사용자에게 읽기 액세스를 허용하며, 이전 정책은 B 계정에 대한 액세스 권한을 부여합니다. 하지만 계정 B의 개별 그룹과 사용자는 그룹 또는 사용자 정책이 리소스에 대한 권한을 명시적으로 부여할 때까지는 그 리소스에 액세스할 수 없습니다. 이 정책의 권한은 이전 크로스 계정 정책에 있는 권한의 하위 집합에 불과할 수 있습니다. 계정 B는 첫 번째 정책에서 계정 A가 계정 B에게 부여한 권한보다 더 많은 권한을 자신의 그룹 또는 사용자에게 위임할 수 없습니다. 이 정책에서 `Action` 요소는 `List` 작업만을 허용하도록 명시적으로 정의되고 이 정책의 `Resource` 요소는 계정 A에 의해 적용되는 버킷 정책의 `Resource`와 일치합니다.

이 정책을 적용하기 위해 계정 B는 IAM을 사용하여 이 정책을 계정 B의 해당 사용자(또는 그룹)에게 연결합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:List*",
    "Resource": [
      "arn:aws:s3:::mybucket",
      "arn:aws:s3:::mybucket/*"
    ]
  }
}
```

리소스 기반의 정책을 사용하여 다른 계정의 Amazon SQS 대기열에 대한 액세스 권한 위임

다음 예에서 계정 A에는 계정 B에 대한 액세스 권한을 대기열에 부여하기 위해 대기열에 연결된 리소스 기반 정책을 사용하는 Amazon SQS 대기열이 있습니다. 그러면 계정 B는 IAM 그룹 정책을 사용하여 계정 B의 그룹에 액세스 권한을 위임합니다.

다음 대기열 정책 예에서는 계정 B에 계정 A의 대기열 queue1에 대한 `SendMessage` 및 `ReceiveMessage` 작업을 2014년 11월 30일 정오부터 오후 3시까지만 수행할 수 있는 권한을 부여합니다. 계정 B의 계정 번호는 1111-2222-3333입니다. 계정 A는 Amazon SQS를 사용하여 이 정책을 구현합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "111122223333"},
    "Action": [
```

```
      "sqs:SendMessage",
      "sqs:ReceiveMessage"
    ],
    "Resource": ["arn:aws:sqs:*:123456789012:queue1"],
    "Condition": {
      "DateGreaterThan": {"aws:CurrentTime": "2014-11-30T12:00Z"},
      "DateLessThan": {"aws:CurrentTime": "2014-11-30T15:00Z"}
    }
  }
}
```

계정 B의 그룹에게 액세스 권한을 위임하기 위한 계정 B의 정책은 다음 예와 같을 수 있습니다. 계정 B는 IAM을 사용하여 이 정책을 그룹(또는 사용자)에게 연결합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sqs:*",
    "Resource": "arn:aws:sqs:*:123456789012:queue1"
  }
}
```

앞의 IAM 사용자 정책에 대한 예제에서 계정 B는 와일드카드를 사용하여 해당 사용자에게 계정 A의 대기열에서 모든 Amazon SQS 작업을 수행할 수 있는 액세스 권한을 부여했습니다. 하지만 계정 B는 액세스 권한이 부여된 범위까지만 액세스 권한을 위임할 수 있습니다. 두 번째 정책이 있는 계정 B 그룹은 2014년 11월 30일 정오부터 오후 3시까지만 대기열에 액세스할 수 있습니다. 사용자는 계정 A의 Amazon SQS 대기열 정책에 정의된 대로 `SendMessage` 및 `ReceiveMessage` 작업만 수행할 수 있습니다.

계정이 액세스 거부될 경우 액세스 권한을 위임할 수 없음

다른 계정에서 사용자의 상위 계정에 대한 액세스를 명시적으로 거부할 경우 AWS 계정은 다른 계정의 리소스에 대한 액세스 권한을 위임할 수 없습니다. 이 거부는 사용자가 액세스 권한을 부여하는 기존 정책을 가지고 있는지 여부에 상관없이 해당 계정의 사용자에게 전파됩니다.

계정 A가 계정 A의 S3 버킷에 계정 A의 버킷에 대한 계정 B의 액세스를 명시적으로 거부하는 버킷 정책을 계정 A의 S3 버킷에 작성하는 경우를 예로 들어 보겠습니다. 계정 B는 계정 B의 사용자에게 계정 A의 버킷에 대한 액세스 권한을 부여하는 IAM 사용자 정책을 작성합니다. 계정 A의 S3 버킷에 적용된 명시적 거부는 계정 B의 사용자에게 전파되고 계정 B의 사용자에게 액세스 권한을 부여하는 IAM 사용자 정책보다 우선합니다. (권한 평가 방식에 대한 자세한 내용은 정책 평가 로직 섹션을 참조하세요.)

계정 A의 버킷 정책은 다음과 같을 수 있습니다. 이 예에서 계정 A의 S3 버킷 이름은 mybucket이고, 계정 B의 계정 번호는 1111-2222-3333입니다. 계정 A는 Amazon S3를 사용하여 이 정책을 구현합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "AccountBDeny",
    "Effect": "Deny",
    "Principal": {"AWS": "111122223333"},
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::mybucket/*"
  }
}
```

이 명시적 거부는 계정 A의 S3 버킷에 액세스할 수 있는 권한을 제공하는 계정 B의 모든 정책을 재정의합니다.

IAM 역할 관리

사용자, 애플리케이션 또는 서비스에서 이전에 생성한 역할을 사용하려면 그 역할로 전환할 수 있는 권한을 부여해야 합니다. 그룹 또는 사용자에게 추가된 어떤 정책도 필요한 권한을 부여하는 데 사용할 수 있습니다. 이 섹션에서는 사용자에게 역할을 사용할 권한을 부여하는 방법을 설명합니다. 또한 사용자가 AWS Management Console, Tools for Windows PowerShell, AWS Command Line Interface(AWS CLI) 및 `AssumeRole` API에서 역할로 전환하는 방법도 설명합니다.

> **Important**
>
> IAM 콘솔 대신 프로그래밍 방식으로 역할을 생성하는 경우에는 사용자의 선택에 따라 최대 64자인 `RoleName`뿐만 아니라 최대 512자인 `Path`도 추가할 수 있습니다. 그러나 AWS Management Console에서 역할 전환(Switch Role) 기능이 있는 역할을 사용하려면 `Path`와 `RoleName`을 합해 64자를 초과할 수 없습니다.

주제

- 역할 액세스 보기
- 액세스 정보를 기반으로 정책 생성
- 사용자에게 역할을 전환할 권한 부여
- 사용자에게 AWS 서비스에 역할을 전달할 권한 부여

- IAM 역할 임시 보안 자격 증명 취소
- 서비스 연결 역할 업데이트
- 역할 트러스트 정책 업데이트
- 역할 권한 업데이트
- 역할 설정 업데이트
- 역할 또는 인스턴스 프로파일 삭제

역할 액세스 보기

역할에 대한 권한을 변경하기 전에 최근 서비스 수준 활동을 검토해야 합니다. 이 기능은 사용 중인 보안 주체(사람 또는 애플리케이션)의 액세스 권한을 제거하지 않으려는 경우 중요합니다. 마지막으로 액세스한 정보 보기에 대한 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 섹션을 참조하세요.

액세스 정보를 기반으로 정책 생성

때로는 IAM 엔터티(사용자 또는 역할)에 필요한 것보다 많은 권한을 부여할 수도 있습니다. 부여하는 권한을 세분화할 수 있도록 엔터티의 액세스 활동을 기반으로 IAM 정책을 생성할 수 있습니다. IAM Access Analyzer는 사용자의 AWS CloudTrail 로그를 검토하고 지정된 날짜 범위에 엔터티에서 사용한 권한을 포함하는 정책 템플릿을 생성합니다. 이 템플릿을 사용하여 세분화된 권한이 포함된 관리형 정책을 생성한 다음 IAM 엔터티에 연결할 수 있습니다. 이렇게 하면 특정 사용 사례에 사용자나 역할이 AWS 리소스와 상호 작용하는 데 필요한 권한만 부여됩니다. 자세한 내용은 액세스 활동을 기반으로 정책 생성 섹션을 참조하세요.

사용자에게 역할을 전환할 권한 부여

관리자가크로스 계정 액세스를 위한 역할을 생성할 때는 역할 및 리소스를 소유하는 계정(신뢰하는 계정)과 사용자를 포함하는 계정(신뢰받는 계정) 간의 신뢰를 설정합니다. 이렇게 하려면 신뢰하는 계정의 관리자가 역할의 신뢰 정책에 신뢰받는 계정 번호를 `Principal`로 지정합니다. 이렇게 하면 잠재적으로 신뢰할 수 있는 계정의 모든 사용자가 역할을 수임할 수 있습니다. 구성을 완료하려면 신뢰 받는 계정의 관리자는 계정에 속한 특정 그룹 또는 사용자에게 역할 전환 권한을 부여해야 합니다.

역할로 전환할 수 있는 권한을 부여하려면 다음을 수행하세요.

1. 신뢰할 수 있는 계정의 관리자로서 사용자에 대한 새 정책을 생성하거나 기존 정책을 편집하여 필요한 요소를 추가합니다. 세부 정보는 정책 생성 또는 편집을 참조하세요.
2. 그런 다음 역할 정보를 공유할 방법을 선택합니다.

- Role link(역할 링크): 이미 세부 정보가 모두 작성되어 있는 Switch Role(역할 전환) 페이지로 이동하는 링크를 사용자에게 보냅니다.
- Account ID or alias(계정 ID 또는 별칭): 각 사용자에게 계정 ID 번호나 계정 별칭과 함께 역할 이름을 제공합니다. 이제 사용자는 역할 전환 페이지로 이동하여 세부 정보를 직접 입력합니다.

세부 정보는 사용자에게 정보 제공을 참조하세요.

IAM 사용자, SAML 페더레이션 역할 또는 웹 아이덴티티 페더레이션 역할로 로그인할 때만 역할을 전환할 수 있습니다. AWS 계정 루트 사용자로 로그인할 때는 역할을 바꿀 수 없습니다.

Important

AWS Management Console에서의 역할을 ExternalId 값이 필요한 역할로 전환할 수 없습니다. `ExternalId` 파라미터를 지원하는 `AssumeRole` API를 호출해야만 이러한 역할로 전환할 수 있습니다.

참고

- 이 주제에서는 최종적으로 사용자에게 태스크를 수행할 수 있는 권한을 부여하기 때문에 사용자에 대한 정책에 대해 설명합니다. 그러나 개별 사용자에게 직접 권한을 부여하지 않는 것이 좋습니다. 사용자가 역할을 수임하면 해당 역할과 관련된 권한이 할당됩니다.
- AWS Management Console에서 역할을 전환하는 경우, 콘솔은 항상 원래 자격 증명을 사용하여 전환을 승인합니다. 이는 IAM 사용자, SAML 페더레이션 역할 또는 웹 자격 증명 페더레이션 역할 중 어느 것으로 로그인하는지 여부에 관계없이 적용됩니다. 예를 들어, RoleA로 전환하는 경우 IAM에서는 원래 사용자 보안 인증 또는 페더레이션 역할 보안 인증을 사용하여 RoleA를 부여할지 여부를 결정합니다. RoleA를 사용하는 중에 RoleB로 전환하려는 경우, 원래 사용자 또는 페더레이션 역할 자격 증명이 인증에 사용됩니다. RoleA의 자격 증명은 이 작업에 사용되지 않습니다.

주제

- 정책 생성 또는 편집
- 사용자에게 정보 제공

정책 생성 또는 편집

역할을 맡기 위한 사용자 권한을 부여하는 정책에는 다음에 적용되는 `Allow` 문이 포함되어야 합니다.

- `sts:AssumeRole` 작업
- `Resource` 요소에 있는 역할의 ARN(Amazon Resource Name)

정책을 가져오는 사용자는 나열된 리소스에 대한 역할을 전환할 수 있습니다(그룹 멤버십 또는 직접 연결을 통해).

> **Note**
>
> `Resource`를 *로 설정하는 경우 사용자가 사용자 계정을 신뢰하는 어떤 계정의 어떤 역할이라도 수임할 수 있습니다. (즉, 역할의 신뢰 정책에서 사용자의 계정을 `Principal`로 지정합니다.) 최소 권한의 원칙에 따라 사용자에게 필요한 역할에 대해서만 완전한 ARN을 지정하는 것이 좋습니다.

다음 예제에서는 단 한 개의 계정에서 사용자가 역할을 맡을 수 있는 정책을 보여 줍니다. 또한 이 정책은 와일드카드(*)를 사용하여 역할 이름이 `Test` 문자로 시작할 경우에만 사용자가 역할을 전환할 수 있도록 지정합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::account-id:role/Test*"
  }
}
```

> **Note**
>
> 이 역할이 사용자에게 부여하는 권한은 사용자에게 이미 부여된 권한에 추가되지는 않습니다. 사용자가 어떤 역할로 전환할 때는 일시적으로 자신의 원래 권한을 버리고 그 역할이 부여하는 권한으로 갈아탑니다. 사용자가 역할을 끝내면 원래 사용자 권한이 자동으로 회복됩니다. 사용자 권한에서 Amazon EC2 인스턴스 작업을 허용하지만, 역할의 권한 정책이 해당 권한을 부여하지 않는 경우를 예로 들어 보겠습니다. 이 경우 역할을 사용할 때 사용자가 콘솔에서

Amazon EC2 인스턴스 작업을 수행할 수 없습니다. 또한 AssumeRole을 통해 얻은 임시 자격 증명은 프로그래밍 방식으로 Amazon EC2 인스턴스 작업을 수행할 수 없습니다.

사용자에게 정보 제공

역할을 만들어 이 역할로 전환하는 권한을 사용자에게 부여한 후, 사용자에게 다음을 제공해야 합니다.

- 역할의 이름
- 해당 역할을 포함하는 계정의 ID 또는 별칭

계정 ID와 역할 이름으로 미리 구성된 링크를 보내 사용자의 액세스를 간소화할 수 있습니다. 역할 생성 마법사를 완료한 후 역할 보기 배너를 선택하거나 크로스 계정이 활성화된 역할에 대한 역할 요약 페이지에서 역할 링크를 볼 수 있습니다.

다음 형식을 사용해 링크를 수동으로 구축할 수도 있습니다. 계정 ID 또는 별칭과 역할 이름을 다음 예제의 파라미터 두 개로 대치합니다.

```
https://signin.aws.amazon.com/switchrole?
account=your_account_ID_or_alias&roleName=optional_path/role_name
```

사용자를 역할로 전환(콘솔) 섹션으로 안내하여 프로세스를 살펴보도록 하는 것이 좋습니다. 역할을 수임할 때 발생할 수 있는 일반적인 문제를 해결하려면 역할을 수임할 수 없음 섹션을 참조하세요.

고려 사항

- 프로그래밍 방식으로 역할을 생성하는 경우에는, 경로와 이름을 사용하여 역할을 생성할 수 있습니다. 이 경우 사용자에게 전체 경로와 역할 이름을 제공해야만 사용자가 AWS Management Console의 역할 전환 페이지에 입력할 수 있습니다. 예: division_abc/subdivision_efg/role_XYZ.
- 프로그래밍 방식으로 역할을 생성하는 경우에는 최대 512자의 Path와 RoleName을 추가할 수 있습니다. 역할 이름은 최대 64자일 수 있습니다. 그러나 AWS Management Console에서 역할 전환(Switch Role) 기능을 통해 역할을 사용하려면 Path와 RoleName을 합해 64자를 초과할 수 없습니다.

- 보안을 위해 AWS CloudTrail 로그를 검토하여 AWS에서 누가 작업을 수행했는지 확인할 수 있습니다. 역할 신뢰 정책에서 sts:SourceIdentity 조건 키를 사용하여 사용자가 역할을 수임할 때 자격 증명을 지정하도록 요구할 수 있습니다. 예를 들어 IAM 사용자가 자신의 사용자 이름을 소스 자격 증명으로 지정하도록 요구할 수 있습니다. 이를 통해 AWS에서 특정 작업을 수행한 사용자를 확

인할 수 있습니다. 자세한 내용은 sts:SourceIdentity 단원을 참조하십시오. 역할 신뢰 정책에서 sts:RoleSessionName을 사용하여 사용자가 역할을 수임할 때 세션 이름을 지정하도록 요구할 수 있습니다. 이렇게 하면 여러 보안 주체가 한 역할을 사용할 때 역할 세션 간을 구분할 수 있습니다.

사용자에게 AWS 서비스에 역할을 전달할 권한 부여

다수의 AWS 서비스를 구성하려면 IAM 역할을 서비스에 전달해야 합니다. 그러면 서비스가 나중에 역할을 수임하고 사용자 대신 작업을 수행할 수 있습니다. 대부분의 서비스에서 설정 중 한 번만 서비스에 역할을 전달하면 됩니다. 서비스가 역할을 수임할 때마다 전달할 필요가 없습니다. 예를 들어 Amazon EC2 인스턴스에서 실행 중인 애플리케이션이 있다고 가정합니다. 해당 애플리케이션에는 인증을 위한 임시 자격 증명과 AWS에서 작업을 수행할 수 있는 애플리케이션을 승인할 권한이 필요합니다. 애플리케이션을 설정할 때는 역할을 Amazon EC2에 전달하여 해당 자격 증명을 제공하는 인스턴스와 함께 사용해야 합니다. IAM 정책을 역할에 연결하여 인스턴스에서 실행 중인 애플리케이션에 대한 권한을 정의합니다. 애플리케이션은 역할이 허용하는 작업을 수행해야 할 때마다 역할을 수임합니다.

AWS 서비스에 역할(및 그 권한)을 전달하려면 사용자에게 서비스에 역할을 전달할 권한이 있어야 합니다. 이를 통해 관리자는 승인된 사용자만 권한이 부여된 역할을 통해 서비스를 구성하도록 할 수 있습니다. 사용자가 AWS 서비스에 역할을 전달하도록 하려면 해당 사용자의 IAM 사용자, 역할 또는 그룹에 `PassRole` 권한을 부여해야 합니다.

> **Warning**
>
> - 같은 AWS 계정을 공유하는 서비스로 IAM 역할을 전달하기 위해서는 `PassRole` 권한만 이용할 수 있습니다. 계정 A의 역할을 계정 B의 서비스에 넘겨주려면 먼저 계정 A의 역할을 맡을 수 있는 IAM 역할을 계정 B에 생성한 다음 계정 B의 역할을 서비스에 넘겨야 합니다. 세부 정보는 IAM의 크로스 계정 리소스 액세스을 참조하세요.
> - 역할에 태그를 지정한 다음 `iam:PassRole` 작업과 함께 정책의 `ResourceTag` 조건 키를 사용하는 것으로 역할을 전달할 수 있는 사람을 제어하려고 하지 마십시오. 이 접근법은 신뢰할 수 있는 결과를 가져오지 못합니다.

`PassRole` 권한을 설정할 때 사용자에게 부여하려는 역할보다 더 많은 권한이 있는 역할을 사용자가 넘기지 않도록 해야 합니다. 예를 들어 Alice가 Amazon S3 작업을 수행하는 것이 허용되지 않을 수 있습니다. Alice가 Amazon S3 작업을 허용하는 서비스에 역할을 넘길 수 있으면 해당 서비스에서 작업 실행 시 Alice를 대신하여 Amazon S3 작업을 수행할 수 있습니다.

서비스 연결 역할을 지정하는 경우 해당 역할을 서비스에 전달할 권한도 있어야 합니다. 일부 서비스는 서비스에서 작업을 수행할 때 계정에 서비스 연결 역할을 자동으로 생성합니다. 예를 들어, Amazon EC2 Auto Scaling에서는 사용자가 Auto Scaling 그룹을 처음으로 생성할 때 사용자를 대신해 `AWSServiceRoleForAutoScaling` 서비스 연결 역할을 생성합니다. Auto Scaling을 생성할 때 `iam:PassRole` 권한이 없는 상태에서 서비스 연결 역할을 지정하려고 하면 오류가 발생합니다. 역할을 명시적으로 지정하지 않는 경우에는 `iam:PassRole` 권한이 필요하지 않으며, 기본적으로 해당 그룹에 대해 수행하는 모든 작업에 `AWSServiceRoleForAutoScaling` 역할이 사용됩니다. 서비스 연결 역할을 지원하는 서비스를 알아보려면 AWS IAM으로 작업하는 서비스 섹션을 참조하세요. 서비스에서 작업 수행 시 자동으로 서비스 연결 역할을 생성하는 서비스를 알아보려면 예 링크를 선택하고 해당 서비스에 대한 서비스 연결 역할 설명서를 확인합니다.

사용자는 역할을 사용하여 서비스에 권한을 할당하는 API 작업에서 파라미터로 역할 ARN을 전달할 수 있습니다. 그런 다음 서비스는 해당 사용자에게 `iam:PassRole` 권한이 있는지 확인합니다. 사용자가 승인된 역할만 전달하도록 제한하려면 IAM 정책 문의 `Resources` 요소로 `iam:PassRole` 권한을 필터링하면 됩니다.

JSON 정책의 `Condition` 요소를 사용하여 모든 AWS 요청의 요청 컨텍스트에 포함된 키 값을 테스트할 수 있습니다. 정책에서 조건 키를 사용하는 방법에 대한 자세한 내용은 IAM JSON 정책 요소: Condition 섹션을 참조하세요. `iam:PassedToService` 조건 키는 역할을 전달할 수 있는 서비스의 서비스 보안 주체를 지정하는 데 사용될 수 있습니다. 정책에서 `iam:PassedToService` 조건 키를 사용하는 방법에 대한 자세한 내용은 iam:PassedToService를 참조하세요.

예 1

인스턴스를 시작한 후 사용자에게 Amazon EC2 서비스에 승인된 역할 집합을 전달할 수 있는 권한을 부여하려 한다고 가정하겠습니다. 다음 세 가지 요소가 필요합니다.

- 역할이 수행할 수 있는 작업을 결정하는 역할에 연결된 IAM 권한 정책입니다. 역할이 수행해야 하는 작업 및 역할이 그러한 작업을 수행하는 데 필요한 리소스만으로 권한을 한정할 수 있습니다. AWS 관리형 또는 고객이 생성한 IAM 권한 정책을 사용할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [ "A list of the permissions the role is allowed to use" ],
        "Resource": [ "A list of the resources the role is allowed to access" ]
    }
}
```

- 서비스에서 역할을 위임하도록 허용하는 역할에 대한 신뢰 정책입니다. 예를 들어, `UpdateAssumeRolePolicy` 작업이 있는 역할에 다음과 같은 신뢰 정책을 연결할 수 있습니다. 이 신뢰 정책을 통해 Amazon EC2는 해당 역할 및 역할에 연결된 권한을 사용할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "TrustPolicyStatementThatAllowsEC2ServiceToAssumeTheAttachedRole",
        "Effect": "Allow",
        "Principal": { "Service": "ec2.amazonaws.com" },
       "Action": "sts:AssumeRole"
    }
}
```

- 사용자가 승인된 역할만 전달하도록 허용하는 IAM 사용자에 연결된 IAM 권한 정책입니다. 일반적으로 `iam:GetRole`을(를) `iam:PassRole`에 추가하여 사용자가 전달할 역할의 세부 정보를 얻을 수 있습니다. 이 예제에서 사용자는 지정된 계정에 있으며 다음과 같이 이름이 `EC2-roles-for-XYZ-`(으)로 시작하는 역할만 전달할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iam:GetRole",
            "iam:PassRole"
        ],
        "Resource": "arn:aws:iam::account-id:role/EC2-roles-for-XYZ-*"
    }]
}
```

이제 사용자는 할당된 역할로 Amazon EC2 인스턴스를 시작할 수 있습니다. 이 인스턴스에서 실행되는 애플리케이션은 인스턴스 프로파일 메타데이터를 통해 역할의 임시 자격 증명에 액세스할 수 있습니다. 역할과 연결된 권한 정책은 인스턴스가 수행할 수 있는 작업을 결정합니다.

예제 2

Amazon Relational Database Service(Amazon RDS)는 Enhanced Monitoring(확장 모니터링)이라는 기능을 지원합니다. 이 기능을 사용하면 Amazon RDS에서 에이전트를 사용하여 데이터베이스 인스턴스를 모니터링할 수 있습니다. 또한 Amazon RDS가 Amazon CloudWatch Logs에 지표를 로그할 수도

있습니다. 이 기능을 사용하려면 서비스 역할을 생성하여 지표를 모니터링하고 로그에 작성할 수 있는 권한을 Amazon RDS에 부여해야 합니다.

Amazon RDS 확장 모니터링을 위한 역할을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 역할을 선택한 다음 역할 생성을 선택합니다.
3. AWS 서비스 역할 유형을 선택한 후 다른 AWS 서비스의 사용 사례에서 RDS 서비스를 선택합니다. RDS - 확장 모니터링(RDS - Enhanced Monitoring)과 다음(Next)을 차례로 선택합니다.
4. AmazonRDSEnhancedMonitoringRole 권한 정책을 선택합니다.
5. Next(다음)를 선택합니다.
6. 역할 이름(Role name)에 이 역할의 목적을 나타내는 역할 이름을 입력합니다. 역할 이름은 AWS 계정 내에서 고유해야 합니다. 역할 이름이 정책에서 또는 ARN의 일부로 사용되는 경우 역할 이름은 대소문자를 구분합니다. 콘솔에서 고객에게 역할 이름이 표시되는 경우(예: 로그인 프로세스 중) 역할 이름은 대소문자를 구분하지 않습니다. 다양한 엔터티가 역할을 참조할 수 있기 때문에 역할이 생성된 후에는 역할 이름을 편집할 수 없습니다.
7. (선택 사항)설명에 새 역할에 대한 설명을 입력합니다.
8. (선택 사항) 태그를 키 값 페어로 연결하여 메타데이터를 사용자에게 추가합니다. IAM에서의 태그 사용에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요.
9. 역할을 검토한 다음 Create role을 선택합니다.

그러면 역할이 `monitoring.rds.amazonaws.com` 서비스에 해당 역할을 수임할 권한을 부여하는 신뢰 정책을 자동으로 얻습니다. 그러면 Amazon RDS는 `AmazonRDSEnhancedMonitoringRole` 정책에서 허용하는 모든 작업을 수행할 수 있습니다.

Enhanced Monitoring에 액세스하려는 사용자는 다음과 같이 사용자가 RDS 역할을 나열하도록 허용하는 문과 역할을 전달할 수 있도록 허용하는 문이 포함된 정책이 필요합니다. 계정 번호를 사용하여 역할 이름을 6단계에서 입력한 이름으로 바꿉니다.

```
    {
      "Sid": "PolicyStatementToAllowUserToListRoles",
      "Effect": "Allow",
      "Action": ["iam:ListRoles"],
      "Resource": "*"
```

```
    },
    {
        "Sid": "PolicyStatementToAllowUserToPassOneSpecificRole",
        "Effect": "Allow",
        "Action": [ "iam:PassRole" ],
        "Resource": "arn:aws:iam::account-id:role/RDS-Monitoring-Role"
    }
```

이 문을 다른 정책의 문과 결합하거나 고유 정책에 포함시킬 수 있습니다. 사용자가 RDS-로 시작하는 모든 역할을 전달할 수 있도록 지정하려면 다음과 같이 리소스 ARN의 역할 이름을 와일드카드로 바꿉니다.

```
        "Resource": "arn:aws:iam::account-id:role/RDS-*"
```

AWS CloudTrail 로그의 **iam:PassRole** 작업

PassRole은 API 호출이 아닙니다. PassRole은 권한입니다. 즉, IAM PassRole에 대해 CloudTrail 로그가 생성되지 않습니다. CloudTrail에서 어떤 AWS 서비스에 어떤 역할이 전달되는지 검토하려면 역할을 받는 AWS 리소스를 생성하거나 수정한 CloudTrail 로그를 검토해야 합니다. 예를 들어, 역할은 생성 시 AWS Lambda 함수에 전달됩니다. CreateFunction 작업에 대한 로그에는 함수에 전달된 역할의 레코드가 표시됩니다.

IAM 역할 임시 보안 자격 증명 취소

> **Warning**
>
> 이 페이지의 단계대로 수행하면, 역할을 수임하여 만들어진 현재 세션의 모든 사용자는 모든 AWS 작업 및 리소스에 액세스할 수 없게 됩니다. 이로 인해 사용자가 저장하지 않은 작업이 사라질 수 있습니다.

세션 지속 시간을 길게 하여(예: 12시간) 사용자가 AWS Management Console에 액세스할 수 있도록 하면 사용자의 임시 자격 증명이 금방 만료되지 않습니다. 사용자가 허가 받지 않은 제3자에게 실수로 보안 인증 정보를 노출한 경우, 해당 제3자는 세션의 지속 기간 동안 액세스 권한을 가지게 됩니다. 그러나 필요하다면 특정 시점 이전에 발행된 역할의 자격 증명에 대한 모든 권한을 즉시 취소할 수 있습니다. 그러면 지정한 시점 이전에 발행된 해당 역할의 임시 자격 증명은 모두 무효가 됩니다. 이에 따라 모든 사용자는 다시 인증을 받고 새 보안 인증 정보를 요청해야 합니다.

> **Note**
>
> 서비스 연결 역할에 대한 세션은 취소할 수 없습니다.

이 주제의 절차에 따라 역할의 권한을 취소하면 AWS는 모든 작업에 대한 모든 권한을 거부하는 새 인라인 정책을 만들어 해당 역할에 연결합니다. 여기에는 권한을 취소한 시점 이전에 역할을 위임한 사용자에게만 제한을 가하는 조건이 포함됩니다. 권한을 취소한 이후에 역할을 위임한 사용자에게는 거부 정책이 적용되지 않습니다.

액세스 거부에 대한 자세한 내용은 임시 보안 자격 증명에 대한 권한 비활성화 섹션을 참조하세요.

> **Important**
>
> 이 거부 정책은 콘솔 세션의 지속 기간이 긴 사용자만이 아니라 지정된 역할의 모든 사용자에게 적용됩니다.

역할의 세션 권한을 취소하기 위한 최소 권한

역할의 세션 권한을 취소하려면 해당 역할에 대한 `PutRolePolicy` 권한이 있어야 합니다. 이렇게 하면 해당 역할에 `AWSRevokeOlderSessions` 인라인 정책을 연결할 수 있게 됩니다.

세션 권한 취소

역할에서 세션 권한을 취소하여 역할을 수임한 사용자에게 모든 권한을 거부할 수 있습니다.

> **Note**
>
> IAM Identity Center 권한 세트에서 생성된 역할은 IAM에서 편집할 수 없습니다. IAM Identity Center에서 사용자의 활성 권한 세트 세션을 취소해야 합니다. 자세한 내용은 IAM Identity Center 사용 설명서의 권한 세트에 의해 생성된 활성 IAM 역할 세션 취소를 참조하세요.

역할 자격 증명의 현재 사용자에 대해 모든 권한을 즉시 거부하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할(Roles)을 선택한 다음, 권한을 취소할 역할의 이름(확인란 아님)을 선택합니다.

3. 선택한 역할의 요약 페이지에서 Revoke sessions(세션 취소) 탭을 선택합니다.
4. Revoke sessions(세션 취소) 탭에서 Revoke active sessions(활성 세션 취소)를 선택합니다.
5. AWS에 작업 확인 메시지가 나타납니다. 이 역할에 대한 모든 활성 세션을 취소함을 인정합니다.(I acknowledge that I am revoking all active sessions for this role.) 확인란을 선택하고 대화 상자에서 활성 세션 취소(Revoke active sessions)를 선택합니다.

   그런 다음 IAM은 역할에 `AWSRevokeOlderSessions`라는 정책을 연결합니다. Revoke active sessions(활성 세션 취소)를 선택하면 정책은 과거에 해당 역할이 부여된 사용자에 대한 모든 액세스를 거부하고 이후의 약 30초 동안에도 액세스를 거부합니다. 이 미래 시간 선택은 특정 리전에 업데이트된 정책이 시행되기 전에 획득되거나 갱신된 새 세션을 처리하기 위해 정책의 전파 지연을 고려합니다. Revoke active sessions(활성 세션 취소)를 선택한 후 약 30초 이후에 역할이 부여되는 사용자는 영향을 받지 않습니다. 변경 사항이 즉시 표시되지 않는 이유를 알아보려면 변경 사항이 매번 즉시 표시되는 것은 아닙니다 섹션을 참조하세요.

> **Note**
>
> 나중에 Revoke active sessions(활성 세션 취소)를 다시 선택하는 경우, 정책의 날짜/시간 스탬프가 새로 고쳐지면서 새로 지정된 시간 이전에 역할을 부여된 모든 사용자의 모든 권한을 거부하게 됩니다.

이러한 식으로 세션이 취소된 유효한 사용자는 작업을 계속하려면 새 세션을 위한 임시 자격 증명을 가져와야 합니다. AWS CLI은(는) 보안 인증이 만료될 때까지 이를 캐시합니다. CLI가 더 이상 유효하지 않은 캐시된 자격 증명을 강제로 삭제하고 새로 고치게 하려면 다음 명령 중 하나를 실행합니다.

Linux, macOS 또는 Unix

```
$ rm -r ~/.aws/cli/cache
```

Windows

```
C:\> del /s /q %UserProfile%\.aws\cli\cache
```

지정된 시간 전에 세션 권한 취소

또한 AWS CLI 또는 SDK를 사용하여 정책의 조건 요소에서 `aws:TokenIssueTime` 키 값을 지정하여 원하는 시간에 세션 권한을 취소할 수도 있습니다.

이 정책은 aws:TokenIssueTime의 값이 지정된 날짜와 시각보다 이른 경우 모든 권한을 거부합니다. aws:TokenIssueTime의 값은 임시 보안 자격 증명이 생성된 정확한 시간과 일치합니다. aws:TokenIssueTime 값은 임시 보안 인증으로 로그인된 AWS 요청의 컨텍스트에서만 존재하므로 정책의 Deny 문은 IAM 사용자의 장기 보안 인증으로 로그인한 요청에는 영향을 미치지 않습니다.

이 정책도 역할에 연결할 수 있습니다. 이 경우 정책은 지정된 시각 및 날짜 이전에 그 역할에 의해 생성된 임시 보안 자격 증명에만 영향을 미칩니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "DateLessThan": {"aws:TokenIssueTime": "2014-05-07T23:47:00Z"}
    }
  }
}
```

이러한 식으로 세션이 취소된 유효한 사용자는 작업을 계속하려면 새 세션을 위한 임시 자격 증명을 가져와야 합니다. AWS CLI은(는) 보안 인증이 만료될 때까지 이를 캐시합니다. CLI가 더 이상 유효하지 않은 캐시된 자격 증명을 강제로 삭제하고 새로 고치게 하려면 다음 명령 중 하나를 실행합니다.

Linux, macOS 또는 Unix

```
$ rm -r ~/.aws/cli/cache
```

Windows

```
C:\> del /s /q %UserProfile%\.aws\cli\cache
```

서비스 연결 역할 업데이트

서비스 연결 역할을 편집하는 데 사용하는 방법은 서비스에 따라 다릅니다. 일부 서비스는 사용자가 서비스 콘솔, API 또는 CLI에서 서비스 연결 역할의 권한을 편집할 수 있도록 허용합니다. 하지만 서비스 연결 역할을 만든 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. IAM 콘솔, API, CLI에서 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 사용을 지원하는 서비스에 대한 자세한 정보는 AWS IAM으로 작업하는 서비스를 참조하고 서비스 연결 역할 열에 예가 있는 서비스를 찾습니다. 서비스가 서비스 연결 역할 편집을 지

원하는지 여부를 알아보려면 예 링크를 선택하여 해당 서비스의 서비스 연결 역할 설명서 섹션을 참조하세요.

서비스 연결 역할 설명 편집(콘솔)

IAM 콘솔을 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 편집하려면(콘솔 사용)

1. IAM 콘솔의 탐색 창에서 역할을 선택합니다.
2. 변경할 역할 이름을 선택합니다.
3. 역할 설명의 맨 오른쪽에서 편집을 선택합니다.
4. 상자에 새 설명을 입력하고 저장을 선택합니다.

서비스 연결 역할 설명 편집(AWS CLI)

AWS CLI에서 IAM 명령을 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 변경하려면(AWS CLI)

1. (옵션) 역할의 현재 설명을 보려면 다음 명령 중 하나를 실행합니다.

   ```
   aws iam get-role --role-name ROLE-NAME
   ```

   CLI 명령에서 역할을 참조하려면 ARN이 아니라 역할 이름을 사용해야 합니다. 예를 들어, 어떤 역할의 ARN이 `arn:aws:iam::123456789012:role/myrole`인 경우 참조할 역할은 **myrole**입니다.
2. 서비스 연결 역할의 설명을 업데이트하려면 다음 명령을 실행합니다.

   ```
   aws iam update-role --role-name ROLE-NAME --description OPTIONAL-DESCRIPTION
   ```

서비스 연결 역할 설명 편집(AWS API)

AWS API를 사용하여 서비스 연결 역할의 설명을 편집할 수 있습니다.

서비스 연결 역할의 설명을 변경하려면(AWS API 사용)

1. (옵션) 역할의 현재 설명을 보려면 다음 작업을 호출하고 역할 이름을 지정합니다.

AWS API: GetRole

2. 역할의 설명을 업데이트하려면 다음 작업을 호출하고 역할 이름 및 설명(선택 사항)을 지정합니다.

   AWS API: UpdateRole

역할 트러스트 정책 업데이트

역할을 맡을 수 있는 주체를 바꾸려면 역할의 신뢰 정책을 변경해야 합니다. 서비스 연결 역할에 대한 신뢰 정책을 수정할 수 없습니다.

> 참고
>
> - 사용자가 역할 신뢰 정책에 보안 주체로 나열되지만 역할을 수임할 수 없는 경우 사용자의 권한 경계를 확인하세요. 사용자에 대한 권한 경계가 설정된 경우 권한 경계에서 `sts:AssumeRole` 작업이 허용되어야 합니다.
> - 사용자가 역할 세션 내에서 현재 역할을 다시 수임할 수 있도록 허용하려면 역할 ARN 또는 AWS 계정 ARN을 역할 신뢰 정책의 보안 주체로 지정합니다. Amazon EC2, Amazon ECS, Amazon EKS, Lambda와 같이 컴퓨팅 리소스를 제공하는 AWS 서비스는 임시 보안 인증을 제공하며 해당 보안 인증을 자동으로 업데이트합니다. 이렇게 하면 항상 유효한 자격 증명 집합을 가질 수 있습니다. 이러한 서비스의 경우 임시 자격 증명을 획득하기 위해 현재 역할을 다시 수임할 필요는 없습니다. 하지만 세션 태그 또는 세션 정책을 전달하려는 경우 현재 역할을 다시 수임해야 합니다.

역할 트러스트 정책 업데이트(콘솔)

AWS Management Console에서 역할 트러스트 정책 변경

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할(Roles)을 선택합니다.
3. 계정의 역할 목록에서 변경할 역할의 이름을 선택합니다.
4. 신뢰 관계(Trust relationships) 탭을 선택한 후 신뢰 정책 편집(Edit trust policy)을 선택합니다.

5. 필요에 따라 신뢰 정책을 편집합니다. 역할을 위임할 수 있는 보안 주체를 추가하려면 `Principal` 요소에 해당 보안 주체를 지정하세요. 예를 들어 다음 정책 조각은 `Principal` 요소에서 AWS 계정 2개를 참조하는 방법을 나타냅니다.

```
"Principal": {
  "AWS": [
    "arn:aws:iam::111122223333:root",
    "arn:aws:iam::444455556666:root"
  ]
},
```

다른 계정에서 보안 주체를 지정할 경우 역할의 신뢰 정책에 계정을 추가하는 것은 신뢰 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 기본적으로 신뢰 계정의 어떠한 사용자도 역할을 위임할 수 없습니다. 새로이 신뢰받는 계정에 대한 관리자는 사용자가 역할을 수임할 수 있는 권한을 허용해야 합니다. 이를 위해 관리자는 사용자와 연결된 정책을 생성 또는 편집하여 `sts:AssumeRole` 작업에 대한 사용자 액세스를 허용합니다. 자세한 정보는 다음 절차 또는 사용자에게 역할을 전환할 권한 부여 섹션을 참조하세요.

다음 정책 조각은 `Principal` 요소에서 두 가지 AWS 서비스를 참조하는 방법을 보여줍니다.

```
"Principal": {
  "Service": [
    "opsworks.amazonaws.com",
    "ec2.amazonaws.com"
  ]
},
```

6. 신뢰 정책 편집을 마쳤으면 정책 업데이트(Update policy)를 선택하여 변경 사항을 저장합니다.

정책 구조 및 구문에 대한 자세한 정보는 IAM의 정책 및 권한 단원과 IAM JSON 정책 요소 참조 섹션을 참조하세요.

신뢰할 수 있는 외부 계정의 사용자가 역할을 사용할 수 있도록 허용하려면(콘솔 사용)

자세한 정보와 이 절차에 대한 세부 정보는 사용자에게 역할을 전환할 권한 부여 섹션을 참조하세요.

1. 신뢰할 수 있는 외부 AWS 계정에 로그인합니다.
2. 사용자 또는 그룹 중 권한을 어디에 추가할지 결정합니다. 결정에 따라 IAM 콘솔의 탐색 창에서 사용자(Users) 또는 사용자 그룹(User groups)을 선택합니다.

3. 액세스 권한을 부여하려는 사용자나 그룹의 이름을 선택한 후 권한 탭을 선택합니다.
4. 다음 중 하나를 수행합니다.

   - 고객 관리형 정책을 편집하려면 정책 이름을 선택하고 정책 편집을 선택한 다음 JSON 탭을 선택합니다. AWS 관리형 정책은 편집할 수 없습니다. AWS 관리형 정책은 AWS 아이콘 () 으로 나타납니다. AWS 관리형 정책과 고객 관리형 정책의 차이점에 대한 자세한 정보는 관리형 정책과 인라인 정책 섹션을 참조하세요.
   - 인라인 정책을 편집하려면 정책 이름 옆에 있는 화살표를 선택하고 정책 편집을 선택합니다.
5. 정책 편집기에서 새로운 `Statement` 요소를 추가하여 다음과 같이 지정합니다.

   ```
   {
     "Effect": "Allow",
     "Action": "sts:AssumeRole",
     "Resource": "arn:aws:iam::ACCOUNT-ID:role/ROLE-NAME"
   }
   ```

   설명문의 ARN을 사용자가 수임할 수 있는 역할의 ARN으로 바꿉니다.
6. 화면의 메시지에 따라 정책 편집을 마칩니다.

역할 트러스트 정책 업데이트(AWS CLI)

AWS CLI를 사용하여 역할을 수임할 수 있는 주체를 변경할 수 있습니다.

역할 신뢰 정책을 수정하려면(AWS CLI)

1. (선택 사항) 수정할 역할의 이름을 모르는 경우 다음 명령을 실행하여 계정의 역할을 나열합니다.

   - aws iam list-roles
2. (옵션) 현재 역할의 신뢰 정책을 확인하려면 다음 명령을 실행합니다.

   - aws iam get-role
3. 역할에 액세스할 수 있는 신뢰할 수 있는 보안 주체를 변경하려면 업데이트된 신뢰 정책을 추가하여 텍스트 파일을 생성합니다. 정책 구조를 작성할 때는 어떤 텍스트 편집기든 사용할 수 있습니다.

   예를 들어 다음 신뢰 정책 조각은 `Principal` 요소에서 AWS 계정 2개를 참조하는 방법을 나타냅니다. 사용자가 개별 AWS 계정 2개를 사용하도록 허용하여 이 역할을 수임하도록 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": {"AWS": [
            "arn:aws:iam::111122223333:root",
            "arn:aws:iam::444455556666:root"
        ]},
        "Action": "sts:AssumeRole"
    }
}
```

다른 계정에서 보안 주체를 지정할 경우 역할의 신뢰 정책에 계정을 추가하는 것은 신뢰 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 기본적으로 신뢰 계정의 어떠한 사용자도 역할을 위임할 수 없습니다. 새로이 신뢰받는 계정에 대한 관리자는 사용자가 역할을 수임할 수 있는 권한을 허용해야 합니다. 이를 위해 관리자는 사용자와 연결된 정책을 생성 또는 편집하여 `sts:AssumeRole` 작업에 대한 사용자 액세스를 허용합니다. 자세한 정보는 다음 절차 또는 사용자에게 역할을 전환할 권한 부여 섹션을 참조하세요.

4. 방금 생성한 파일을 사용하여 신뢰 정책을 업데이트하려면 다음 명령을 실행합니다.

   - aws iam update-assume-role-policy

신뢰할 수 있는 외부 계정 사용자에게 역할 사용을 허용하려면(AWS CLI)

자세한 정보와 이 절차에 대한 세부 정보는 사용자에게 역할을 전환할 권한 부여 섹션을 참조하세요.

1. 역할에 대한 권한 정책을 포함하는 JSON 파일을 생성하여 역할을 수임할 수 있는 권한을 허용합니다. 예를 들어 다음 정책에는 필요한 최소 권한이 포함되어 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::ACCOUNT-ID-THAT-CONTAINS-ROLE:role/ROLE-NAME"
  }
}
```

설명문의 ARN을 사용자가 수임할 수 있는 역할의 ARN으로 바꿉니다.

2. 다음 명령을 실행하여 신뢰 정책을 포함하는 JSON 파일을 IAM에 업로드합니다.

   - aws iam create-policy

   이 명령의 출력 화면에는 정책의 ARN이 포함됩니다. 이후 단계에서 사용해야 하므로 이 ARN을 기록해 두세요.
3. 정책을 연결할 사용자 또는 그룹을 결정합니다. 원하는 사용자 또는 그룹의 이름을 모르는 경우에는 다음 명령 중 하나를 사용하여 계정에 속한 사용자 또는 그룹 목록을 조회합니다.

   - aws iam list-users
   - aws iam list-groups
4. 다음 명령 중 한 가지를 사용하여 이전 단계에서 생성한 정책을 사용자 또는 그룹에게 추가합니다.

   - aws iam attach-user-policy
   - aws iam attach-group-policy

역할 트러스트 정책 업데이트(AWS API)

AWS API를 사용하여 역할을 수임할 수 있는 주체를 변경할 수 있습니다.

역할 신뢰 정책을 수정하려면(AWS API)

1. (선택 사항) 변경할 역할의 이름을 모르는 경우 다음 연산을 호출하여 계정의 역할을 나열합니다.

   - ListRoles
2. (옵션) 현재 역할의 신뢰 정책을 확인하려면 다음 연산을 호출합니다.

   - GetRole
3. 역할에 액세스할 수 있는 신뢰할 수 있는 보안 주체를 변경하려면 업데이트된 신뢰 정책을 추가하여 텍스트 파일을 생성합니다. 정책 구조를 작성할 때는 어떤 텍스트 편집기든 사용할 수 있습니다.

   예를 들어 다음 신뢰 정책 조각은 `Principal` 요소에서 AWS 계정 2개를 참조하는 방법을 나타냅니다. 사용자가 개별 AWS 계정 2개를 사용하도록 허용하여 이 역할을 수임하도록 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
```

```
        "Effect": "Allow",
        "Principal": {"AWS": [
            "arn:aws:iam::111122223333:root",
            "arn:aws:iam::444455556666:root"
        ]},
        "Action": "sts:AssumeRole"
    }
}
```

다른 계정에서 보안 주체를 지정할 경우 역할의 신뢰 정책에 계정을 추가하는 것은 신뢰 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 기본적으로 신뢰 계정의 어떠한 사용자도 역할을 위임할 수 없습니다. 새로이 신뢰받는 계정에 대한 관리자는 사용자가 역할을 수임할 수 있는 권한을 허용해야 합니다. 이를 위해 관리자는 사용자와 연결된 정책을 생성 또는 편집하여 `sts:AssumeRole` 작업에 대한 사용자 액세스를 허용합니다. 자세한 정보는 다음 절차 또는 사용자에게 역할을 전환할 권한 부여 섹션을 참조하세요.

4. 방금 생성한 파일을 사용하여 신뢰 정책을 업데이트하려면 다음 작업을 호출합니다.

   - UpdateAssumeRolePolicy

신뢰할 수 있는 외부 계정 사용자에게 역할 사용을 허용하려면(AWS API)

자세한 정보와 이 절차에 대한 세부 정보는 사용자에게 역할을 전환할 권한 부여 섹션을 참조하세요.

1. 역할에 대한 권한 정책을 포함하는 JSON 파일을 생성하여 역할을 수임할 수 있는 권한을 허용합니다. 예를 들어 다음 정책에는 필요한 최소 권한이 포함되어 있습니다.

   ```
   {
     "Version": "2012-10-17",
     "Statement": {
       "Effect": "Allow",
       "Action": "sts:AssumeRole",
       "Resource": "arn:aws:iam::ACCOUNT-ID-THAT-CONTAINS-ROLE:role/ROLE-NAME"
     }
   }
   ```

   설명문의 ARN을 사용자가 수임할 수 있는 역할의 ARN으로 바꿉니다.

2. 다음 작업을 호출하여 신뢰 정책을 포함하는 JSON 파일을 IAM에 업로드합니다.

   - CreatePolicy

이 연산의 출력 화면에는 정책의 ARN이 포함됩니다. 이후 단계에서 사용해야 하므로 이 ARN을 기록해 두세요.

3. 정책을 연결할 사용자 또는 그룹을 결정합니다. 원하는 사용자 또는 그룹의 이름을 모르는 경우에는 다음 작업 중 하나를 호출하여 계정에 속한 사용자 또는 그룹 목록을 조회합니다.

   - ListUsers
   - ListGroups

4. 다음 연산 중 하나를 호출하여 이전 단계에서 생성한 정책을 사용자 또는 그룹에게 추가합니다.

   - API: AttachUserPolicy
   - AttachGroupPolicy

역할 권한 업데이트

다음 절차를 사용하여 역할의 권한 정책 및 권한 경계를 업데이트합니다.

사전 조건: 역할 액세스 보기

역할에 대한 권한을 변경하기 전에 최근 서비스 수준 활동을 검토해야 합니다. 이 기능은 사용 중인 보안 주체(사람 또는 애플리케이션)의 액세스 권한을 제거하지 않으려는 경우 중요합니다. 마지막으로 액세스한 정보 보기에 대한 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 섹션을 참조하세요.

역할의 권한 정책 업데이트

역할이 허용하는 권한을 변경하려면, 역할의 권한 정책을 수정합니다. IAM에서 서비스 연결 역할에 대한 권한 정책을 수정할 수 없습니다. 역할에 따른 서비스 내 권한 정책을 수정할 수 있는 가능성이 있습니다. 서비스에서 이 기능을 지원하는지 여부를 알아보려면 AWS IAM으로 작업하는 서비스를 참조하고 서비스 연결 역할(Service-linked roles) 열에 예(Yes)가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

역할 권한 정책 업데이트(콘솔)

역할이 허용하는 권한을 변경하려면(콘솔 사용)

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할을 선택합니다.
3. 수정하려는 역할의 이름을 선택한 후 권한 탭을 선택합니다.

4. 다음 중 하나를 수행합니다.

   - 기존의 고객 관리형 정책을 편집하려면 정책 이름을 선택한 후 정책 편집을 선택합니다.

     > **Note**
     >
     > AWS 관리형 정책은 편집할 수 없습니다. AWS 관리형 정책은 AWS 아이콘 () 으로 나타납니다. AWS 관리형 정책과 고객 관리형 정책의 차이점에 대한 자세한 정보는 관리형 정책과 인라인 정책 섹션을 참조하세요.

   - 기존 관리형 정책을 역할에 연결하려면 권한 추가(Add permissions)를 선택한 다음 정책 연결(Attach policies)을 선택합니다.
   - 기존 인라인 정책을 편집하려면 정책을 확장하고 편집(Edit)을 선택합니다.
   - 새 인라인 정책을 포함하려면 권한 추가(Add permissions)를 선택한 다음 인라인 정책 생성(Create inline policy)을 선택합니다.
   - 역할에서 기존 정책을 제거하려면 정책 이름 옆의 확인란을 선택한 다음 제거를 선택합니다.

역할 권한 정책 업데이트(AWS CLI)

역할이 허용하는 권한을 변경하려면, 역할의 권한 정책을 수정합니다. IAM에서 서비스 연결 역할에 대한 권한 정책을 수정할 수 없습니다. 역할에 따른 서비스 내 권한 정책을 수정할 수 있는 가능성이 있습니다. 서비스에서 이 기능을 지원하는지 여부를 알아보려면 AWS IAM으로 작업하는 서비스를 참조하고 서비스 연결 역할(Service-linked roles) 열에 예(Yes)가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

역할에서 허용되는 권한을 변경하려면(AWS CLI)

1. (옵션) 현재 역할과 연동되어 있는 권한을 확인하려면 다음 명령을 실행합니다.

   1. 인라인 정책을 나열하기 위한 aws iam list-role-policies
   2. 관리형 정책을 나열하기 위한 aws iam list-attached-role-policies

2. 역할 권한의 업데이트 명령은 관리형 정책을 업데이트할 때와 인라인 정책을 업데이트할 때 서로 다릅니다.

   관리형 정책을 업데이트하려면 다음 명령을 실행하여 새로운 버전의 관리형 정책을 생성합니다.

   - aws iam create-policy-version

인라인 정책을 업데이트하려면 다음 명령을 실행합니다.

- aws iam put-role-policy

역할 권한 정책 업데이트(AWS API)

역할이 허용하는 권한을 변경하려면, 역할의 권한 정책을 수정합니다. IAM에서 서비스 연결 역할에 대한 권한 정책을 수정할 수 없습니다. 역할에 따른 서비스 내 권한 정책을 수정할 수 있는 가능성이 있습니다. 서비스에서 이 기능을 지원하는지 여부를 알아보려면 AWS IAM으로 작업하는 서비스를 참조하고 서비스 연결 역할(Service-linked roles) 열에 예(Yes)가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

역할에서 허용되는 권한을 변경하려면(AWS API)

1. (옵션) 현재 역할과 연동되어 있는 권한을 확인하려면 다음 연산을 호출합니다.
   1. 인라인 정책을 나열하기 위한 ListRolePolicies
   2. 관리형 정책을 나열하기 위한 ListAttachedRolePolicies
2. 역할 권한의 업데이트 작업은 관리형 정책을 업데이트할 때와 인라인 정책을 업데이트할 때 서로 다릅니다.

   관리형 정책을 업데이트하려면 다음 연산을 호출하여 새로운 버전의 관리형 정책을 생성합니다.

   - CreatePolicyVersion

   인라인 정책을 업데이트하려면 다음 연산을 호출합니다.

   - PutRolePolicy

역할의 권한 경계 업데이트

역할이 허용하는 최대 권한을 변경하려면, 역할의 권한 경계를 수정합니다.

역할 권한 경계 업데이트(콘솔)

역할에 대한 권한 경계 설정에 사용된 정책을 변경하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 변경하려는 권한 경계가 있는 역할의 이름을 선택합니다.
4. 권한 탭을 선택합니다. 필요하다면 Permissions boundary(권한 경계) 섹션을 열고 Change boundary(경계 변경)을 선택합니다.
5. 정책을 선택하여 원하는 권한 경계를 사용하세요.
6. Change boundary(경계 변경)을 선택합니다.

   다음에 다른 사람이 이 역할을 수임할 때까지 변경 사항은 적용되지 않습니다.

역할 권한 경계 업데이트(AWS CLI)

역할(AWS CLI)에 대한 권한 경계 설정에 사용된 관리형 정책을 변경하려면

1. (선택 사항) 역할의 현재 권한 경계를 확인하려면 다음 명령을 실행합니다.
   - aws iam get-role
2. 다른 관리형 정책을 사용하여 역할에 대한 권한 경계를 업데이트하려면 다음 명령 중 하나를 실행합니다.
   - aws iam put-role-permissions-boundary

   역할은 권한 경계로서 하나의 관리형 정책만 가질 수 있습니다. 권한 경계를 변경하면 역할이 허용하는 최대 권한을 변경합니다.

역할 권한 경계 업데이트(AWS API)

역할(AWS API)에 대한 권한 경계 설정에 사용된 관리형 정책을 변경하려면

1. (선택 사항) 역할의 현재 권한 경계를 확인하려면 다음 작업을 호출합니다.
   - GetRole

2. 다른 관리형 정책을 사용하여 역할에 대한 권한 경계를 업데이트하려면 다음 작업 중 하나를 호출합니다.

   - PutRolePermissionsBoundary

   역할은 권한 경계로서 하나의 관리형 정책만 가질 수 있습니다. 권한 경계를 변경하면 역할이 허용하는 최대 권한을 변경합니다.

역할 설정 업데이트

다음 절차를 사용하여 역할 설명을 업데이트하거나 역할의 최대 세션 기간을 변경합니다.

역할 설명 업데이트

역할의 설명을 변경하려면 설명 텍스트를 수정합니다.

역할 설명 업데이트(콘솔)

역할의 설명을 변경하려면(콘솔 사용)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할을 선택합니다.
3. 변경할 역할 이름을 선택합니다.
4. 요약(Summary) 섹션에서 편집(Edit)을 선택합니다.
5. 상자에 새 설명을 입력하고 변경 사항 저장을 선택합니다.

역할 설명 업데이트(AWS CLI)

역할의 설명을 변경하려면(AWS CLI)

1. (옵션) 역할의 현재 설명을 보려면 다음 명령을 실행합니다.

   - aws iam get-role

2. 역할의 설명을 업데이트하려면 설명 파라미터와 함께 다음 명령을 실행합니다.

   - aws iam update-role

역할 설명 업데이트(AWS API)

역할의 설명을 변경하려면(AWS API)

1. (옵션) 현재 역할의 설명을 확인하려면 다음 연산을 호출합니다.
   - GetRole
2. 역할의 설명을 업데이트하려면 설명 파라미터와 함께 다음 연산을 호출합니다.
   - UpdateRole

역할의 최대 세션 기간 업데이트

콘솔, AWS CLI 또는 AWS API를 사용하여 수임한 역할에 대한 최대 세션 기간 설정을 지정하려면 최대 세션 지속 시간 설정의 값을 수정합니다. 이 설정에는 1~12시간의 값을 지정할 수 있습니다. 값을 지정하지 않으면 기본 최댓값인 1시간이 적용됩니다. 이 설정은 AWS 서비스에서 수임하는 세션을 제한하지 않습니다.

역할 최대 세션 기간 업데이트(콘솔)

콘솔, AWS CLI 또는 AWS API를 사용하여 수임한 역할에 대한 최대 세션 지속 시간 설정을 변경하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할을 선택합니다.
3. 변경할 역할 이름을 선택합니다.
4. 요약(Summary) 섹션에서 편집(Edit)을 선택합니다.
5. 최대 세션 지속 시간(Maximum session duration)에서 값을 선택합니다. 또는 사용자 지정 기간(Custom duration)을 선택하고 값(초)을 입력합니다.
6. Save changes(변경 사항 저장)를 선택합니다.

   다음에 다른 사람이 이 역할을 수임할 때까지 변경 사항은 적용되지 않습니다. 이 역할에 대한 기존 세션을 취소하는 방법을 알아보려면 IAM 역할 임시 보안 자격 증명 취소 섹션을 참조하세요.

AWS Management Console에서 IAM 사용자 세션은 기본적으로 12시간입니다. 콘솔에서 역할을 전환하는 IAM 사용자에게는 역할 최대 세션 기간 또는 사용자 세션의 남은 시간 중 더 적은 시간이 부여됩니다.

AWS CLI 또는 AWS API에서 역할을 수임한 사람은 이 최대값까지 더 긴 세션을 요청할 수 있습니다. `MaxSessionDuration` 설정은 요청할 수 있는 역할 세션의 최대 지속 시간을 결정합니다.

- AWS CLI을 사용하여 세션 지속 시간을 지정하려면 `duration-seconds` 파라미터를 사용합니다. 자세한 내용은 IAM 역할로 전환(AWS CLI) 섹션을 참조하세요.
- AWS API를 사용하여 세션 지속 시간을 지정하려면 `DurationSeconds` 파라미터를 사용합니다. 자세한 내용은 IAM 역할로 전환(AWS API) 섹션을 참조하세요.

역할 최대 세션 기간 업데이트(AWS CLI)

> **Note**
>
> AWS CLI 또는 API에서 역할을 수임한 사람은 누구나 `duration-seconds` CLI 파라미터 또는 `DurationSeconds` API 파라미터를 사용해 더 긴 세션을 요청할 수 있습니다. `MaxSessionDuration` 설정은 `DurationSeconds` 파라미터를 사용해 요청할 수 있는 역할 세션에 대한 최대 기간을 결정합니다. 사용자가 `DurationSeconds` 파라미터의 값을 지정하지 않으면 보안 자격 증명이 한 시간 동안 유효하게 됩니다.

AWS CLI를 사용하여 수임한 역할에 대한 최대 세션 기간 설정을 변경하려면(AWS CLI)

1. (옵션) 역할에 대한 현재 최대 세션 기간 설정을 확인하려면 다음 명령을 실행합니다.
   - aws iam get-role
2. 역할의 최대 세션 기간 설정을 업데이트하려면 `max-session-duration` CLI 파라미터 또는 `MaxSessionDuration` API 파라미터와 함께 다음 명령을 실행합니다.
   - aws iam update-role

   다음에 다른 사람이 이 역할을 수임할 때까지 변경 사항은 적용되지 않습니다. 이 역할에 대한 기존 세션을 취소하는 방법을 알아보려면 IAM 역할 임시 보안 자격 증명 취소 섹션을 참조하세요.

역할 최대 세션 기간 업데이트(AWS API)

> **Note**
>
> AWS CLI 또는 API에서 역할을 수임한 사람은 누구나 `duration-seconds` CLI 파라미터 또는 `DurationSeconds` API 파라미터를 사용해 더 긴 세션을 요청할 수 있습니다. `MaxSessionDuration` 설정은 `DurationSeconds` 파라미터를 사용해 요청할 수 있는 역할 세션에 대한 최대 기간을 결정합니다. 사용자가 `DurationSeconds` 파라미터의 값을 지정하지 않으면 보안 자격 증명이 한 시간 동안 유효하게 됩니다.

API를 사용하여 수임한 역할에 대한 최대 세션 기간 설정을 변경하려면(AWS API)

1. (옵션) 역할에 대한 현재 최대 세션 기간 설정을 확인하려면 다음 연산을 호출합니다.
   - GetRole
2. 역할의 최대 세션 기간 설정을 업데이트하려면 `max-sessionduration` CLI 파라미터 또는 `MaxSessionDuration` API 파라미터와 함께 다음 연산을 호출합니다.
   - UpdateRole

   다음에 다른 사람이 이 역할을 수임할 때까지 변경 사항은 적용되지 않습니다. 이 역할에 대한 기존 세션을 취소하는 방법을 알아보려면 IAM 역할 임시 보안 자격 증명 취소 섹션을 참조하세요.

역할 또는 인스턴스 프로파일 삭제

역할이 더 이상 필요하지 않은 경우 역할 및 연결된 권한을 삭제하는 것이 좋습니다. 그렇게 하면 적극적으로 모니터링하거나 유지 관리하지 않은 미사용 엔터티가 없습니다.

역할이 EC2 인스턴스와 연결된 경우 인스턴스 프로파일에서 해당 역할을 제거한 다음 인스턴스 프로파일을 삭제할 수도 있습니다.

> **Warning**
>
> 삭제하려는 역할 또는 인스턴스 프로파일로 실행 중인 Amazon EC2 인스턴스가 없는지 확인합니다. 실행 중인 인스턴스와 연결된 역할 또는 인스턴스 프로파일을 삭제하면 인스턴스에서 실행 중인 모든 애플리케이션이 중단됩니다.

역할을 영구적으로 삭제하지 않으려면 역할을 비활성화하면 됩니다. 이렇게 하려면 역할 정책을 변경한 다음 현재 세션을 모두 취소합니다. 예를 들어 모든 AWS에 대한 액세스를 거부한 정책을 역할에 추가할 수 있습니다. 역할을 수임하려는 모든 사용자에 대한 액세스를 거부하도록 신뢰 정책을 편집할 수도 있습니다. 세션 취소에 대한 자세한 내용은 IAM 역할 임시 보안 자격 증명 취소 섹션을 참조하세요.

주제

- 역할 액세스 보기
- 서비스 연결 역할 삭제
- IAM 역할 삭제(콘솔)
- IAM 역할 삭제(AWS CLI)
- IAM 역할 삭제(AWS API)
- 관련 정보

역할 액세스 보기

역할을 삭제하기 전에 역할이 마지막으로 사용된 시기를 검토하는 것이 좋습니다. AWS Management Console, AWS CLI 또는 AWS API를 사용하여 이 작업을 수행할 수 있습니다. 이 정보를 확인해야 하는 이유는 역할을 사용하는 사용자의 액세스 권한을 제거하지 않기 위해서입니다.

역할의 마지막 활동 날짜가 액세스 관리자(Access Advisor) 탭에 보고된 마지막 날짜와 일치하지 않을 수 있습니다. 액세스 관리자(Access Advisor) 탭은 역할의 권한 정책에서 허용하는 서비스에 대한 활동만 보고합니다. 역할의 마지막 활동 날짜에는 AWS의 모든 서비스에 액세스하려는 마지막 시도가 포함됩니다.

> **Note**
>
> 역할의 마지막 활동 및 액세스 관리자 데이터에 대한 추적 기간은 이후 400일입니다. 해당 리전이 이러한 기능 지원을 시작한 날짜가 작년이었다면 이 기간이 더 짧아질 수 있습니다. 이 역할이 400일 전에 사용되었을 수 있습니다. 추적 기간에 대한 자세한 내용은 AWS에서 마지막으로 액세스한 정보를 추적하는 위치 섹션을 참조하세요.

역할이 마지막으로 사용된 시기를 보려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

2. 탐색 창에서 역할을 선택합니다.
3. 활동을 보려는 역할의 행을 찾습니다. 검색 필드를 사용하여 결과를 좁힐 수 있습니다. 마지막 활동 열에서 역할이 마지막으로 사용된 이후 일 수를 확인합니다. 역할이 추적 기간 내에 사용되지 않은 경우 테이블에 없음이 표시됩니다.
4. 자세한 정보를 볼 역할의 이름을 선택합니다. 역할의 요약 페이지에는 해당 역할이 마지막으로 사용된 날짜를 표시하는 마지막 활동(Last activity)도 포함되어 있습니다. 역할이 지난 400일 이내에 사용되지 않은 경우 마지막 활동에 Not accessed in the tracking period(추적 기간 동안 액세스되지 않음)가 표시됩니다.

역할이 마지막으로 사용된 시기를 보려면(AWS CLI)

`aws iam get-role` - `RoleLastUsed` 객체를 포함하여 역할에 대한 정보를 반환하려면 이 명령을 실행합니다. 이 객체에는 역할이 마지막으로 사용된 `LastUsedDate` 및 `Region`이 있습니다. `RoleLastUsed`가 있지만 값을 포함하지 않는 경우 추적 기간 내에 해당 역할이 사용되지 않은 것입니다.

역할이 마지막으로 사용된 시기를 보려면(AWS API)

`GetRole` - `RoleLastUsed` 객체를 포함하여 역할에 대한 정보를 반환하려면 이 작업을 호출합니다. 이 객체에는 역할이 마지막으로 사용된 `LastUsedDate` 및 `Region`이 있습니다. `RoleLastUsed`가 있지만 값을 포함하지 않는 경우 추적 기간 내에 해당 역할이 사용되지 않은 것입니다.

서비스 연결 역할 삭제

서비스 연결 역할을 삭제하는 데 사용하는 방법은 서비스에 따라 다릅니다. 일부 경우에서는 서비스 연결 역할을 수동으로 삭제할 필요가 없습니다. 예를 들어, 서비스에서 특정 작업(예: 리소스 제거)을 완료하면 서비스에서 사용자의 서비스 연결 역할을 삭제할 수 있습니다. 서비스에서 서비스 연결 역할을 서비스 콘솔, API 또는 AWS CLI에서 수동으로 삭제하는 것이 지원되지 않는 경우도 있을 수 있습니다.

연결된 서비스의 서비스 연결 역할 관련 설명서를 참조하여 역할을 삭제하는 방법을 알아보세요. 콘솔의 IAM 역할 페이지에서 계정의 서비스 연결 역할을 볼 수 있습니다. 서비스 연결 역할은 테이블의 Trusted entities(신뢰할 수 있는 개체) 열에 (Service-linked role)((서비스 연결 역할))로 표시됩니다. 역할의 요약 페이지 배너에도 해당 역할이 서비스 연결 역할임이 표시됩니다.

서비스에 서비스 연결 역할 삭제에 대한 설명서가 없는 경우 IAM 콘솔, AWS CLI 또는 API를 사용하여 역할을 삭제할 수 있습니다.

IAM 역할 삭제(콘솔)

AWS Management Console를 사용하여 역할을 삭제하는 경우, IAM에서는 해당 역할과 관리형 정책을 자동으로 삭제합니다. 또한 역할과 연결된 모든 인라인 정책과 역할이 포함된 모든 Amazon EC2 인스턴스 프로파일도 자동으로 삭제됩니다.

> **Important**
>
> 경우에 따라 역할이 Amazon EC2 인스턴스 프로파일과 연결될 수 있으며, 역할과 인스턴스 프로파일의 이름이 같을 수 있습니다. 이 경우 AWS Management Console을 사용하여 역할 및 인스턴스 프로파일을 삭제할 수 있습니다. 이 연결은 콘솔에서 생성한 인스턴스 프로파일과 역할에서 자동으로 발생합니다. AWS CLI, Tools for Windows PowerShell 또는 AWS API에서 역할을 생성한 경우 역할과 인스턴스 프로파일의 이름이 서로 다를 수 있습니다. 이 경우 콘솔을 사용하여 역할과 인스턴스 프로파일을 삭제할 수 없습니다. 그 대신 AWS CLI, Tools for Windows PowerShell 또는 AWS API를 사용하여 먼저 인스턴스 프로파일에서 역할을 제거해야 합니다. 그런 다음 별도의 단계로 역할을 삭제해야 합니다.

역할을 삭제하려면(콘솔 사용)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택한 다음 삭제할 역할 이름 옆에 있는 확인란을 선택합니다.
3. 페이지 상단에서 삭제(Delete)를 선택합니다.
4. 확인 대화 상자가 나타나면 마지막으로 액세스한 정보를 검토합니다. 이 정보는 선택한 각 역할이 AWS 서비스를 마지막으로 액세스한 일시를 보여줍니다. 이를 통해 역할이 현재 활성 상태인지 확인할 수 있습니다. 계속하려면 텍스트 입력 필드에 역할 이름을 입력하고 삭제(Delete)를 선택합니다. 확신한다면, 마지막으로 액세스한 정보가 로드되고 있을 때에도 삭제를 진행할 수 있습니다.

> **Note**
>
> 인스턴스 프로파일이 역할과 이름이 동일한 경우를 제외하고는 콘솔을 사용하여 인스턴스 프로파일을 삭제할 수 없습니다. 이전 절차에서 설명한 바와 같이 역할 삭제 과정의 일부로 인스턴스 프로파일도 삭제됩니다. 역할까지 삭제하지 않고 인스턴스 프로파일을 삭제하려면 AWS CLI 또는 AWS API를 사용해야 합니다. 자세한 내용은 다음 섹션을 참조하세요.

IAM 역할 삭제(AWS CLI)

AWS CLI를 사용하여 역할을 삭제하는 경우, 먼저 해당 역할과 연결된 인라인 정책을 삭제해야 합니다. 또한 역할과 연결된 관리형 정책을 분리해야 합니다. 해당 역할이 들어 있는 연결된 인스턴스 프로파일은 별도로 삭제해야 합니다.

역할을 삭제하려면(AWS CLI)

1. 삭제할 역할의 이름을 모르는 경우 다음 명령을 입력하여 계정의 역할을 나열합니다.

   ```
   aws iam list-roles
   ```

   목록에는 각 역할의 Amazon 리소스 이름(ARN)이 포함되어 있습니다. CLI 명령에서 역할을 참조하려면 ARN이 아니라 역할 이름을 사용해야 합니다. 예를 들어, 어떤 역할의 ARN이 `arn:aws:iam::123456789012:role/myrole`인 경우 참조할 역할은 **myrole**입니다.
2. 역할과 관련 있는 모든 인스턴스 프로파일에서 역할을 제거합니다.

   a. 역할이 연결된 모든 인스턴스 프로파일을 나열하려면 다음 명령을 입력합니다.

      ```
      aws iam list-instance-profiles-for-role --role-name role-name
      ```

   b. 인스턴스 프로파일에서 역할을 제거하려면 각 인스턴스 프로파일에 대해 다음 명령을 입력합니다.

      ```
      aws iam remove-role-from-instance-profile --instance-profile-name instance-profile-name --role-name role-name
      ```
3. 역할과 연결된 모든 정책을 삭제합니다.

   a. 해당 역할에 있는 모든 인라인 정책을 나열하려면 다음 명령을 입력합니다.

      ```
      aws iam list-role-policies --role-name role-name
      ```

   b. 역할에서 각 인라인 정책을 삭제하려면 각 정책에 대해 다음 명령을 입력합니다.

      ```
      aws iam delete-role-policy --role-name role-name --policy-name policy-name
      ```

   c. 해당 역할에 연계돼 있는 모든 관리형 정책을 나열하려면 다음 명령을 입력합니다.

      ```
      aws iam list-attached-role-policies --role-name role-name
      ```

d. 역할에서 각 관리 정책을 분리하려면 각 정책에 대해 다음 명령을 입력합니다.

```
aws iam detach-role-policy --role-name role-name --policy-arn policy-arn
```

4. 다음 명령을 입력하여 역할을 삭제합니다.

```
aws iam delete-role --role-name role-name
```

5. 역할과 연결된 인스턴스 프로파일을 다시 사용할 계획이 없는 경우 다음 명령을 입력하여 삭제할 수 있습니다.

```
aws iam delete-instance-profile --instance-profile-name instance-profile-name
```

IAM 역할 삭제(AWS API)

IAM API를 사용하여 역할을 삭제하려면 먼저 해당 역할과 연결된 인라인 정책을 삭제해야 합니다. 또한 역할과 연결된 관리형 정책을 분리해야 합니다. 해당 역할이 들어 있는 연결된 인스턴스 프로파일은 별도로 삭제해야 합니다.

역할을 삭제하려면(AWS API)

1. 역할과 관련 있는 모든 인스턴스 프로파일을 나열하려면 ListInstanceProfilesForRole을 호출하세요.

   인스턴스 프로파일에서 해당 역할을 제거하려면 RemoveRoleFromInstanceProfile을 호출하세요. 역할 이름과 인스턴스 프로파일 이름을 전달해야 합니다.

   역할과 연결된 인스턴스 프로파일을 다시 사용하지 않을 경우 DeleteInstanceProfile을 호출하여 삭제할 수 있습니다.

2. 역할에 대한 모든 인라인 정책을 나열하려면 ListRolePolicies를 호출하세요.

   역할과 연결된 모든 인라인 정책을 삭제하려면 DeleteRolePolicy를 호출하세요. 역할 이름과 인라인 정책 이름을 전달해야 합니다.

3. 역할에 연결된 모든 관리형 정책을 나열하려면 ListAttachedRolePolicies를 직접적으로 호출합니다.

   역할에 연결된 관리형 정책을 분리하려면 DetachRolePolicy를 직접적으로 호출합니다. 역할 이름과 관리형 정책 ARN을 전달해야 합니다.

4. DeleteRole을 호출하여 역할을 삭제하세요.

관련 정보

인스턴스 프로파일에 대한 일반적인 정보는 인스턴스 프로파일 사용 섹션을 참조하세요.

서비스 연결 역할에 대한 일반적인 내용은 서비스 연결 역할 생성 섹션을 참조하세요.

역할 수임 방법

사용자, 애플리케이션 또는 서비스에서 이전에 생성한 역할을 사용하려면 그 역할로 전환할 수 있는 권한을 부여해야 합니다. 그룹 또는 사용자에게 추가된 어떤 정책도 필요한 권한을 부여하는 데 사용할 수 있습니다. 이 섹션에서는 사용자에게 역할을 사용할 권한을 부여하는 방법을 설명합니다. 또한 사용자가 AWS Management Console, Tools for Windows PowerShell, AWS Command Line Interface(AWS CLI) 및 `AssumeRole` API에서 역할로 전환하는 방법도 설명합니다.

> **Important**
>
> IAM 콘솔 대신 프로그래밍 방식으로 역할을 생성하는 경우에는 사용자의 선택에 따라 최대 64자인 `RoleName`뿐만 아니라 최대 512자인 `Path`도 추가할 수 있습니다. 그러나 AWS Management Console에서 역할 전환(Switch Role) 기능이 있는 역할을 사용하려면 `Path`와 `RoleName`을 합해 64자를 초과할 수 없습니다.

AWS Management Console에서 역할을 전환할 수 있습니다. AWS CLI 또는 API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 위임할 수 있습니다. 사용한 방법에 따라 역할을 수임할 수 있는 사용자와 역할 세션의 지속 가능 기간이 결정됩니다. `AssumeRole*` API 작업을 사용하는 경우 여러분이 맡는 IAM 역할은 리소스입니다. `AssumeRole*` API 작업을 호출하는 사용자 또는 역할이 보안 주체입니다.

다음 표에서는 역할 사용 방법을 비교합니다.

역할을 수임하는 방법	역할을 위임할 수 있는 사용자	자격 증명의 수명을 지정하는 방법	자격 증명의 수명(최소 | 최대 | 기본)
AWS Management Console	사용자(역할 전환을 통해)	최대 세션 기간에 역할 요약 페이지에서	15분 | 최대 세션 기간 설정[2] | 1시간
assume-role CLI 또는 AssumeRole API 작업	사용자 또는 역할[1]	duration-seconds CLI 또는 DurationSeconds API 파라미터	15분 | 최대 세션 기간 설정[2] | 1시간
assume-role-with-saml CLI 또는 AssumeRoleWithSAML API 작업	SAML을 사용하여 인증된 모든 사용자	duration-seconds CLI 또는 DurationSeconds API 파라미터	15분 | 최대 세션 기간 설정[2] | 1시간
assume-role-with-web-identity CLI 또는 AssumeRoleWithWebIdentity API 작업	OIDC 공급자를 사용하여 인증된 모든 사용자	duration-seconds CLI 또는 DurationSeconds API 파라미터	15분 | 최대 세션 기간 설정[2] | 1시간
AssumeRole로 생성된 콘솔 URL	사용자 또는 역할	URL의 SessionDuration HTML 파라미터	15분 | 12시간 | 1시간
AssumeRoleWithSAML로 생성된 콘솔 URL	SAML을 사용하여 인증된 모든 사용자	URL의 SessionDu	15분 | 12시간 | 1시간

역할을 수임하는 방법	역할을 위임할 수 있는 사용자	자격 증명의 수명을 지정하는 방법	자격 증명의 수명 (최소 | 최대 | 기본)
		ration HTML 파라미터	
AssumeRoleWithWebIdentity 로 생성된 콘솔 URL	OIDC 공급자를 사용하여 인증된 모든 사용자	URL의 SessionDuration HTML 파라미터	15분 | 12시간 | 1시간

[1] 하나의 역할이 자격 증명을 사용하여 다른 역할을 수임하는 것을 역할 체인이라고 합니다. 역할 함께 묶기를 사용하는 경우 새 자격 증명의 유효 기간은 최대 1시간으로 제한됩니다. 역할을 사용하여 EC2 인스턴스에서 실행되는 애플리케이션에 권한을 부여하는 경우, 이러한 애플리케이션에는 이 제한이 적용되지 않습니다.

[2] 이 설정에는 1~12시간의 값을 지정할 수 있습니다. 최대 세션 기간 설정을 수정하는 방법에 대한 자세한 내용은 IAM 역할 관리 섹션을 참조하세요. 이 설정은 역할 자격 증명을 얻을 때 요청할 수 있는 최대 세션 기간을 결정합니다. 예를 들어 AssumeRole* API 작업을 사용하여 역할을 위임할 때 DurationSeconds 파라미터를 사용하여 세션 길이를 지정할 수 있습니다. 이 파라미터를 사용하여 역할 세션 기간을 900초(15분)에서 해당 역할에 대한 최대 세션 기간 설정까지 지정합니다. 콘솔에서 역할을 전환하는 IAM 사용자에게는 최대 세션 기간 또는 사용자 세션의 남은 시간 중 더 적은 시간이 부여됩니다. 역할에서 최대 기간을 5시간으로 설정한다고 가정합니다. 콘솔에 10시간(기본 최대 12시간 중) 동안 로그인한 IAM 사용자가 해당 역할로 전환합니다. 이 경우 사용 가능한 역할 세션 기간은 2시간입니다. 역할에 대한 최댓값을 확인하는 방법을 알아보려면 이 페이지 뒷부분에 나오는 역할의 최대 세션 기간 업데이트 섹션을 참조하세요.

> **참고**
>
> - 최대 세션 기간 설정은 AWS 서비스에서 수임하는 세션을 제한하지 않습니다.
> - Amazon EC2 IAM 역할 보안 인증에는 역할에 구성된 최대 세션 기간이 적용되지 않습니다.
> - 사용자가 역할 세션 내에서 현재 역할을 다시 수임할 수 있도록 허용하려면 역할 ARN 또는 AWS 계정 ARN을 역할 신뢰 정책의 보안 주체로 지정합니다. Amazon EC2, Amazon ECS, Amazon EKS, Lambda와 같이 컴퓨팅 리소스를 제공하는 AWS 서비스는 임시 보안 인증을 제공하며 해당 보안 인증을 자동으로 업데이트합니다. 이렇게 하면 항상 유효한 자격 증명

집합을 가질 수 있습니다. 이러한 서비스의 경우 임시 자격 증명을 획득하기 위해 현재 역할을 다시 수임할 필요는 없습니다. 하지만 세션 태그 또는 세션 정책을 전달하려는 경우 현재 역할을 다시 수임해야 합니다. 보안 주체 역할 ARN 또는 AWS 계정 ARN을 추가하기 위해 역할 신뢰 정책을 수정하는 방법을 알아보려면 역할 트러스트 정책 업데이트 을 참조하세요.

주제

- 역할로 전환(콘솔)
- IAM 역할로 전환(AWS CLI)
- IAM 역할로 전환(Tools for Windows PowerShell)
- IAM 역할로 전환(AWS API)
- IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여
- 인스턴스 프로파일 사용

역할로 전환(콘솔)

역할은 필요한 AWS 리소스에 액세스하는 데 사용할 수 있는 일련의 권한을 지정합니다. 이러한 면에서 AWS Identity and Access Management(IAM)의 사용자와 비슷합니다. 사용자로 로그인할 때는 특정 권한이 부여됩니다. 하지만 역할로 로그인하지는 못하기 때문에 일단 로그인한 후에 역할로 전환할 수 있습니다. 이 경우 초기의 사용자 권한은 잠시 무효화되고 역할에게 할당된 권한이 부여됩니다. 역할은 자신의 계정 또는 그 밖의 다른 AWS 계정에 속한 것일 수 있습니다. 역할, 역할의 이점 및 생성 방법에 대한 자세한 내용은 다음(IAM 역할 및 IAM 역할 생성)을 참조하세요.

> **Important**
>
> 사용자의 권한과 전환 대상인 역할의 권한은 누적되지 않습니다. 한 번에 오직 하나의 권한 집합만이 활성화됩니다. 어떤 역할로 전환할 때 사용자 권한은 일시적으로 포기하고 역할에 할당된 권한을 가지고 작업합니다. 역할을 끝내면 사용자 권한이 자동으로 회복됩니다.

AWS Management Console에서 역할을 전환하는 경우, 콘솔은 항상 원래 자격 증명을 사용하여 전환을 승인합니다. 이는 IAM 사용자, IAM Identity Center의 사용자, SAML 페더레이션 역할 또는 웹 아이덴티티 페더레이션 역할 중 어느 것으로 로그인하는지와 관계없이 적용됩니다. 예를 들어, RoleA로 전환하는 경우 IAM에서는 원래 사용자 또는 페더레이션 역할 자격 증명을 사용하여 RoleA를 수임할 수 있는지 여부를 결정합니다. RoleA를 사용하는 중에 RoleB로 전환하는 경우에도 AWS에서는 RoleA의 자격 증명이 아닌, 원래 사용자 자격 증명 또는 연동된 역할 자격 증명을 사용하여 전환을 승인합니다.

콘솔에서 역할 전환에 관해 알아야 할 사항

이 섹션에서는 IAM 콘솔을 사용하여 역할을 전환하는 방법에 대한 추가 정보를 제공합니다.

> 참고:
>
> - AWS 계정 루트 사용자로 로그인할 경우 역할을 바꿀 수 없습니다. IAM 사용자, IAM Identity Center의 사용자, SAML 페더레이션 역할 또는 웹 아이덴티티 페더레이션 역할로 로그인할 때 역할을 전환할 수 있습니다.
> - AWS Management Console에서의 역할을 ExternalId 값이 필요한 역할로 전환할 수 없습니다. `ExternalId` 파라미터를 지원하는 `AssumeRole` API를 호출해야만 이러한 역할로 전환할 수 있습니다.

- 관리자가 링크를 제공하는 경우 링크를 선택하여 다음 절차의 Step 5 단계로 넘어갑니다. 링크를 클릭하면 적절한 웹 페이지로 이동하고 계정 ID(또는 별칭)와 역할 이름이 채워집니다.
- 링크를 수동으로 구성한 후 다음 절차의 Step 5 단계로 건너뛸 수 있습니다. 링크를 구성하려면 다음 형식을 사용합니다.

  `https://signin.aws.amazon.com/switchrole?account=`*`account_id_number`*`&roleName=`*`role_name`*`&displayName=`*`text_to_display`*

  여기서 다음 텍스트를 바꿉니다.

  - *`account_id_number`* - 관리자가 제공한 12자리 계정 식별자. 또는 URL에 계정 ID 대신 계정 이름이 포함되도록 관리자가 계정 별칭을 생성할 수 있습니다. 자세한 내용을 알아보려면 AWS 로그인 사용 설명서의 사용자 유형을 참조하세요.
  - *`role_name`* - 수임하려는 역할의 이름. 이 이름은 역할의 ARN 끝에서 가져올 수 있습니다. 예를 들어 역할 ARN: `TestRole`에서 수임하려는 역할의 이름은 `arn:aws:iam::123456789012:role/TestRole`입니다.
  - (선택 사항) *`text_to_display`* - 이 역할이 활성화되었을 때 탐색 표시줄에 사용자 이름 대신 표시되도록 하고 싶은 텍스트
- 관리자가 제공하는 정보를 사용하여 아래 절차를 통해 역할을 수동으로 전환할 수 있습니다.

기본적으로 역할을 전환할 때 AWS Management Console 세션은 1시간 동안 지속됩니다. IAM 사용자 세션은 기본적으로 12시간입니다. 콘솔에서 역할을 전환하는 IAM 사용자에게는 역할 최대 세션 기간 또는 사용자 세션의 남은 시간 중 더 적은 시간이 부여됩니다. 예를 들어 역할의 최대 세션 기간이 10시

간으로 설정되어 있다고 가정합니다. IAM 사용자가 역할로 전환하기로 결정할 때까지 8시간 동안 콘솔에 로그인해 있었습니다. 사용자 세션에는 4시간이 남아 있으므로 허용되는 역할 세션 기간은 4시간입니다. 다음 표에서는 콘솔에서 역할을 전환할 때 IAM 사용자의 세션 기간을 결정하는 방법을 설명합니다.

IAM 사용자 세션 남은 시간…	역할 세션 기간…
역할 최대 세션 기간보다 작음	사용자 세션의 남은 시간
역할 최대 세션 기간보다 큼	최대 세션 기간 값
역할 최대 세션 기간과 같음	최대 세션 기간 값(근사치)

> **Note**
>
> 일부 AWS 서비스 콘솔에서는 역할 세션이 만료되는 경우 아무 조치를 취하지 않아도 역할 세션을 자동으로 갱신할 수 있습니다. 또한 브라우저를 다시 로드하여 세션을 다시 인증하라는 메시지를 표시하는 경우도 있습니다.

역할을 수임할 때 발생할 수 있는 일반적인 문제를 해결하려면 역할을 수임할 수 없음 섹션을 참조하세요.

역할을 전환하려면(콘솔)

1. AWS Management Console에 IAM 사용자로 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. IAM 콘솔에서 상단 오른쪽 모서리에 있는 탐색 표시줄에서 사용자 이름을 선택합니다. 일반적인 형식은 ***username*@*account_ID_number_or_alias***입니다.
3. 역할 전환을 선택합니다. 이 옵션을 처음 선택하면 자세한 정보를 제공하는 페이지가 나타납니다. 그 정보를 읽은 후에 역할 전환(Switch Role)을 클릭합니다. 브라우저 쿠키를 청소하면 이 페이지가 다시 나타나게 할 수 있습니다.
4. 역할 전환 페이지에서 계정 ID 번호 또는 계정 별칭 및 관리자가 제공한 역할 이름을 입력합니다.

> **Note**
>
> 관리자가 경로를 포함하여 역할을 생성한 경우(예: `division_abc/subdivision_efg/roleToDoX`)에는 역할 상자에 전체 경로와 이름을 입력해야 합니다. 역할 이름만 입력하는 경우 또는 결합된 `Path` 및 `RoleName`이 64자를 초과하는 경우 역할 전환에 실패합니다. 이것은 역할 이름을 저장하는 브라우저 쿠키의 한계입니다. 이러한 경우 관리자에게 문의해 경로 및 역할 이름의 크기를 줄여 달라고 요청하세요.

5. (선택 사항) 표시 이름(Display name)을 선택합니다. 이 역할이 활성화되었을 때 탐색 표시줄에 사용자 이름 대신 표시되도록 하려는 텍스트를 입력합니다. 이름은 계정 및 역할 정보에 따라 다르게 제시되지만 특별한 의미를 갖도록 직접 변경하는 것도 가능합니다. 또한, 표시되는 이름이 돋보이도록 색상을 선택할 수도 있습니다. 이름과 색상은 이 역할이 활성화되어 권한이 변경되는 시점을 다시 한 번 알려줍니다. 예를 들어 테스트 환경에 대한 액세스 권한을 부여하는 역할에 대해 표시 이름(Display name)을 **`Test`**로 지정하고 색상(Color)은 녹색으로 선택합니다. 프로덕션에 대한 액세스 권한을 부여하는 역할에 대해서는 표시 이름(Display name)을 **`Production`**으로 지정하고 색상(Color)은 빨간색으로 선택합니다.
6. 역할 전환을 선택합니다. 표시 이름과 색상이 탐색 표시줄에 사용자 이름 대신 나타나고, 역할에서 부여하는 권한을 사용하여 시작할 수 있습니다.

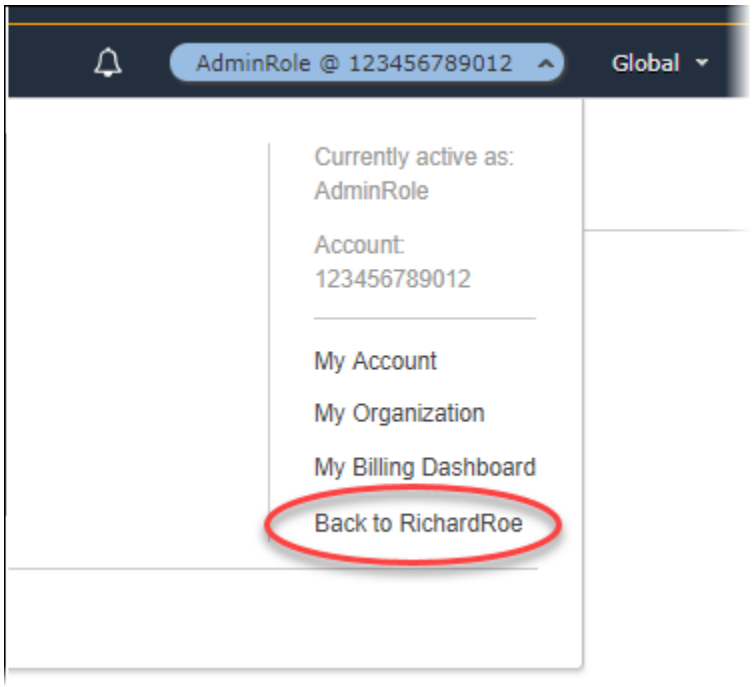
> **도움말**
>
> 마지막으로 사용한 몇 가지 역할은 []] 메뉴에 표시됩니다. 다음에 이 중 하나로 역할을 전환해야 할 때는 원하는 역할을 선택하기만 하면 됩니다. 역할이 메뉴에 표시되지 않으면 계정 및 역할 정보를 수동으로 입력하면 됩니다.

역할 사용을 중지하려면(콘솔)

1. IAM 콘솔의 탐색 모음 오른쪽 위쪽에서 역할의 표시 이름(Display name)을 선택합니다. 일반적인 형식은 ***`rolename`***@***`account_ID_number_or_alias`***입니다.
2. ***`username`***으로 돌아가기(Back to username)를 선택합니다. 역할과 그 권한이 비활성화되면서 IAM 사용자 및 그룹에 연결된 권한이 자동으로 복구됩니다.

   예를 들어, 사용자 이름 `123456789012`을 사용하여 계정 번호 `RichardRoe`로 로그인했다고 가정하십시다. `AdminRole` 역할을 사용한 후, 사용자가 역할 사용을 중지하고 원래 사용자 권

한으로 돌아가고자 합니다. 역할 사용을 중지하려면 AdminRole @ 123456789012를 선택한 후 RichardRoe로 돌아가기(Back to RichardRoe)를 선택합니다.



IAM 역할로 전환(AWS CLI)

역할은 필요한 AWS 리소스에 액세스하는 데 사용할 수 있는 일련의 권한을 지정합니다. 이러한 면에서 AWS Identity and Access Management(IAM)의 사용자와 비슷합니다. 사용자로 로그인할 때는 특정 권한이 부여됩니다. 하지만 역할로 로그인하지는 못하기 때문에 일단 사용자로 로그인한 후에 역할로 전환할 수 있습니다. 이 경우 초기의 사용자 권한은 잠시 무효화되고 역할에게 할당된 권한이 부여됩니다. 역할은 자신의 계정 또는 그 밖의 다른 AWS 계정에 속한 것일 수 있습니다. 역할과 역할의 이점, 역할 생성 및 구성 방법에 대한 자세한 내용은 IAM 역할 및 IAM 역할 생성 섹션을 참조하세요. 역할을 수임하는 데 사용할 수 있는 여러 방법을 알아보려면 역할 수임 방법 섹션을 참조하세요.

> **Important**
>
> IAM 사용자의 권한과 수임한 역할의 권한은 누적되지 않습니다. 한 번에 오직 하나의 권한 집합만이 활성화됩니다. 어떤 역할을 수임할 때 이전 사용자 또는 역할 권한은 일시적으로 포기하고 해당 역할에 할당된 권한을 가지고 작업합니다. 역할을 끝내면 사용자 권한이 자동으로 회복됩니다.

IAM 사용자로 로그인한 경우 역할을 사용하여 AWS CLI 명령을 실행할 수 있습니다. 역할을 이미 사용 중인 외부에서 인증된 사용자(SAML 또는 OIDC)로 로그인한 경우에도 역할을 사용하여 AWS CLI 명령을 실행할 수 있습니다. 또한 인스턴스 프로파일 전체에서 명령에 연결된 Amazon EC2 인스턴스 내에서 역할을 사용하여 AWS CLI 명령을 실행할 수 있습니다. AWS 계정 루트 사용자로 로그인되어 있을 때는 역할을 수임할 수 없습니다.

역할 체인 - 역할의 권한을 사용하여 두 번째 역할에 액세스하는 역할 체인을 사용할 수도 있습니다.

기본적으로 역할 세션은 한 시간 동안 지속됩니다. `assume-role*` CLI 작업을 사용하여 역할을 수임하는 경우 `duration-seconds` 파라미터에 대한 값을 지정할 수 있습니다. 이 값의 범위는 900초(15분)에서 해당 역할에 대한 최대 세션 기간 설정까지일 수 있습니다. 콘솔에서 역할을 전환하면 세션 시간이 최대 1시간으로 제한됩니다. 역할에 대한 최댓값을 확인하는 방법을 알아보려면 역할의 최대 세션 기간 업데이트 섹션을 참조하세요.

역할 함께 묶기를 사용하는 경우 세션 기간은 최대 1시간으로 제한됩니다. 그런 다음 `duration-seconds` 파라미터를 사용하여 1시간보다 큰 값을 입력하면 이 작업에 실패합니다.

예제 시나리오: 프로덕션 역할로 전환

개발 환경에서 작업하는 IAM 사용자라고 가정합니다. AWS CLI의 명령줄로 프로덕션 환경에서 작업해야 하는 경우가 종종 있습니다. 사용할 수 있는 액세스 키 자격 증명 세트가 이미 하나 있습니다. 이 세트는 표준 IAM 사용자에게 할당된 액세스 키 페어일 수도 있고, 페더레이션 사용자로 로그인한 경우에는 초기에 할당된 역할에 대한 액세스 키 페어일 수도 있습니다. 현재 권한에 의해 특정 IAM 역할을 수임할 수 있는 능력이 부여되는 경우, AWS CLI 구성 파일의 'profile'에서 해당 역할을 식별할 수 있습니다. 그러면 해당 명령은 원래 자격 증명이 아닌 지정된 IAM 역할의 권한으로 실행됩니다. AWS CLI 명령에서 해당 프로파일을 지정하는 경우에는 새 역할을 사용하게 됩니다. 이 경우 개발 계정의 원래 권한을 동시에 사용할 수 없습니다. 한 번에 한 가지 권한 세트만 적용될 수 있기 때문입니다.

> **Note**
>
> 보안을 위해 관리자는 AWS CloudTrail 로그를 검토하여 AWS에서 누가 작업을 수행했는지 확인할 수 있습니다. 관리자는 사용자가 역할을 수임할 때 소스 자격 증명이나 역할 세션 이름을 지정하도록 요구할 수 있습니다. 자세한 내용은 `sts:SourceIdentity` 및 `sts:RoleSessionName`을(를) 참조하세요.

프로덕션 역할로 전환(AWS CLI)

1. AWS CLI를 사용한 적이 없을 경우 먼저 기본 CLI 프로필을 구성해야 합니다. 명령 프롬프트를 열고 IAM 사용자 또는 페더레이션 역할의 액세스 키를 사용하도록 AWS CLI 설치를 설정합니다. 자

세한 내용은 AWS Command Line Interface 사용 설명서의 AWS Command Line Interface 구성을 참조하세요.

다음과 같이 aws configure 명령을 실행합니다.

```
aws configure
```

요청 메시지가 나타나면 다음 정보를 입력합니다.

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-east-2
Default output format [None]: json
```

2. Unix 또는 Linux의 `.aws/config` 파일, 또는 Windows의 `C:\Users\USERNAME\.aws\config` 파일에서 역할에 대한 새 프로필을 만듭니다. 다음 예에서는 123456789012 계정의 *ProductionAccessRole* 역할로 전환하는 `prodaccess`라는 프로필을 만듭니다. 해당 역할을 만든 계정 관리자에게서 역할 ARN을 받습니다. 이 프로필이 호출되면 AWS CLI에서는 `source_profile`의 자격 증명을 사용하여 해당 역할의 자격 증명을 요청합니다. 이로 인해 `role_arn`로 참조되는 자격 증명에는 `source_profile`에 지정된 역할에 대한 `sts:AssumeRole` 권한이 있어야 합니다.

```
[profile prodaccess]
    role_arn = arn:aws:iam::123456789012:role/ProductionAccessRole
    source_profile = default
```

3. 새 프로필을 만든 후 `--profile prodaccess` 파라미터를 지정하는 AWS CLI 명령은 기본 사용자 대신 IAM 역할 `ProductionAccessRole`에 연결된 권한에 따라 실행됩니다.

```
aws iam list-users --profile prodaccess
```

이 명령은 `ProductionAccessRole`에 할당된 권한이 현재 AWS 계정에 사용자를 나열하는 것을 가능하게 하는 경우에 작동합니다.

4. 원래 자격 증명에 의해 부여된 권한으로 돌아가려면 명령을 `--profile` 파라미터 없이 실행합니다. AWS CLI에서 다시 기본 프로필의 자격 증명(Step 1에서 구성)이 사용됩니다.

자세한 내용은 AWS Command Line Interface 사용 설명서의 역할 수임을 참조하세요.

예제 시나리오: 인스턴스 프로파일 역할이 다른 계정의 역할로 전환하도록 허용

두 개의 AWS 계정을 사용 중이고 Amazon EC2 인스턴스에서 실행 중인 특정 애플리케이션이 두 계정 모두에서 AWS CLI 명령을 실행하도록 허용하고자 한다고 가정합니다. EC2 인스턴스가 111111111111 계정에 존재한다고 가정합니다. 해당 인스턴스에는 애플리케이션이 동일한 111111111111 계정 내에 있는 my-bucket-1 버킷에서 읽기 전용 Amazon S3 작업을 수행하도록 허용하는 abcd 인스턴스 프로파일 역할이 포함되어 있습니다. 하지만 애플리케이션은 efgh 크로스 계정 역할을 수임하여 222222222222 계정에서 작업을 수행하도록 허용되어야 합니다. 이를 위해 abcd EC2 인스턴스 프로파일 역할에 다음과 같은 권한 정책이 있어야 합니다.

계정 111111111111 ***abcd*** 역할 권한 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccountLevelS3Actions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowListAndReadS3ActionOnMyBucket",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket-1/*",
                "arn:aws:s3:::my-bucket-1"
            ]
        },
        {
            "Sid": "AllowIPToAssumeCrossAccountRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::222222222222:role/efgh"
```

```
        }
    ]
}
```

efgh 크로스 계정 역할이 동일한 222222222222 계정 내에 있는 my-bucket-2 버킷에서 읽기 전용 Amazon S3 작업을 수행할 수 있다고 가정합니다. 이를 위해 efgh 크로스 계정 역할에 다음과 같은 권한 정책이 있어야 합니다.

계정 222222222222 ***efgh*** 역할 권한 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccountLevelS3Actions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowListAndReadS3ActionOnMyBucket",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket-2/*",
                "arn:aws:s3:::my-bucket-2"
            ]
        }
    ]
}
```

efgh 역할은 abcd 인스턴스 프로파일 역할이 이를 수임하도록 허용해야 합니다. 이를 위해 efgh 역할에 다음과 같은 신뢰 정책이 있어야 합니다.

계정 222222222222 ***efgh*** 역할 신뢰 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "efghTrustPolicy",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "arn:aws:iam::111111111111:role/abcd"}
        }
    ]
}
```

계정 222222222222에서 AWS CLI 명령을 실행하려면 CLI 구성 파일을 업데이트해야 합니다. AWS CLI 구성 파일에서 efgh 역할을 "프로파일"로 식별하고 abcd EC2 인스턴스 프로파일 역할을 "자격 증명 소스"로 식별합니다. 그런 다음 CLI 명령은 기존의 abcd 역할이 아닌 efgh 역할의 권한을 사용하여 실행됩니다.

> **Note**
>
> 계정에서 보안을 목적으로 AWS CloudTrail을 사용해 계정의 역할 사용을 감사할 수 있습니다. CloudTrail 로그에서 여러 보안 주체가 한 역할을 사용할 때 역할 세션 간을 구분하려면 역할 세션 이름을 사용할 수 있습니다. 이 항목에서 설명하는 것처럼 AWS CLI에서 사용자를 대신해 역할을 수임하면 역할 세션 이름이 AWS-CLI-session-*nnnnnnnn*으로 자동으로 생성됩니다. 여기서 *nnnnnnnn*은 Unix epoch time(1970년 1월 1일 자정 UTC 이후 경과된 초 수)으로 시간을 나타낸 정수입니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 이벤트 참조를 참조하세요.

EC2 인스턴스 프로파일 역할의 크로스 계정 역할 전환 허용(AWS CLI)

1. 기본 CLI 프로파일을 구성할 필요가 없습니다. 대신 EC2 인스턴스 프로파일 메타데이터에서 자격 증명을 불러올 수 있습니다. .aws/config 파일에서 역할에 대한 새 프로파일을 만듭니다. 다음 예에서는 222222222222 계정의 *efgh* 역할로 전환하는 instancecrossaccount라는 프로필을 만듭니다. 이 프로필이 호출되면 AWS CLI에서는 EC2 인스턴스 프로파일 메타데이터의 자격 증명을 사용하여 해당 역할의 자격 증명을 요청합니다. 이로 인해 EC2 인스턴스 프로파일 역할에는 role_arn에 지정된 역할에 대한 sts:AssumeRole 권한이 있어야 합니다.

   ```
   [profile instancecrossaccount]
   role_arn = arn:aws:iam::222222222222:role/efgh
   ```

```
credential_source = Ec2InstanceMetadata
```

2. 새 프로필을 만든 후 `--profile instancecrossaccount` 파라미터를 지정하는 AWS CLI 명령은 222222222222 계정의 `efgh` 역할에 연결된 권한에 따라 실행됩니다.

   ```
   aws s3 ls my-bucket-2 --profile instancecrossaccount
   ```

   이 명령은 `efgh` 역할에 할당된 권한이 현재 AWS 계정에 사용자를 나열하는 것을 허용하는 경우에 작동합니다.
3. 111111111111 계정의 원래 EC2 인스턴스 프로파일 권한을 반환하려면 `--profile` 파라미터 없이 CLI 명령을 실행합니다.

자세한 내용은 AWS Command Line Interface 사용 설명서의 역할 수임을 참조하세요.

IAM 역할로 전환(Tools for Windows PowerShell)

역할은 필요한 AWS 리소스에 액세스하는 데 사용할 수 있는 일련의 권한을 지정합니다. 이러한 면에서 AWS Identity and Access Management(IAM)의 사용자와 비슷합니다. 사용자로 로그인할 때는 특정 권한이 부여됩니다. 하지만 역할로 로그인하지는 못하기 때문에 일단 로그인한 후에 역할로 전환할 수 있습니다. 이 경우 초기의 사용자 권한은 잠시 무효화되고 역할에게 할당된 권한이 부여됩니다. 역할은 자신의 계정 또는 그 밖의 다른 AWS 계정에 속한 것일 수 있습니다. 역할과 역할의 이점, 역할 생성 및 구성 방법에 대한 자세한 내용은 IAM 역할 및 IAM 역할 생성 섹션을 참조하세요.

> **Important**
>
> IAM 사용자의 권한과 전환 대상인 역할의 권한은 누적되지 않습니다. 한 번에 오직 하나의 권한 집합만이 활성화됩니다. 어떤 역할로 전환할 때 사용자 권한은 일시적으로 포기하고 역할에 할당된 권한을 가지고 작업합니다. 역할을 끝내면 사용자 권한이 자동으로 회복됩니다.

이 섹션에서는 AWS Tools for Windows PowerShell에서 명령줄로 작업할 때 역할을 전환하는 방법에 대해 기술합니다.

개발 환경에서 계정을 하나 갖고 있는데 이따금 명령줄에서 Tools for Windows PowerShell을 사용하여 프로덕션 환경으로 작업해야 할 때가 있다고 가정합시다. 사용할 수 있는 액세스 키 자격 증명 세트가 이미 하나 있습니다. 이 세트는 표준 IAM 사용자에게 할당된 액세스 키 페어일 수도 있고, 페더레이션 사용자로 로그인한 경우에는 초기에 할당된 역할에 대한 액세스 키 페어일 수도 있습니다. 이 자격 증명을 사용해 새 역할의 ARN을 파라미터로 전달하는 `Use-STSRole` cmdlet을 실행할 수 있습니다.

해당 명령은 요청된 역할에 대한 임시 보안 자격 증명을 반환합니다. 그런 다음 생산 중인 리소스에 액세스할 수 있는 해당 역할의 권한으로 후속 PowerShell 명령에서 이 자격 증명을 사용할 수 있습니다. 한 번에 한 가지 권한 세트만 적용되기 때문에 해당 역할을 사용하는 동안에는 개발 계정의 사용자 권한을 사용할 수 없습니다.

> **Note**
>
> 보안을 위해 관리자는 AWS CloudTrail 로그를 검토하여 AWS에서 누가 작업을 수행했는지 확인할 수 있습니다. 관리자는 사용자가 역할을 수임할 때 소스 자격 증명이나 역할 세션 이름을 지정하도록 요구할 수 있습니다. 자세한 내용은 `sts:SourceIdentity` 및 `sts:RoleSessionName`을(를) 참조하세요.

모든 액세스 키와 토큰은 예시일 뿐이며 표시된 대로 사용할 수 없습니다. 라이브 환경의 적절한 값으로 바꾸세요.

역할로 전환하려면(Tools for Windows PowerShell)

1. PowerShell 명령 프롬프트를 열고 현재 IAM 사용자 또는 페더레이션 역할의 액세스 키를 사용하도록 기본 프로필을 구성합니다. 이전에 Tools for Windows PowerShell을 사용했다면 이미 그렇게 했을 가능성이 큽니다. AWS 계정 루트 사용자가 아닌 IAM 사용자로 로그인한 경우에 한해 역할을 바꿀 수 있다는 것에 유의하십시오.

   ```
   PS C:\> Set-AWSCredentials -AccessKey AKIAIOSFODNN7EXAMPLE -
   SecretKey wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY -StoreAs MyMainUserProfile
   PS C:\> Initialize-AWSDefaults -ProfileName MyMainUserProfile -Region us-east-2
   ```

   자세한 내용은 AWS Tools for Windows PowerShell 사용 설명서에서 AWS 자격 증명 사용을 참조하세요.

2. 새 역할에 대한 자격 증명을 가져오려면, 다음 명령을 실행하여 123456789012 계정의 *RoleName* 역할로 전환합니다. 해당 역할을 만든 계정 관리자에게서 역할 ARN을 받습니다. 그 명령은 세션 이름도 제공할 것을 요구합니다. 세션 이름에 대해서는 어떤 텍스트도 선택 가능합니다. 다음 명령은 자격 증명을 요청한 다음, 반환된 결과 객체로부터 `Credentials` 속성 객체를 캡쳐해 `$Creds` 변수에 저장합니다.

   ```
   PS C:\> $Creds = (Use-STSRole -RoleArn "arn:aws:iam::123456789012:role/RoleName" -
   RoleSessionName "MyRoleSessionName").Credentials
   ```

$Creds는 다음 절차에서 필요한 AccessKeyId, SecretAccessKey 및 SessionToken 요소를 포함하는 객체입니다. 다음 샘플 명령은 전형적인 값을 보여줍니다.

```
PS C:\> $Creds.AccessKeyId
AKIAIOSFODNN7EXAMPLE

PS C:\> $Creds.SecretAccessKey
wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

PS C:\> $Creds.SessionToken
AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPLEeYjs1M2FUIgIJx9tQqNMBEXAMPLECvSRyh0FW7jEXAMPLEW+vE/7s1HRp
XviG7b+qYf4nD00EXAMPLEmj4wxS04L/uZEXAMPLECihzFB5lTYLto9dyBgSDyEXAMPLE9/
g7QRUhZp4bqbEXAMPLENwGPy
Oj59pFA4lNKCIkVgkREXAMPLEjlzxQ7y52gekeVEXAMPLEDiB9ST3UuysgsKdEXAMPLE1TVastU1A0SKFEXAMPLEiyw
C
s8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP+4eZScEXAMPLEsnf87eNhyDHq6ikBQ==

PS C:\> $Creds.Expiration
Thursday, June 18, 2018 2:28:31 PM
```

3. 후속 명령에 대해 이 자격 증명을 사용하려면 -Credential 파라미터로 자격 증명을 포함시키세요. 예를 들어 다음 명령은 그 역할에 iam:ListRoles 권한이 부여되고, 따라서 Get-IAMRoles cmdlet을 실행할 수 있는 경우에 한해 역할에서 얻은 자격 증명을 사용하고 작동됩니다.

```
PS C:\> get-iamroles -Credential $Creds
```

4. 원래 자격 증명으로 돌아가려면 -Credentials $Creds 파라미터 사용을 중지하고 PowerShell이 기본 프로필에 저장된 자격 증명으로 복귀할 수 있도록 허용하기만 하면 됩니다.

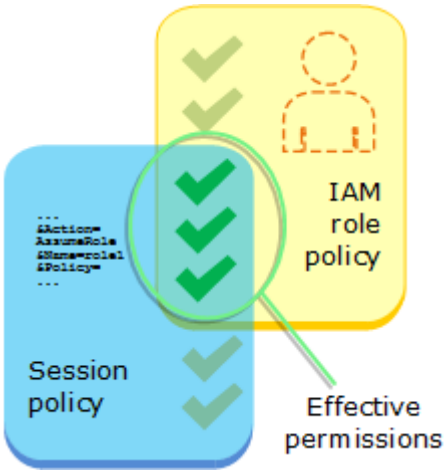
IAM 역할로 전환(AWS API)

역할은 AWS 리소스에 액세스하는 데 사용할 수 있는 일련의 권한을 지정합니다. 이러한 면에서 IAM 사용자와 비슷합니다. 보안 주체(개인 또는 애플리케이션)은 역할을 수임하여 필요한 작업을 수행하고 AWS 리소스와 상호작용할 수 있는 임시 권한을 부여받습니다. 역할은 자신의 계정 또는 그 밖의 다른 AWS 계정에 속한 것일 수 있습니다. 역할과 역할의 이점, 역할 생성 및 구성 방법에 대한 자세한 내용은 IAM 역할 및 IAM 역할 생성 섹션을 참조하세요. 역할을 수임하는 데 사용할 수 있는 여러 방법을 알아보려면 역할 수임 방법 섹션을 참조하세요.

> **Important**
>
> IAM 사용자의 권한과 수임한 역할의 권한은 누적되지 않습니다. 한 번에 오직 하나의 권한 집합만이 활성화됩니다. 어떤 역할을 수임할 때 이전 사용자 또는 역할 권한은 일시적으로 포기하고 해당 역할에 할당된 권한을 가지고 작업합니다. 이 역할을 끝내면 원래 권한이 자동으로 회복됩니다.

이때 역할 위임을 위해 애플리케이션은 AWS STS `AssumeRole` API 작업을 호출하고 사용할 역할의 ARN을 전달합니다. 이 작업은 임시 자격 증명으로 사용하여 새 세션을 생성합니다. 이 세션에는 해당 역할에 대한 자격 증명 기반 정책과 동일한 권한이 지정됩니다.

`AssumeRole` 호출 시 선택적으로 인라인 또는 관리형 세션 정책을 전달할 수 있습니다. 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 자격 증명 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. `Policy` 파라미터를 사용하여 단일 JSON 인라인 세션 정책 문서를 전달할 수 있습니다. `PolicyArns` 파라미터를 사용하여 최대 10개까지 관리형 세션 정책을 지정할 수 있습니다. 결과적으로 얻는 세션의 권한은 엔터티의 자격 증명 기반 정책과 세션 정책의 교집합입니다 세션 정책은 다른 사람에게 역할의 임시 보안 자격 증명을 부여할 필요가 있을 때 유용합니다. 후속 AWS API 호출 시에도 역할의 임시 자격 증명을 사용하여 역할이 속한 계정의 리소스에 액세스할 수 있습니다. 세션 정책을 사용하여 자격 증명 기반 정책에서 허용되는 권한을 부여할 수는 없습니다. 이 역할의 효과적인 권한을 AWS가 어떻게 결정하는지 자세히 알아보려면 정책 평가 로직 섹션을 참조하세요.



IAM 사용자 또는 역할을 이미 사용 중인 외부에서 인증된 사용자(SAML 또는 OIDC)로 로그인한 경우 `AssumeRole`을(를) 호출할 수 있습니다. 또한 역할을 사용해 두 번째 역할을 수임하는 역할 함께 묶기를 사용할 수도 있습니다. AWS 계정 루트 사용자로 로그인되어 있을 때는 역할을 수임할 수 없습니다.

기본적으로 역할 세션은 한 시간 동안 지속됩니다. AWS STS `AssumeRole*` API 작업을 사용하여 역할을 수임하는 경우 `DurationSeconds` 파라미터에 대한 값을 지정할 수 있습니다. 이 값의 범위는 900초(15분)에서 해당 역할에 대한 최대 세션 기간 설정까지일 수 있습니다. 역할에 대한 최댓값을 확인하는 방법을 알아보려면 역할의 최대 세션 기간 업데이트 섹션을 참조하세요.

역할 함께 묶기를 사용하는 경우 세션은 최대 1시간으로 제한됩니다. 그런 다음 `DurationSeconds` 파라미터를 사용하여 1시간보다 큰 값을 입력하면 이 작업에 실패합니다.

> **Note**
>
> 보안을 위해 관리자는 AWS CloudTrail 로그를 검토하여 AWS에서 누가 작업을 수행했는지 확인할 수 있습니다. 관리자는 사용자가 역할을 수임할 때 소스 자격 증명이나 역할 세션 이름을 지정하도록 요구할 수 있습니다. 자세한 내용은 `sts:SourceIdentity` 및 `sts:RoleSessionName`을(를) 참조하세요.

다음 코드 예제에서는 사용자를 생성하고 역할을 수임하는 방법을 보여줍니다.

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 자격 증명 공급자를 통한 페더레이션을 사용하세요.

- 권한이 없는 사용자를 생성합니다.
- 계정에 대한 Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할을 생성합니다.
- 사용자가 역할을 수임할 수 있도록 정책을 추가합니다.
- 역할을 수임하고 임시 보안 인증 정보를 사용하여 S3 버킷을 나열한 후 리소스를 정리합니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


namespace IAMActions;

public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;

    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
    /// Add an existing IAM user to an existing IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to add.</param>
    /// <param name="groupName">The name of the group to add the user to.</param>
```

```
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AddUserToGroupAsync(string userName, string
 groupName)
    {
        var response = await _IAMService.AddUserToGroupAsync(new
 AddUserToGroupRequest
        {
            GroupName = groupName,
            UserName = userName,
        });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
```

```
        var response = await _IAMService.CreateAccessKeyAsync(new
 CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }


    /// <summary>
    /// Create an IAM group.
    /// </summary>
    /// <param name="groupName">The name to give the IAM group.</param>
    /// <returns>The IAM group that was created.</returns>
    public async Task<Group> CreateGroupAsync(string groupName)
    {
        var response = await _IAMService.CreateGroupAsync(new CreateGroupRequest
 { GroupName = groupName });
        return response.Group;
    }


    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }
```

```
    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
 rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }


    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }


    /// <summary>
```

```
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupAsync(string groupName)
    {
        var response = await _IAMService.DeleteGroupAsync(new DeleteGroupRequest
{ GroupName = groupName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Delete an IAM policy associated with an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group associated with the
    /// policy.</param>
    /// <param name="policyName">The name of the policy to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupPolicyAsync(string groupName, string
policyName)
    {
        var request = new DeleteGroupPolicyRequest()
        {
            GroupName = groupName,
            PolicyName = policyName,
        };

        var response = await _IAMService.DeleteGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
```

```
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
 { RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
 { UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
```

```
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }
```

```
    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }


    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }


    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
 { UserName = userName });
        return response.User;
```

```
    }


    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }


    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }
```

```
    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }


    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }


    /// <summary>
    /// List IAM roles.
    /// </summary>
```

```
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }


    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }


    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
```

```
    }


    /// <summary>
    /// Remove a user from an IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to remove.</param>
    /// <param name="groupName">The name of the IAM group to remove the user
from.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> RemoveUserFromGroupAsync(string userName, string
groupName)
    {
        // Remove the user from the group.
        var removeUserRequest = new RemoveUserFromGroupRequest()
        {
            UserName = userName,
            GroupName = groupName,
        };

        var response = await
_IAMService.RemoveUserFromGroupAsync(removeUserRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutGroupPolicyAsync(string groupName, string
policyName, string policyDocument)
    {
        var request = new PutGroupPolicyRequest
        {
            GroupName = groupName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };
```

```
        var response = await _IAMService.PutGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM user.
    /// </summary>
    /// <param name="userName">The name of the IAM user.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
 policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutUserPolicyAsync(string userName, string
 policyName, string policyDocument)
    {
        var request = new PutUserPolicyRequest
        {
            UserName = userName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
```

```
        };

        var response = await _IAMService.PutUserPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Wait for a new access key to be ready to use.
    /// </summary>
    /// <param name="accessKeyId">The Id of the access key.</param>
    /// <returns>A boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
    {
        var keyReady = false;

        do
        {
            try
            {
                var response = await _IAMService.GetAccessKeyLastUsedAsync(
                    new GetAccessKeyLastUsedRequest { AccessKeyId =
 accessKeyId });
                if (response.UserName is not null)
                {
                    keyReady = true;
                }
            }
            catch (NoSuchEntityException)
            {
                keyReady = false;
            }
        } while (!keyReady);

        return keyReady;
    }
}



using Microsoft.Extensions.Configuration;

namespace IAMBasics;

public class IAMBasics
```

```
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonIdentityManagementService>()
            .AddTransient<IAMWrapper>()
            .AddTransient<UIWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMBasics>();


        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // Values needed for user, role, and policies.
        string userName = configuration["UserName"]!;
        string s3PolicyName = configuration["S3PolicyName"]!;
        string roleName = configuration["RoleName"]!;


        var iamWrapper = host.Services.GetRequiredService<IAMWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        uiWrapper.DisplayBasicsOverview();
        uiWrapper.PressEnter();

        // First create a user. By default, the new user has
```

```
        // no permissions.
        uiWrapper.DisplayTitle("Create User");
        Console.WriteLine($"Creating a new user with user name: {userName}.");
        var user = await iamWrapper.CreateUserAsync(userName);
        var userArn = user.Arn;

        Console.WriteLine($"Successfully created user: {userName} with ARN:
{userArn}.");
        uiWrapper.WaitABit(15, "Now let's wait for the user to be ready for
use.");

        // Define a role policy document that allows the new user
        // to assume the role.
        string assumeRolePolicyDocument = "{" +
          "\"Version\": \"2012-10-17\"," +
          "\"Statement\": [{" +
              "\"Effect\": \"Allow\"," +
              "\"Principal\": {" +
              $" \"AWS\": \"{userArn}\"" +
              "}," +
              "\"Action\": \"sts:AssumeRole\"" +
          "}]" +
        "}";

        // Permissions to list all buckets.
        string policyDocument = "{" +
            "\"Version\": \"2012-10-17\"," +
            " \"Statement\" : [{" +
                " \"Action\" : [\"s3:ListAllMyBuckets\"]," +
                " \"Effect\" : \"Allow\"," +
                " \"Resource\" : \"*\"" +
            "}]" +
        "}";

        // Create an AccessKey for the user.
        uiWrapper.DisplayTitle("Create access key");
        Console.WriteLine("Now let's create an access key for the new user.");
        var accessKey = await iamWrapper.CreateAccessKeyAsync(userName);

        var accessKeyId = accessKey.AccessKeyId;
        var secretAccessKey = accessKey.SecretAccessKey;

        Console.WriteLine($"We have created the access key with Access key id:
{accessKeyId}.");
```

```
        Console.WriteLine("Now let's wait until the IAM access key is ready to
use.");
        var keyReady = await iamWrapper.WaitUntilAccessKeyIsReady(accessKeyId);

        // Now try listing the Amazon Simple Storage Service (Amazon S3)
        // buckets. This should fail at this point because the user doesn't
        // have permissions to perform this task.
        uiWrapper.DisplayTitle("Try to display Amazon S3 buckets");
        Console.WriteLine("Now let's try to display a list of the user's Amazon
S3 buckets.");
        var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
        var stsClient1 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        var s3Wrapper = new S3Wrapper(s3Client1, stsClient1);
        var buckets = await s3Wrapper.ListMyBucketsAsync();

        Console.WriteLine(buckets is null
            ? "As expected, the call to list the buckets has returned a null
list."
            : "Something went wrong. This shouldn't have worked.");

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create IAM role");
        Console.WriteLine($"Creating the role: {roleName}");

        // Creating an IAM role to allow listing the S3 buckets. A role name
        // is not case sensitive and must be unique to the account for which it
        // is created.
        var roleArn = await iamWrapper.CreateRoleAsync(roleName,
assumeRolePolicyDocument);

        uiWrapper.PressEnter();

        // Create a policy with permissions to list S3 buckets.
        uiWrapper.DisplayTitle("Create IAM policy");
        Console.WriteLine($"Creating the policy: {s3PolicyName}");
        Console.WriteLine("with permissions to list the Amazon S3 buckets for the
account.");
        var policy = await iamWrapper.CreatePolicyAsync(s3PolicyName,
policyDocument);
```

```
        // Wait 15 seconds for the IAM policy to be available.
        uiWrapper.WaitABit(15, "Waiting for the policy to be available.");

        // Attach the policy to the role you created earlier.
        uiWrapper.DisplayTitle("Attach new IAM policy");
        Console.WriteLine("Now let's attach the policy to the role.");
        await iamWrapper.AttachRolePolicyAsync(policy.Arn, roleName);

        // Wait 15 seconds for the role to be updated.
        Console.WriteLine();
        uiWrapper.WaitABit(15, "Waiting for the policy to be attached.");

        // Use the AWS Security Token Service (AWS STS) to have the user
        // assume the role we created.
        var stsClient2 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        // Wait for the new credentials to become valid.
        uiWrapper.WaitABit(10, "Waiting for the credentials to be valid.");

        var assumedRoleCredentials = await
s3Wrapper.AssumeS3RoleAsync("temporary-session", roleArn);

        // Try again to list the buckets using the client created with
        // the new user's credentials. This time, it should work.
        var s3Client2 = new AmazonS3Client(assumedRoleCredentials);

        s3Wrapper.UpdateClients(s3Client2, stsClient2);

        buckets = await s3Wrapper.ListMyBucketsAsync();

        uiWrapper.DisplayTitle("List Amazon S3 buckets");
        Console.WriteLine("This time we should have buckets to list.");
        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName} created:
{bucket.CreationDate}");
            });
        }

        uiWrapper.PressEnter();
```

```
        // Now clean up all the resources used in the example.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Thank you for watching. The IAM Basics demo is
 complete.");
        Console.WriteLine("Please wait while we clean up the resources we
 created.");

        await iamWrapper.DetachRolePolicyAsync(policy.Arn, roleName);

        await iamWrapper.DeletePolicyAsync(policy.Arn);

        await iamWrapper.DeleteRoleAsync(roleName);

        await iamWrapper.DeleteAccessKeyAsync(accessKeyId, userName);

        await iamWrapper.DeleteUserAsync(userName);

        uiWrapper.PressEnter();

        Console.WriteLine("All done cleaning up our resources. Thank you for your
 patience.");
    }
}


namespace IamScenariosCommon;

using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
    /// Constructor for the S3Wrapper class.
    /// </summary>
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
```

```
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</
param>
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
 roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };

        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
    }


    /// <summary>
    /// Delete an S3 bucket.
    /// </summary>
    /// <param name="bucketName">Name of the S3 bucket to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
 { BucketName = bucketName });
        return result.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the buckets that are owned by the user's account.
```

```
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Create a new S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name for the new bucket.</param>
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
{ BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
```

```
    }
}


namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
 (IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it
 full access to Amazon S3.");
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
 creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
```

```
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries
to list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
```

```
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
###############################################################################
# function iam_create_user_assume_role
#
# Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
#
#     "IAM access" permissions are needed to run this code.
#     "STS assume role" permissions are needed to run this code. (Note: It might
 be necessary to
#           create a custom policy).
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function iam_create_user_assume_role() {
  {
    if [ "$IAM_OPERATIONS_SOURCED" != "True" ]; then

      source ./iam_operations.sh
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the IAM create user and assume role demo."
  echo
  echo "This demo will create an IAM user, create an IAM role, and apply the role
 to the user."
  echo_repeat "*" 88
  echo

  echo -n "Enter a name for a new IAM user: "
```

```
  get_input
  user_name=$get_input_result

  local user_arn
  user_arn=$(iam_create_user -u "$user_name")

  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created demo IAM user named $user_name"
  else
    errecho "$user_arn"
    errecho "The user failed to create. This demo will exit."
    return 1
  fi

  local access_key_response
  access_key_response=$(iam_create_user_access_key -u "$user_name")
  # shellcheck disable=SC2181
  if [[ ${?} != 0 ]]; then
    errecho "The access key failed to create. This demo will exit."
    clean_up "$user_name"
    return 1
  fi

  IFS=$'\t ' read -r -a access_key_values <<<"$access_key_response"
  local key_name=${access_key_values[0]}
  local key_secret=${access_key_values[1]}

  echo "Created access key named $key_name"

  echo "Wait 10 seconds for the user to be ready."
  sleep 10
  echo_repeat "*" 88
  echo

  local iam_role_name
  iam_role_name=$(generate_random_name "test-role")
  echo "Creating a role named $iam_role_name with user $user_name as the
principal."

  local assume_role_policy_document="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
```

```
        \"Principal\": {\"AWS\": \"$user_arn\"},
        \"Action\": \"sts:AssumeRole\"
        }]
    }"

  local role_arn
  role_arn=$(iam_create_role -n "$iam_role_name" -p
"$assume_role_policy_document")

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Created IAM role named $iam_role_name"
  else
    errecho "The role failed to create. This demo will exit."
    clean_up "$user_name" "$key_name"
    return 1
  fi

  local policy_name
  policy_name=$(generate_random_name "test-policy")
  local policy_document="{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]}"

  local policy_arn
  policy_arn=$(iam_create_policy -n "$policy_name" -p "$policy_document")
  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created  IAM policy named $policy_name"
  else
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name"
    return 1
  fi

  if (iam_attach_role_policy -n "$iam_role_name" -p "$policy_arn"); then
    echo "Attached policy $policy_arn to role $iam_role_name"
  else
    errecho "The policy failed to attach."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
    return 1
```

```
 fi

 local assume_role_policy_document="{
               \"Version\": \"2012-10-17\",
               \"Statement\": [{
                   \"Effect\": \"Allow\",
                   \"Action\": \"sts:AssumeRole\",
                   \"Resource\": \"$role_arn\"}]}"

 local assume_role_policy_name
 assume_role_policy_name=$(generate_random_name "test-assume-role-")

 # shellcheck disable=SC2181
 local assume_role_policy_arn
 assume_role_policy_arn=$(iam_create_policy -n "$assume_role_policy_name" -p
"$assume_role_policy_document")
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   echo "Created  IAM policy named $assume_role_policy_name for sts assume role"
 else
   errecho "The policy failed to create."
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn"
   return 1
 fi

 echo "Wait 10 seconds to give AWS time to propagate these new resources and
connections."
 sleep 10
 echo_repeat "*" 88
 echo

 echo "Try to list buckets without the new user assuming the role."
 echo_repeat "*" 88
 echo

 # Set the environment variables for the created user.
 # bashsupport disable=BP2001
 export AWS_ACCESS_KEY_ID=$key_name
 # bashsupport disable=BP2001
 export AWS_SECRET_ACCESS_KEY=$key_secret

 local buckets
 buckets=$(s3_list_buckets)
```

```
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   local bucket_count
   bucket_count=$(echo "$buckets" | wc -w | xargs)
   echo "There are $bucket_count buckets in the account. This should not have
happened."
 else
   errecho "Because the role with permissions has not been assumed, listing
buckets failed."
 fi

 echo
 echo_repeat "*" 88
 echo "Now assume the role $iam_role_name and list the buckets."
 echo_repeat "*" 88
 echo

 local credentials

 credentials=$(sts_assume_role -r "$role_arn" -n "AssumeRoleDemoSession")
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   echo "Assumed role $iam_role_name"
 else
   errecho "Failed to assume role."
   export AWS_ACCESS_KEY_ID=""
   export AWS_SECRET_ACCESS_KEY=""
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
   return 1
 fi

 IFS=$'\t ' read -r -a credentials <<<"$credentials"

 export AWS_ACCESS_KEY_ID=${credentials[0]}
 export AWS_SECRET_ACCESS_KEY=${credentials[1]}
 # bashsupport disable=BP2001
 export AWS_SESSION_TOKEN=${credentials[2]}

 buckets=$(s3_list_buckets)

 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
```

```
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. Listing buckets
 succeeded because of "
    echo "the assumed role."
  else
    errecho "Failed to list buckets. This should not happen."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    export AWS_SESSION_TOKEN=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
 "$policy_arn" "$assume_role_policy_arn"
    return 1
  fi

  local result=0
  export AWS_ACCESS_KEY_ID=""
  export AWS_SECRET_ACCESS_KEY=""

  echo
  echo_repeat "*" 88
  echo "The created resources will now be deleted."
  echo_repeat "*" 88
  echo

  clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
 "$assume_role_policy_arn"

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    result=1
  fi

  return $result
}
```

이 시나리오에 사용된 IAM 함수입니다.

```
###############################################################################
# function iam_user_exists
#
```

```
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
###############################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}

###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
```

```
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an WS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
```

```
  iecho "    User name:   $user_name"
  iecho ""

  # If the user already exists, we don't want to try to create it.
  if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
  fi

  response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
  echo "function iam_create_user_access_key"
  echo "Creates an AWS Identity and Access Management (IAM) key pair."
  echo "  -u user_name   The name of the IAM user."
  echo "  [-f file_name]   Optional file name for the access key output."
  echo ""
}

# Retrieve the calling parameters.
while getopts "u:f:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    f) file_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

response=$(aws iam create-access-key \
  --user-name "$user_name" \
  --output text)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-access-key operation failed.$response"
  return 1
```

```
  fi

  if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
  fi

  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}

###############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_json -- The assume role policy document."
    echo ""
  }
```

```
# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-role \
  --role-name "$role_name" \
  --assume-role-policy-document "$policy_document" \
  --output text \
  --query Role.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-role operation failed.\n$response"
  return 1
fi
```

```
  echo "$response"

  return 0
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) policy_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
```

```
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_name" ]]; then
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
```

```
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
```

```
    usage
    return 1
  fi

  response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
```

```
  echo "  -p policy_ARN -- The IAM policy document ARN."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy ARN with the -p parameter."
  usage
  return 1
fi

response=$(aws iam detach-role-policy \
  --role-name "$role_name" \
  --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
  return 1
```

```
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an WS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
```

```
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Policy arn:  $policy_arn"
  iecho ""

  response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_role() {
```

```
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an WS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")

  local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an WS Identity and Access Management (IAM) access key for the
 specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:k:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      k) access_key="${OPTARG}" ;;
```

```
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

if [[ -z "$access_key" ]]; then
  errecho "ERROR: You must provide an access key with the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Username:   $user_name"
iecho "    Access key:   $access_key"
iecho ""

response=$(aws iam delete-access-key \
  --user-name "$user_name" \
  --access-key-id "$access_key")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
  return 1
fi

iecho "delete-access-key response:$response"
iecho
```

```
  return 0
}

###############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an WS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
  fi

  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- API 세부 정보는 AWS CLI 명령 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole

- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace AwsDoc {
    namespace IAM {

        //! Cleanup by deleting created entities.
        /*!
          \sa DeleteCreatedEntities
          \param client: IAM client.
          \param role: IAM role.
          \param user: IAM user.
          \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                          const Aws::IAM::Model::Role &role,
                                          const Aws::IAM::Model::User &user,
                                          const Aws::IAM::Model::Policy &policy);
    }

    static const int LIST_BUCKETS_WAIT_SEC = 20;
```

```
    static const char ALLOCATION_TAG[] = "example_code";
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
 necessary to
//    create a custom policy).
/*!
  \sa iamCreateUserAssumeRoleScenario
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            std::cout << "Successfully created IAM user " << userName <<
 std::endl;
        }

        user = outcome.GetResult().GetUser();
    }
```

```
    // 2. Create a role.
    {
        // Get the IAM user for the current client in order to access its ARN.
        Aws::String iamUserArn;
        {
            Aws::IAM::Model::GetUserRequest request;
            Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error getting Iam user. " <<
                          outcome.GetError().GetMessage() << std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            else {
                std::cout << "Successfully retrieved Iam user "
                          << outcome.GetResult().GetUser().GetUserName()
                          << std::endl;
            }

            iamUserArn = outcome.GetResult().GetUser().GetArn();
        }

        Aws::IAM::Model::CreateRoleRequest request;

        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleName = "iam-demo-role-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleName(roleName);

        // Build policy document for role.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");

        Aws::Utils::Document jsonPrincipal;
        jsonPrincipal.WithString("AWS", iamUserArn);
        jsonStatement.WithObject("Principal", jsonPrincipal);
        jsonStatement.WithString("Action", "sts:AssumeRole");
        jsonStatement.WithObject("Condition", Aws::Utils::Document());

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
```

```
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Setting policy for role\n   "
                  << policyDocument.View().WriteCompact() << std::endl;

        // Set role policy document as JSON string.

request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating role. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a role with name " << roleName
                      << std::endl;
        }

        role = outcome.GetResult().GetRole();
    }

    // 3. Create an IAM policy.
    {
        Aws::IAM::Model::CreatePolicyRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String policyName = "iam-demo-policy-" +
                                 Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetPolicyName(policyName);

        // Build IAM policy document.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");
        jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
        jsonStatement.WithString("Resource", "arn:aws:s3:::*");

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
```

```
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Creating a policy.\n   " <<
policyDocument.View().WriteCompact()
                  << std::endl;

        // Set IAM policy document as JSON string.
        request.SetPolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreatePolicyOutcome outcome =
client.CreatePolicy(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a policy with name, " <<
policyName <<
                      "." << std::endl;
        }

        policy = outcome.GetResult().GetPolicy();
    }

    // 4. Assume the new role using the AWS Security Token Service (STS).
    Aws::STS::Model::Credentials credentials;
    {
        Aws::STS::STSClient stsClient(clientConfig);

        Aws::STS::Model::AssumeRoleRequest request;
        request.SetRoleArn(role.GetArn());
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleSessionName = "iam-demo-role-session-" +

Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleSessionName(roleSessionName);

        Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

        // Repeatedly call AssumeRole, because there is often a delay
```

```
        // before the role is available to be assumed.
        // Repeat at most 20 times when access is denied.
        int count = 0;
        while (true) {
            assumeRoleOutcome = stsClient.AssumeRole(request);
            if (!assumeRoleOutcome.IsSuccess()) {
                if (count > 20 ||
                    assumeRoleOutcome.GetError().GetErrorType() !=
                    Aws::STS::STSErrors::ACCESS_DENIED) {
                    std::cerr << "Error assuming role after 20 tries. " <<
                              assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

                    DeleteCreatedEntities(client, role, user, policy);
                    return false;
                }
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                std::cout << "Successfully assumed the role after " << count
                          << " seconds." << std::endl;
                break;
            }
            count++;
        }

        credentials = assumeRoleOutcome.GetResult().GetCredentials();
    }


    // 5. List objects in the bucket (This should fail).
    {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if (listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets. " <<
```

```
                        listBucketsOutcome.GetError().GetMessage() <<
std::endl;
            }
            else {
                std::cout
                        << "Access to list buckets denied because privileges have
not been applied."
                        << std::endl;
            }
        }
        else {
            std::cerr
                    << "Successfully retrieved bucket lists when this should not
happen."
                    << std::endl;
        }
    }

    // 6. Attach the policy to the role.
    {
        Aws::IAM::Model::AttachRolePolicyRequest request;
        request.SetRoleName(role.GetRoleName());
        request.WithPolicyArn(policy.GetArn());

        Aws::IAM::Model::AttachRolePolicyOutcome outcome =
client.AttachRolePolicy(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully attached the policy with name, "
                      << policy.GetPolicyName() <<
                      ", to the role, " << role.GetRoleName() << "." <<
std::endl;
        }
    }

    int count = 0;
    // 7. List objects in the bucket (this should succeed).
```

```
    // Repeatedly call ListBuckets, because there is often a delay
    // before the policy with ListBucket permissions has been applied to the
 role.
    // Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
    while (true) {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
 s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if ((count > LIST_BUCKETS_WAIT_SEC) ||
                listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets after " <<
 LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                          listBucketsOutcome.GetError().GetMessage() <<
 std::endl;
                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }

            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {

            std::cout << "Successfully retrieved bucket lists after " << count
                      << " seconds." << std::endl;
            break;
        }
        count++;
    }

    // 8. Delete all the created resources.
    return DeleteCreatedEntities(client, role, user, policy);
}

bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                        const Aws::IAM::Model::Role &role,
                                        const Aws::IAM::Model::User &user,
                                        const Aws::IAM::Model::Policy &policy) {
```

```
    bool result = true;
    if (policy.ArnHasBeenSet()) {
        // Detach the policy from the role.
        {
            Aws::IAM::Model::DetachRolePolicyRequest request;
            request.SetPolicyArn(policy.GetArn());
            request.SetRoleName(role.GetRoleName());

            Aws::IAM::Model::DetachRolePolicyOutcome outcome =
client.DetachRolePolicy(
                    request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error Detaching policy from roles. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully detached the policy with arn "
                          << policy.GetArn()
                          << " from role " << role.GetRoleName() << "." <<
std::endl;
            }
        }

        // Delete the policy.
        {
            Aws::IAM::Model::DeletePolicyRequest request;
            request.WithPolicyArn(policy.GetArn());

            Aws::IAM::Model::DeletePolicyOutcome outcome =
client.DeletePolicy(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error deleting policy. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully deleted the policy with arn "
                          << policy.GetArn() << std::endl;
            }
        }

    }
```

```
    if (role.RoleIdHasBeenSet()) {
        // Delete the role.
        Aws::IAM::Model::DeleteRoleRequest request;
        request.SetRoleName(role.GetRoleName());

        Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting role. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the role with name "
                      << role.GetRoleName() << std::endl;
        }
    }

    if (user.ArnHasBeenSet()) {
        // Delete the user.
        Aws::IAM::Model::DeleteUserRequest request;
        request.WithUserName(user.GetUserName());

        Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting user. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the user with name "
                      << user.GetUserName() << std::endl;
        }
    }

    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy

- CreateRole
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
// AssumeRoleScenario shows you how to use the AWS Identity and Access Management
 (IAM)
// service to perform the following actions:
//
// 1. Create a user who has no permissions.
// 2. Create a role that grants permission to list Amazon Simple Storage Service
//    (Amazon S3) buckets for the account.
// 3. Add a policy to let the user assume the role.
// 4. Try and fail to list buckets without permissions.
// 5. Assume the role and list S3 buckets using temporary credentials.
// 6. Delete the policy, role, and user.
type AssumeRoleScenario struct {
 sdkConfig aws.Config
 accountWrapper actions.AccountWrapper
 policyWrapper actions.PolicyWrapper
```

```
 roleWrapper actions.RoleWrapper
 userWrapper actions.UserWrapper
 questioner demotools.IQuestioner
 helper IScenarioHelper
 isTestRun bool
}

// NewAssumeRoleScenario constructs an AssumeRoleScenario instance from a
 configuration.
// It uses the specified config to get an IAM client and create wrappers for the
 actions
// used in the scenario.
func NewAssumeRoleScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
  helper IScenarioHelper) AssumeRoleScenario {
 iamClient := iam.NewFromConfig(sdkConfig)
 return AssumeRoleScenario{
  sdkConfig:   sdkConfig,
  accountWrapper: actions.AccountWrapper{IamClient: iamClient},
  policyWrapper:  actions.PolicyWrapper{IamClient: iamClient},
  roleWrapper:    actions.RoleWrapper{IamClient: iamClient},
  userWrapper:    actions.UserWrapper{IamClient: iamClient},
  questioner:     questioner,
  helper:         helper,
 }
}

// addTestOptions appends the API options specified in the original configuration
 to
// another configuration. This is used to attach the middleware stubber to
 clients
// that are constructed during the scenario, which is needed for unit testing.
func (scenario AssumeRoleScenario) addTestOptions(scenarioConfig *aws.Config) {
 if scenario.isTestRun {
  scenarioConfig.APIOptions = append(scenarioConfig.APIOptions,
 scenario.sdkConfig.APIOptions...)
 }
}

// Run runs the interactive scenario.
func (scenario AssumeRoleScenario) Run() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong with the demo.\n")
```

```
    log.Println(r)
   }
  }()

  log.Println(strings.Repeat("-", 88))
  log.Println("Welcome to the AWS Identity and Access Management (IAM) assume role
  demo.")
  log.Println(strings.Repeat("-", 88))

  user := scenario.CreateUser()
  accessKey := scenario.CreateAccessKey(user)
  role := scenario.CreateRoleAndPolicies(user)
  noPermsConfig := scenario.ListBucketsWithoutPermissions(accessKey)
  scenario.ListBucketsWithAssumedRole(noPermsConfig, role)
  scenario.Cleanup(user, role)

  log.Println(strings.Repeat("-", 88))
  log.Println("Thanks for watching!")
  log.Println(strings.Repeat("-", 88))
}

// CreateUser creates a new IAM user. This user has no permissions.
func (scenario AssumeRoleScenario) CreateUser() *types.User {
  log.Println("Let's create an example user with no permissions.")
  userName := scenario.questioner.Ask("Enter a name for the example user:",
  demotools.NotEmpty{})
  user, err := scenario.userWrapper.GetUser(userName)
  if err != nil {
   panic(err)
  }
  if user == nil {
   user, err = scenario.userWrapper.CreateUser(userName)
   if err != nil {
    panic(err)
   }
   log.Printf("Created user %v.\n", *user.UserName)
  } else {
   log.Printf("User %v already exists.\n", *user.UserName)
  }
  log.Println(strings.Repeat("-", 88))
  return user
}

// CreateAccessKey creates an access key for the user.
```

```
func (scenario AssumeRoleScenario) CreateAccessKey(user *types.User)
 *types.AccessKey {
 accessKey, err := scenario.userWrapper.CreateAccessKeyPair(*user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("Created access key %v for your user.", *accessKey.AccessKeyId)
 log.Println("Waiting a few seconds for your user to be ready...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return accessKey
}

// CreateRoleAndPolicies creates a policy that grants permission to list S3
 buckets for
// the current account and attaches the policy to a newly created role. It also
 adds an
// inline policy to the specified user that grants the user permission to assume
 the role.
func (scenario AssumeRoleScenario) CreateRoleAndPolicies(user *types.User)
 *types.Role {
 log.Println("Let's create a role and policy that grant permission to list S3
 buckets.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 listBucketsRole, err :=
 scenario.roleWrapper.CreateRole(scenario.helper.GetName(), *user.Arn)
 if err != nil {panic(err)}
 log.Printf("Created role %v.\n", *listBucketsRole.RoleName)
 listBucketsPolicy, err := scenario.policyWrapper.CreatePolicy(
  scenario.helper.GetName(), []string{"s3:ListAllMyBuckets"}, "arn:aws:s3:::*")
 if err != nil {panic(err)}
 log.Printf("Created policy %v.\n", *listBucketsPolicy.PolicyName)
 err = scenario.roleWrapper.AttachRolePolicy(*listBucketsPolicy.Arn,
 *listBucketsRole.RoleName)
 if err != nil {panic(err)}
 log.Printf("Attached policy %v to role %v.\n", *listBucketsPolicy.PolicyName,
  *listBucketsRole.RoleName)
 err = scenario.userWrapper.CreateUserPolicy(*user.UserName,
 scenario.helper.GetName(),
  []string{"sts:AssumeRole"}, *listBucketsRole.Arn)
 if err != nil {panic(err)}
 log.Printf("Created an inline policy for user %v that lets the user assume the
 role.\n",
  *user.UserName)
```

```
 log.Println("Let's give AWS a few seconds to propagate these new resources and
 connections...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return listBucketsRole
}

// ListBucketsWithoutPermissions creates an Amazon S3 client from the user's
 access key
// credentials and tries to list buckets for the account. Because the user does
 not have
// permission to perform this action, the action fails.
func (scenario AssumeRoleScenario) ListBucketsWithoutPermissions(accessKey
 *types.AccessKey) *aws.Config {
  log.Println("Let's try to list buckets without permissions. This should return
 an AccessDenied error.")
  scenario.questioner.Ask("Press Enter when you're ready.")
  noPermsConfig, err := config.LoadDefaultConfig(context.TODO(),
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *accessKey.AccessKeyId, *accessKey.SecretAccessKey, ""),
 ))
  if err != nil {panic(err)}

  // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&noPermsConfig)

  s3Client := s3.NewFromConfig(noPermsConfig)
  _, err = s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
  if err != nil {
   // The SDK for Go does not model the AccessDenied error, so check ErrorCode
 directly.
  var ae smithy.APIError
  if errors.As(err, &ae) {
   switch ae.ErrorCode() {
   case "AccessDenied":
    log.Println("Got AccessDenied error, which is the expected result because\n"
 +
     "the ListBuckets call was made without permissions.")
   default:
    log.Println("Expected AccessDenied, got something else.")
    panic(err)
   }
  }
```

```
  } else {
   log.Println("Expected AccessDenied error when calling ListBuckets without
 permissions,\n" +
    "but the call succeeded. Continuing the example anyway...")
 }
 log.Println(strings.Repeat("-", 88))
  return &noPermsConfig
}

// ListBucketsWithAssumedRole performs the following actions:
//
// 1. Creates an AWS Security Token Service (AWS STS) client from the config
 created from
//   the user's access key credentials.
// 2. Gets temporary credentials by assuming the role that grants permission to
 list the
//    buckets.
// 3. Creates an Amazon S3 client from the temporary credentials.
// 4. Lists buckets for the account. Because the temporary credentials are
 generated by
//    assuming the role that grants permission, the action succeeds.
func (scenario AssumeRoleScenario) ListBucketsWithAssumedRole(noPermsConfig
 *aws.Config, role *types.Role) {
 log.Println("Let's assume the role that grants permission to list buckets and
 try again.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 stsClient := sts.NewFromConfig(*noPermsConfig)
 tempCredentials, err := stsClient.AssumeRole(context.TODO(),
 &sts.AssumeRoleInput{
  RoleArn:           role.Arn,
  RoleSessionName:   aws.String("AssumeRoleExampleSession"),
  DurationSeconds:   aws.Int32(900),
 })
 if err != nil {
  log.Printf("Couldn't assume role %v.\n", *role.RoleName)
  panic(err)
 }
 log.Printf("Assumed role %v, got temporary credentials.\n", *role.RoleName)
 assumeRoleConfig, err := config.LoadDefaultConfig(context.TODO(),
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *tempCredentials.Credentials.AccessKeyId,
   *tempCredentials.Credentials.SecretAccessKey,
   *tempCredentials.Credentials.SessionToken),
  ),
```

```
 )
 if err != nil {panic(err)}

 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&assumeRoleConfig)

 s3Client := s3.NewFromConfig(assumeRoleConfig)
 result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
 if err != nil {
  log.Println("Couldn't list buckets with assumed role credentials.")
  panic(err)
 }
 log.Println("Successfully called ListBuckets with assumed role credentials, \n"
 +
  "here are some of them:")
 for i := 0; i < len(result.Buckets) && i < 5; i++ {
  log.Printf("\t%v\n", *result.Buckets[i].Name)
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup deletes all resources created for the scenario.
func (scenario AssumeRoleScenario) Cleanup(user *types.User, role *types.Role) {
 if scenario.questioner.AskBool(
  "Do you want to delete the resources created for this example? (y/n)", "y",
 ) {
   policies, err := scenario.roleWrapper.ListAttachedRolePolicies(*role.RoleName)
   if err != nil {panic(err)}
   for _, policy := range policies {
    err = scenario.roleWrapper.DetachRolePolicy(*role.RoleName,
 *policy.PolicyArn)
    if err != nil {panic(err)}
    err = scenario.policyWrapper.DeletePolicy(*policy.PolicyArn)
    if err != nil {panic(err)}
    log.Printf("Detached policy %v from role %v and deleted the policy.\n",
     *policy.PolicyName, *role.RoleName)
   }
   err = scenario.roleWrapper.DeleteRole(*role.RoleName)
   if err != nil {panic(err)}
   log.Printf("Deleted role %v.\n", *role.RoleName)

   userPols, err := scenario.userWrapper.ListUserPolicies(*user.UserName)
   if err != nil {panic(err)}
```

```
   for _, userPol := range userPols {
    err = scenario.userWrapper.DeleteUserPolicy(*user.UserName, userPol)
    if err != nil {panic(err)}
    log.Printf("Deleted policy %v from user %v.\n", userPol, *user.UserName)
   }
   keys, err := scenario.userWrapper.ListAccessKeys(*user.UserName)
   if err != nil {panic(err)}
   for _, key := range keys {
    err = scenario.userWrapper.DeleteAccessKey(*user.UserName, *key.AccessKeyId)
    if err != nil {panic(err)}
    log.Printf("Deleted access key %v from user %v.\n", *key.AccessKeyId,
 *user.UserName)
   }
   err = scenario.userWrapper.DeleteUser(*user.UserName)
   if err != nil {panic(err)}
   log.Printf("Deleted user %v.\n", *user.UserName)
   log.Println(strings.Repeat("-", 88))
 }

}
```

계정 작업을 래핑하는 구조체를 정의합니다.

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy() (*types.PasswordPolicy,
 error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(context.TODO(),
```

```
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}



// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders() ([]types.SAMLProviderListEntry,
 error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(context.TODO(),
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

정책 작업을 래핑하는 구조체를 정의합니다.

```

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect string
 Action []string
 Principal map[string]string `json:",omitempty"`
 Resource *string `json:",omitempty"`
```

```
}



// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(maxPolicies int32) ([]types.Policy,
 error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(context.TODO(),
 &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}



// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(policyName string, actions []string,
  resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
```

```
    Action: actions,
    Resource: aws.String(resourceArn),
   }},
  }
  policyBytes, err := json.Marshal(policyDoc)
  if err != nil {
   log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
  resourceArn, err)
   return nil, err
  }
  result, err := wrapper.IamClient.CreatePolicy(context.TODO(),
  &iam.CreatePolicyInput{
   PolicyDocument: aws.String(string(policyBytes)),
   PolicyName:     aws.String(policyName),
  })
  if err != nil {
   log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
  } else {
   policy = result.Policy
  }
  return policy, err
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(policyArn string) (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(context.TODO(), &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(policyArn string) error {
 _, err := wrapper.IamClient.DeletePolicy(context.TODO(), &iam.DeletePolicyInput{
```

```
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

역할 작업을 래핑하는 구조체를 정의합니다.

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(maxRoles int32) ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(context.TODO(),
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
 and
```

```
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(roleName string, trustedUserArn string)
 (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action: []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreateRole(context.TODO(),
 &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(roleName string) (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(context.TODO(),
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
```

```
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(serviceName string,
 description string) (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(context.TODO(),
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(roleName string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(context.TODO(),
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
 roleName, err)
 }
 return err
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(policyArn string, roleName string)
 error {
```

```
 _, err := wrapper.IamClient.AttachRolePolicy(context.TODO(),
 &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(roleName string)
 ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(context.TODO(),
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(roleName string, policyArn string)
 error {
 _, err := wrapper.IamClient.DetachRolePolicy(context.TODO(),
 &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
```

```
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(roleName string) ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(context.TODO(),
 &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(roleName string) error {
 _, err := wrapper.IamClient.DeleteRole(context.TODO(), &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

사용자 작업을 래핑하는 구조체를 정의합니다.

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(maxUsers int32) ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(context.TODO(), &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(userName string) (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(context.TODO(), &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
```

```
  user = result.User
 }
 return user, err
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(userName string) (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(context.TODO(),
 &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}



// CreateUserPolicy adds an inline policy to a user. This example creates a
 policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(userName string, policyName string,
 actions []string,
  roleArn string) error {
 policyDoc := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Action: actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
```

```
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(context.TODO(),
 &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(userName string) ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(context.TODO(),
 &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(userName string, policyName string)
 error {
 _, err := wrapper.IamClient.DeleteUserPolicy(context.TODO(),
 &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
```

```
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(userName string) error {
 _, err := wrapper.IamClient.DeleteUser(context.TODO(), &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(userName string)
 (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(context.TODO(),
 &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}



// DeleteAccessKey deletes an access key from a user.
```

```
func (wrapper UserWrapper) DeleteAccessKey(userName string, keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(context.TODO(),
 &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(context.TODO(),
 &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey

- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

IAM 사용자 작업을 래핑하는 함수를 생성합니다.

```
/*
  To run this Java V2 code example, set up your development environment,
 including your credentials.

  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html

  This example performs these operations:

  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3
 Service client object with the temporary credentials.
  6. Deletes the resources.
 */
```

```
public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static String userArn;

    public static void main(String[] args) throws Exception {

        final String usage = """

                Usage:
                    <username> <policyName> <roleName> <roleSessionName>
 <bucketName>\s

                Where:
                    username - The name of the IAM user to create.\s
                    policyName - The name of the policy to create.\s
                    roleName - The name of the role to create.\s
                    roleSessionName - The name of the session required for the
 assumeRole operation.\s
                    bucketName - The name of the Amazon S3 bucket from which
 objects are read.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        String policyName = args[1];
        String roleName = args[2];
```

```
        String roleSessionName = args[3];
        String bucketName = args[4];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IAM example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 1. Create the IAM user.");
        User createUser = createIAMUser(iam, userName);

        System.out.println(DASHES);
        userArn = createUser.arn();

        AccessKey myKey = createIAMAccessKey(iam, userName);
        String accessKey = myKey.accessKeyId();
        String secretKey = myKey.secretAccessKey();
        String assumeRolePolicyDocument = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                " \"AWS\": \"" + userArn + "\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        System.out.println(assumeRolePolicyDocument);
        System.out.println(userName + " was successfully created.");
        System.out.println(DASHES);
        System.out.println("2. Creates a policy.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully
created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
```

```
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the
role.");
        System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);

waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static String createIAMRole(IamClient iam, String rolename, String
json) {

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(json)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " +
response.role().arn());
```

```
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();
            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();
```

```
            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
            String polArn;
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);
            System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Invoke an Amazon S3 operation using the Assumed Role.
    public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
            String keySecret) {

        // Use the creds of the new IAM user that was created in this code
example.
        AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
        StsClient stsClient = StsClient.builder()
                .region(Region.US_EAST_1)

.credentialsProvider(StaticCredentialsProvider.create(credentials))
                .build();
```

```
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();
            String key = myCreds.accessKeyId();
            String secKey = myCreds.secretAccessKey();
            String secToken = myCreds.sessionToken();

            // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
            // invoking assumeRole.
            Region region = Region.US_EAST_1;
            S3Client s3 = S3Client.builder()
                    .credentialsProvider(

StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
                    .region(region)
                    .build();

            System.out.println("Created a S3Client using temp credentials.");
            System.out.println("Listing objects in " + bucketName);
            ListObjectsRequest listObjects = ListObjectsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.println("The name of the key is " + myValue.key());
                System.out.println("The owner is " + myValue.owner());
            }

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
   public static void deleteRole(IamClient iam, String roleName, String polArn)
{

       try {
           // First the policy needs to be detached.
           DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                   .policyArn(polArn)
                   .roleName(roleName)
                   .build();

           iam.detachRolePolicy(rolePolicyRequest);

           // Delete the policy.
           DeletePolicyRequest request = DeletePolicyRequest.builder()
                   .policyArn(polArn)
                   .build();

           iam.deletePolicy(request);
           System.out.println("*** Successfully deleted " + polArn);

           // Delete the role.
           DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                   .roleName(roleName)
                   .build();

           iam.deleteRole(roleRequest);
           System.out.println("*** Successfully deleted " + roleName);

       } catch (IamException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void deleteKey(IamClient iam, String username, String
accessKey) {
       try {
           DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                   .accessKeyId(accessKey)
                   .userName(username)
                   .build();

           iam.deleteAccessKey(request);
```

```
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("*** Successfully deleted " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy

- PutUserPolicy

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할과 IAM 사용자를 생성합니다. 사용자는 역할을 수임할 수 있는 권한만 있습니다. 역할을 수임한 후 임시 자격 증명을 사용하여 계정의 버킷을 나열합니다.

```
import {
  CreateUserCommand,
  GetUserCommand,
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachRolePolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario/index.js";

// Set the parameters.
const iamClient = new IAMClient({});
const userName = "test_name";
const policyName = "test_policy";
const roleName = "test_role";
```

```
/**
 * Create a new IAM user. If the user already exists, give
 * the option to delete and re-create it.
 * @param {string} name
 */
export const createUser = async (name, confirmAll = false) => {
  try {
    const { User } = await iamClient.send(
      new GetUserCommand({ UserName: name }),
    );
    const input = new ScenarioInput(
      "deleteUser",
      "Do you want to delete and remake this user?",
      { type: "confirm" },
    );
    const deleteUser = await input.handle({}, { confirmAll });
    // If the user exists, and you want to delete it, delete the user
    // and then create it again.
    if (deleteUser) {
      await iamClient.send(new DeleteUserCommand({ UserName: User.UserName }));
      await iamClient.send(new CreateUserCommand({ UserName: name }));
    } else {
      console.warn(
        `${name} already exists. The scenario may not work as expected.`,
      );
      return User;
    }
  } catch (caught) {
    // If there is no user by that name, create one.
    if (caught instanceof Error && caught.name === "NoSuchEntityException") {
      const { User } = await iamClient.send(
        new CreateUserCommand({ UserName: name }),
      );
      return User;
    } else {
      throw caught;
    }
  }
};

export const main = async (confirmAll = false) => {
  // Create a user. The user has no permissions by default.
  const User = await createUser(userName, confirmAll);
```

```
 if (!User) {
   throw new Error("User not created");
 }

 // Create an access key. This key is used to authenticate the new user to
 // Amazon Simple Storage Service (Amazon S3) and AWS Security Token Service
(AWS STS).
 // It's not best practice to use access keys. For more information, see
https://aws.amazon.com/iam/resources/best-practices/.
 const createAccessKeyResponse = await iamClient.send(
   new CreateAccessKeyCommand({ UserName: userName }),
 );

 if (
   !createAccessKeyResponse.AccessKey?.AccessKeyId ||
   !createAccessKeyResponse.AccessKey?.SecretAccessKey
 ) {
   throw new Error("Access key not created");
 }

 const {
   AccessKey: { AccessKeyId, SecretAccessKey },
 } = createAccessKeyResponse;

 let s3Client = new S3Client({
   credentials: {
     accessKeyId: AccessKeyId,
     secretAccessKey: SecretAccessKey,
   },
 });

 // Retry the list buckets operation until it succeeds. InvalidAccessKeyId is
 // thrown while the user and access keys are still stabilizing.
 await retry({ intervalInMs: 1000, maxRetries: 300 }, async () => {
   try {
     return await listBuckets(s3Client);
   } catch (err) {
     if (err instanceof Error && err.name === "InvalidAccessKeyId") {
       throw err;
     }
   }
 });
```

```
 // Retry the create role operation until it succeeds. A MalformedPolicyDocument
error
 // is thrown while the user and access keys are still stabilizing.
 const { Role } = await retry(
   {
     intervalInMs: 2000,
     maxRetries: 60,
   },
   () =>
     iamClient.send(
       new CreateRoleCommand({
         AssumeRolePolicyDocument: JSON.stringify({
           Version: "2012-10-17",
           Statement: [
             {
               Effect: "Allow",
               Principal: {
                 // Allow the previously created user to assume this role.
                 AWS: User.Arn,
               },
               Action: "sts:AssumeRole",
             },
           ],
         }),
         RoleName: roleName,
       }),
     ),
 );

 if (!Role) {
   throw new Error("Role not created");
 }

 // Create a policy that allows the user to list S3 buckets.
 const { Policy: listBucketPolicy } = await iamClient.send(
   new CreatePolicyCommand({
     PolicyDocument: JSON.stringify({
       Version: "2012-10-17",
       Statement: [
         {
           Effect: "Allow",
           Action: ["s3:ListAllMyBuckets"],
           Resource: "*",
         },
```

```
        ],
      }),
      PolicyName: policyName,
    }),
  );

  if (!listBucketPolicy) {
    throw new Error("Policy not created");
  }

  // Attach the policy granting the 's3:ListAllMyBuckets' action to the role.
  await iamClient.send(
    new AttachRolePolicyCommand({
      PolicyArn: listBucketPolicy.Arn,
      RoleName: Role.RoleName,
    }),
  );

  // Assume the role.
  const stsClient = new STSClient({
    credentials: {
      accessKeyId: AccessKeyId,
      secretAccessKey: SecretAccessKey,
    },
  });

  // Retry the assume role operation until it succeeds.
  const { Credentials } = await retry(
    { intervalInMs: 2000, maxRetries: 60 },
    () =>
      stsClient.send(
        new AssumeRoleCommand({
          RoleArn: Role.Arn,
          RoleSessionName: `iamBasicScenarioSession-${Math.floor(
            Math.random() * 1000000,
          )}`,
          DurationSeconds: 900,
        }),
      ),
  );

  if (!Credentials?.AccessKeyId || !Credentials?.SecretAccessKey) {
    throw new Error("Credentials not created");
  }
```

```
s3Client = new S3Client({
  credentials: {
    accessKeyId: Credentials.AccessKeyId,
    secretAccessKey: Credentials.SecretAccessKey,
    sessionToken: Credentials.SessionToken,
  },
});

// List the S3 buckets again.
// Retry the list buckets operation until it succeeds. AccessDenied might
// be thrown while the role policy is still stabilizing.
await retry({ intervalInMs: 2000, maxRetries: 60 }, () =>
  listBuckets(s3Client),
);

// Clean up.
await iamClient.send(
  new DetachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);

await iamClient.send(
  new DeletePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
  }),
);

await iamClient.send(
  new DeleteRoleCommand({
    RoleName: Role.RoleName,
  }),
);

await iamClient.send(
  new DeleteAccessKeyCommand({
    UserName: userName,
    AccessKeyId,
  }),
);

await iamClient.send(
```

```
    new DeleteUserCommand({
      UserName: userName,
    }),
  );
};

/**
 *
 * @param {S3Client} s3Client
 */
const listBuckets = async (s3Client) => {
  const { Buckets } = await s3Client.send(new ListBucketsCommand({}));

  if (!Buckets) {
    throw new Error("Buckets not listed");
  }

  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

IAM 사용자 작업을 래핑하는 함수를 생성합니다.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
 <bucketName>

    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
 operation.
        fileLocation - The file location to the JSON required to create the role
 (see Readme).
        bucketName - The name of the Amazon S3 bucket from which objects are
 read.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
    val roleSessionName = args[3]
    val fileLocation = args[4]
    val bucketName = args[5]

    createUser(userName)
```

```
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("*** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue: String =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
```

```
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?,
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
```

```
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role
 $roleNameVal")
    }
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String,
) {
    val stsClient =
        StsClient {
```

```
            region = "us-east-1"
        }

    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials =
        StaticCredentialsProvider {
            accessKeyId = key
            secretAccessKey = secKey
            sessionToken = secToken
        }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 =
        S3Client {
            credentialsProvider = staticCredentials
            region = "us-east-1"
        }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")

    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}
```

```
suspend fun deleteRole(
    roleNameVal: String,
    polArn: String,
) {
    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}
```

```
@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace Iam\Basics;

require 'vendor/autoload.php';
```

```
use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
echo("--------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("--------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
```

```
$inlinePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}
```

```
$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할과 IAM 사용자를 생성합니다. 사용자는 역할을 수임할 수 있는 권한만 있습니다. 역할을 수임한 후 임시 자격 증명을 사용하여 계정의 버킷을 나열합니다.

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError


def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print(".", end="")
        sys.stdout.flush()
    print()


def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 resource
                         that has permissions to create users, roles, and
 policies
                         in the account.
    :return: The newly created user, user key, and role.
    """
    try:
        user = iam_resource.create_user(UserName=f"demo-user-{uuid4()}")
        print(f"Created user {user.name}.")
    except ClientError as error:
        print(
```

```
            f"Couldn't create a user for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user_key = user.create_access_key_pair()
        print(f"Created access key pair for user.")
    except ClientError as error:
        print(
            f"Couldn't create access keys for user {user.name}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    try:
        role = iam_resource.create_role(
            RoleName=f"demo-role-{uuid4()}",
            AssumeRolePolicyDocument=json.dumps(
                {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"AWS": user.arn},
                            "Action": "sts:AssumeRole",
                        }
                    ],
                }
            ),
        )
        print(f"Created role {role.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a role for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        policy = iam_resource.create_policy(
```

```
            PolicyName=f"demo-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "s3:ListAllMyBuckets",
                            "Resource": "arn:aws:s3:::*",
                        }
                    ],
                }
            ),
        )
        role.attach_policy(PolicyArn=policy.arn)
        print(f"Created policy {policy.policy_name} and attached it to the
role.")
    except ClientError as error:
        print(
            f"Couldn't create a policy and attach it to role {role.name}. Here's
why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user.create_policy(
            PolicyName=f"demo-user-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "sts:AssumeRole",
                            "Resource": role.arn,
                        }
                    ],
                }
            ),
        )
        print(
            f"Created an inline policy for {user.name} that lets the user assume
"
```

```
            f"the role."
        )
    except ClientError as error:
        print(
            f"Couldn't create an inline policy for user {user.name}. Here's why:
 "
            f"{error.response['Error']['Message']}"
        )
        raise

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, role


def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
                     have permission to list buckets in the account.
    """
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        "s3", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
    )
    try:
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
        raise RuntimeError("Expected to get AccessDenied error when listing
 buckets!")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise


def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
```

```
    Assumes a role that grants permission to list the Amazon S3 buckets in the
account.
    Uses the temporary credentials from the role to list the buckets that are
owned
    by the assumed role's account.

    :param user_key: The access key of a user that has permission to assume the
role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        "sts", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
    )
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name
        )
        temp_credentials = response["Credentials"]
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(
            f"Couldn't assume role {assume_role_arn}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    # Create an S3 resource that can access the account with the temporary
credentials.
    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )
    print(f"Listing buckets for the assumed role's account:")
    try:
        for bucket in s3_resource.buckets.all():
            print(bucket.name)
    except ClientError as error:
        print(
            f"Couldn't list buckets for the account. Here's why: "
```

```
            f"{error.response['Error']['Message']}"
        )
        raise




def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print(
            "Couldn't detach policy, delete policy, or delete role. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
        for key in user.access_keys.all():
            key.delete()
            print("Deleted user's access key.")
        user.delete()
        print(f"Deleted {user.name}.")
    except ClientError as error:
        print(
            "Couldn't delete user policy or delete user. Here's why: "
            f"{error.response['Error']['Message']}"
        )
```

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the IAM create user and assume role demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn,
 "AssumeRoleDemoSession")
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
        print("Thanks for watching!")


if __name__ == "__main__":
    usage_demo()
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy

- PutUserPolicy

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할과 IAM 사용자를 생성합니다. 사용자는 역할을 수임할 수 있는 권한만 있습니다. 역할을 수임한 후 임시 자격 증명을 사용하여 계정의 버킷을 나열합니다.

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
    puts("Give AWS time to propagate resources...")
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_client.create_user(user_name: user_name).user
    @logger.info("Created demo user named #{user.user_name}.")
```

```
  rescue Aws::Errors::ServiceError => e
    @logger.info("Tried and failed to create demo user.")
    @logger.info("\t#{e.code}: #{e.message}")
    @logger.info("\nCan't continue the demo without a user!")
    raise
  else
    user
  end

  # Creates an access key for a user.
  #
  # @param user [Aws::IAM::User] The user that owns the key.
  # @return [Aws::IAM::AccessKeyPair] The newly created access key.
  def create_access_key_pair(user)
    user_key = @iam_client.create_access_key(user_name:
user.user_name).access_key
    @logger.info("Created accesskey pair for user #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create access keys for user #{user.user_name}.")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    user_key
  end

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
      Version: "2012-10-17",
      Statement: [{
                   Effect: "Allow",
                   Principal: {'AWS': user.arn},
                   Action: "sts:AssumeRole"
                 }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
```

```
    @logger.info("Created role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a role for the demo. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account,
and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
      Version: "2012-10-17",
      Statement: [{
                   Effect: "Allow",
                   Action: "s3:ListAllMyBuckets",
                   Resource: "arn:aws:s3:::*"
                 }]
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role
#{role.role_name}. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an inline policy for a user that lets the user assume a role.
  #
```

```
 # @param policy_name [String] The name to give the policy.
 # @param user [Aws::IAM::User] The user that owns the policy.
 # @param role [Aws::IAM::Role] The role that can be assumed.
 # @return [Aws::IAM::UserPolicy] The newly created policy.
 def create_user_policy(policy_name, user, role)
   policy_document = {
     Version: "2012-10-17",
     Statement: [{
                  Effect: "Allow",
                  Action: "sts:AssumeRole",
                  Resource: role.arn
                }]
   }.to_json
   @iam_client.put_user_policy(
     user_name: user.user_name,
     policy_name: policy_name,
     policy_document: policy_document
   )
   puts("Created an inline policy for #{user.user_name} that lets the user
assume role #{role.role_name}.")
 rescue Aws::Errors::ServiceError => e
   @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
   @logger.info("\t#{e.code}: #{e.message}")
   raise
 end

 # Creates an Amazon S3 resource with specified credentials. This is separated
into a
 # factory function so that it can be mocked for unit testing.
 #
 # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
 def create_s3_resource(credentials)
   Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
 end

 # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
 # Because the resource uses credentials with limited access, it may not be able
to
 # list the S3 buckets.
 #
 # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
 def list_buckets(s3_resource)
```

```
    count = 10
    s3_resource.buckets.each do |bucket|
      @logger.info "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
  rescue Aws::Errors::ServiceError => e
    if e.code == "AccessDenied"
      puts("Attempt to list buckets with no permissions: AccessDenied.")
    else
      @logger.info("Couldn't list buckets for the account. Here's why: ")
      @logger.info("\t#{e.code}: #{e.message}")
      raise
    end
  end

  # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
  # This is separated into a factory function so that it can be mocked for unit
testing.
  #
  # @param key_id [String] The ID of the access key used by the STS client.
  # @param key_secret [String] The secret part of the access key used by the STS
client.
  def create_sts_client(key_id, key_secret)
    Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
  end

  # Gets temporary credentials that can be used to assume a role.
  #
  # @param role_arn [String] The ARN of the role that is assumed when these
credentials
  #                          are used.
  # @param sts_client [AWS::STS::Client] An AWS STS client.
  # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
  def assume_role(role_arn, sts_client)
    credentials = Aws::AssumeRoleCredentials.new(
      client: sts_client,
      role_arn: role_arn,
      role_session_name: "create-use-assume-role-scenario"
    )
    @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
```

```
  end

  # Deletes a role. If the role has policies attached, they are detached and
  # deleted before the role is deleted.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    @iam_client.list_attached_role_policies(role_name:
 role_name).attached_policies.each do |policy|
      @iam_client.detach_role_policy(role_name: role_name, policy_arn:
 policy.policy_arn)
      @iam_client.delete_policy(policy_arn: policy.policy_arn)
      @logger.info("Detached and deleted policy #{policy.policy_name}.")
    end
    @iam_client.delete_role({ role_name: role_name })
    @logger.info("Role deleted: #{role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
 why:")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes a user. If the user has inline policies or access keys, they are
 deleted
  # before the user is deleted.
  #
  # @param user [Aws::IAM::User] The user to delete.
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id,
 user_name: user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end
```

```
# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the IAM create a user and assume a role demo!")
  puts("-" * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-
#{Random.uuid}", role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user,
 role)
  scenario.wait(10)
  puts("Try to list buckets with credentials for a user who has no permissions.")
  puts("Expect AccessDenied from this call.")
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key)))
  puts("Now, assume the role that grants permission.")
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key))
  puts("Here are your buckets:")
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts("Thanks for watching!")
  puts("-" * 88)
rescue Aws::Errors::ServiceError => e
  puts("Something went wrong with the demo.")
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
 $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey

- CreatePolicy
- CreateRole
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{config::Credentials as iamCredentials, config::Region, Client
 as iamClient};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document)
 =
        initialize_variables().await;
```

```
    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{:?}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = iamClient::new(&shared_config);
    let uuid = Uuid::new_v4().to_string();

    let list_all_buckets_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
    }"
    .to_string();
    let inline_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{}\"}]
    }"
    .to_string();

    (
        client,
        uuid,
        list_all_buckets_policy_document,
```

```
        inline_policy_document,
    )
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}{}",
 "iam_demo_user_", uuid)).await?;
    println!("Created the user with the name: {}", user.user_name());
    let key = iam_service::create_access_key(&client, user.user_name()).await?;

    let assume_role_policy_document = "{
        \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }"
    .to_string()
    .replace("{}", user.arn());

    let assume_role_role = iam_service::create_role(
        &client,
        &format!("{}{}", "iam_demo_role_", uuid),
        &assume_role_policy_document,
    )
    .await?;
    println!("Created the role with the ARN: {}", assume_role_role.arn());

    let list_all_buckets_policy = iam_service::create_policy(
        &client,
        &format!("{}{}", "iam_demo_policy_", uuid),
        &list_all_buckets_policy_document,
    )
    .await?;
    println!(
        "Created policy: {}",
        list_all_buckets_policy.policy_name.as_ref().unwrap()
    );
```

```
    let attach_role_policy_result =
        iam_service::attach_role_policy(&client, &assume_role_role,
&list_all_buckets_policy)
            .await?;
    println!(
        "Attached the policy to the role: {:?}",
        attach_role_policy_result
    );

    let inline_policy_name = format!("{}{}", "iam_demo_inline_policy_", uuid);
    let inline_policy_document = inline_policy_document.replace("{}",
assume_role_role.arn());
    iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
        .await?;
    println!("Created inline policy.");

    //First, fail to list the buckets with the user.
    let creds = iamCredentials::from_keys(key.access_key_id(),
key.secret_access_key(), None);
    let fail_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    println!("Fail config: {:?}", fail_config);
    let fail_client: s3Client = s3Client::new(&fail_config);
    match fail_client.list_buckets().send().await {
        Ok(e) => {
            println!("This should not run. {:?}", e);
        }
        Err(e) => {
            println!("Successfully failed with error: {:?}", e)
        }
    }

    let sts_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    let sts_client: stsClient = stsClient::new(&sts_config);
    sleep(Duration::from_secs(10)).await;
    let assumed_role = sts_client
        .assume_role()
```

```
        .role_arn(assume_role_role.arn())
        .role_session_name(&format!("{}{}", "iam_demo_assumerole_session_",
uuid))
        .send()
        .await;
    println!("Assumed role: {:?}", assumed_role);
    sleep(Duration::from_secs(10)).await;

    let assumed_credentials = iamCredentials::from_keys(
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .access_key_id(),
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .secret_access_key(),
        Some(
            assumed_role
                .as_ref()
                .unwrap()
                .credentials
                .as_ref()
                .unwrap()
                .session_token
                .clone(),
        ),
    );

    let succeed_config = aws_config::from_env()
        .credentials_provider(assumed_credentials)
        .load()
        .await;
    println!("succeed config: {:?}", succeed_config);
    let succeed_client: s3Client = s3Client::new(&succeed_config);
    sleep(Duration::from_secs(10)).await;
    match succeed_client.list_buckets().send().await {
        Ok(_) => {
```

```
            println!("This should now run successfully.")
        }
        Err(e) => {
            println!("This should not run. {:?}", e);
            panic!()
        }
    }

    //Clean up.
    iam_service::detach_role_policy(
        &client,
        assume_role_role.role_name(),
        list_all_buckets_policy.arn().unwrap_or_default(),
    )
    .await?;
    iam_service::delete_policy(&client, list_all_buckets_policy).await?;
    iam_service::delete_role(&client, &assume_role_role).await?;
    println!("Deleted role {}", assume_role_role.role_name());
    iam_service::delete_access_key(&client, &user, &key).await?;
    println!("Deleted key for {}", key.user_name());
    iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
    println!("Deleted inline user policy: {}", inline_policy_name);
    iam_service::delete_user(&client, &user).await?;
    println!("Deleted user {}", user.user_name());

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser

- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여

Amazon EC2 인스턴스에서 실행되는 애플리케이션에는 AWS API 요청에 AWS 보안 인증 정보가 포함되어 있어야 합니다. 개발자에게 Amazon EC2 인스턴스 내에서 직접 AWS 보안 인증 정보를 저장하고 해당 인스턴스의 애플리케이션에서 해당 보안 인증 정보를 사용하는 것을 허용하도록 했을 수 있습니다. 그러나 개발자는 그런 다음에 보안 인증을 관리해야 하며, 각 인스턴스에 보안 인증을 안전하게 전달하고 보안 인증을 업데이트해야 할 때 각 Amazon EC2 인스턴스를 업데이트해야 합니다. 이처럼 여기에는 많은 작업이 요구됩니다.

이렇게 하는 대신 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 대한 임시 보안 인증 정보를 관리할 수 있고 또 그렇게 해야 합니다. 역할을 사용할 때 Amazon EC2 인스턴스에 장기 보안 인증 정보(예: 로그인 보안 인증 정보 또는 액세스 키)를 배포하지 않아도 됩니다. 그 대신 역할은 애플리케이션에서 다른 AWS 리소스에 호출할 때 사용할 수 있는 임시 권한을 제공합니다. Amazon EC2 인스턴스를 시작할 때 인스턴스와 연결할 IAM 역할을 지정합니다. 그러면 이 인스턴스에서 실행되는 애플리케이션은 역할 제공 임시 자격 증명을 사용하여 API 요청에 서명할 수 있습니다.

역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한을 부여하기 위해서는 약간의 추가적인 구성이 필요합니다. Amazon EC2 인스턴스에서 실행되는 애플리케이션은 가상화된 운영 체제에 의해 AWS에서 추상화됩니다. 이러한 추가적인 분리로 인해 Amazon EC2 인스턴스에 AWS 역할 및 관련 권한을 할당하고 이를 그 애플리케이션도 사용 가능하게 만들려면 추가 절차가 필요합니다. 여기서 추가 절차란 인스턴스에 연결된 인스턴스 프로파일을 생성하는 것입니다. 그러면 인스턴스 프로필은 해당 역할을 포함하게 되며 인스턴스에서 실행되는 애플리케이션에 이 역할의 임시 자격 증명을 제공할 수 있습니다. 이 임시 자격 증명은 애플리케이션의 API 호출에 사용되어 리소스에 액세스하고 이 역할이 지정하는 리소스에 대해서만 액세스를 제한할 수 있습니다.

> **Note**
>
> 한 번에 하나의 역할만 Amazon EC2 인스턴스에 할당할 수 있으며, 인스턴스의 모든 애플리케이션은 동일한 역할과 권한을 공유한다는 점에 유의하세요. Amazon ECS를 활용하여 Amazon EC2 인스턴스를 관리하는 경우 실행 중인 Amazon EC2 인스턴스의 역할과 구별되는 역할을 Amazon ECS 작업에 할당할 수 있습니다. 각 작업에 역할을 할당하면 최소 권한 액세스 원칙에 부합하며, 작업과 리소스를 보다 정밀하게 제어할 수 있습니다.

자세한 내용은 Amazon Elastic Container Service 모범 사례 가이드의 Amazon ECS 작업에 IAM 역할 사용을 참조하세요.

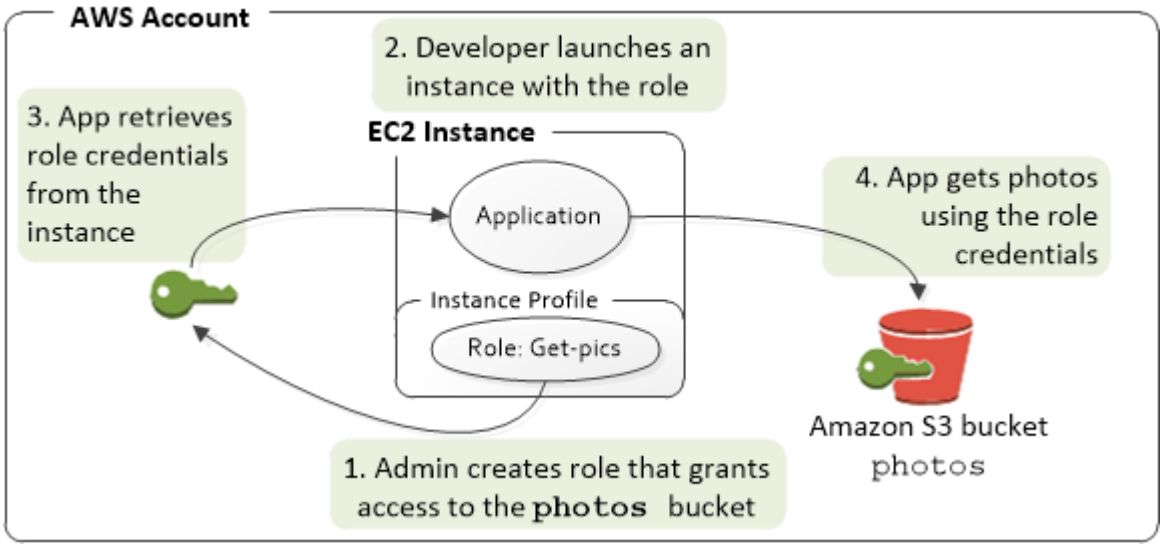
이러한 방식으로 역할을 사용하면 여러 가지 장점이 있습니다. 역할 보안 인증은 임시 정보이며 자동으로 업데이트되므로 보안 인증을 관리하지 않아도 될 뿐만 아니라 장기적인 보안 위험을 염려하지 않아도 됩니다. 또한, 여러 인스턴스에 대해 역할을 하나만 사용하는 경우 그 역할을 변경할 수 있는데, 변경 사항은 모든 인스턴스에 자동으로 전파됩니다.

> **Note**
>
> 일반적으로 역할은 Amazon EC2 인스턴스를 시작할 때 할당되지만, 현재 실행 중인 Amazon EC2 인스턴스에도 연결될 수 있습니다. 실행 중인 인스턴스에 역할을 연결하는 방법을 알아보려면 Amazon EC2의 IAM 역할 섹션을 참조하세요.

주제

Amazon EC2 인스턴스의 역할은 어떻게 작동하나요?

다음 그림에서는 개발자가 `photos`라는 S3 버킷에 대한 액세스 권한이 필요한 Amazon EC2 인스턴스에서 애플리케이션을 실행하고 있습니다. 관리자가 `Get-pics` 서비스 역할을 생성해 Amazon EC2 인스턴스에 연결합니다. 이 역할에는 지정된 S3 버킷에 대한 읽기 전용 액세스 권한을 부여하는 권한 정책이 포함되어 있습니다. 또한 Amazon EC2 인스턴스가 해당 역할을 수임하고 임시 보안 인증 정보를 가져오도록 허용하는 신뢰 정책도 포함되어 있습니다. 애플리케이션이 인스턴스에서 실행되면 역할의 임시 자격 증명을 사용하여 photos 버킷에 액세스할 수 있습니다. 관리자는 개발자 권한을 부여하지 않아도 photos 버킷에 액세스할 수 있고 개발자는 자격 증명을 공유하거나 관리할 필요가 전혀 없습니다.



1. 관리자가 IAM을 사용하여 **`Get-pics`** 역할을 생성합니다. 역할의 신뢰 정책에서 관리자는 Amazon EC2 인스턴스만이 역할을 맡을 수 있도록 지정합니다. 역할의 권한 정책에서 관리자는 `photos` 버킷에 읽기 전용 권한을 지정합니다.
2. 개발자는 Amazon EC2 인스턴스를 시작하고 이 인스턴스에 `Get-pics` 역할을 할당합니다.

> **Note**
>
> IAM 콘솔을 사용하는 경우, 인스턴스 프로파일은 콘솔에서 관리하고 대개 사용자가 파악하기 쉽습니다. 그러나 AWS CLI 또는 API를 사용하여 역할 및 Amazon EC2 인스턴스를 만들고 관리하는 경우 사용자는 인스턴스 프로파일을 만들고 별도 절차에 따라 여기에 역할을 할당해야 합니다. 그런 다음 인스턴스를 시작할 때 역할 이름이 아닌 인스턴스 프로파일 이름을 지정해야 합니다.

3. 애플리케이션이 실행되면 인스턴스 메타데이터에서 보안 자격 증명 검색에 설명된 대로 Amazon EC2 인스턴스 메타데이터에서 임시 보안 자격 증명을 가져옵니다. 이러한 자격 증명은 제한된 시간 동안에만 유효한 임시 보안 자격 증명으로 역할을 나타냅니다.

   개발자는 몇몇 AWS SDK를 사용하여 임시 보안 자격 증명을 투명하게 관리하는 공급자를 사용할 수 있습니다. (자격 증명 관리를 위해 해당 SDK가 지원하는 기능에 대한 설명은 각각의 AWS SDK 설명서를 참조하세요.)

   또는 애플리케이션이 Amazon EC2 인스턴스의 인스턴스 메타데이터에서 임시 보안 인증 정보를 가져올 수 있습니다. 자격 증명과 관련 값은 메타데이터의 `iam/security-credentials/`*`role-name`* 범주(이 경우 `iam/security-credentials/Get-pics`)에서 구할 수 있습니다. 애플리케이션이 인스턴스 메타데이터에서 자격 증명을 가져오면 자격 증명을 캐시할 수 있습니다.

4. 애플리케이션은 가져온 임시 자격 증명을 사용하여 photo 버킷에 액세스합니다. **`Get-pics`** 역할에 연결된 정책으로 인해 이 애플리케이션에는 읽기 전용 권한만 있습니다.

   인스턴스에서 사용 가능한 임시 보안 인증은 만료되기 전에 자동으로 업데이트되므로 항상 유효한 설정을 사용할 수 있습니다. 애플리케이션은 현재 자격 증명이 만료되기 전에 인스턴스 메타데이터에서 새 자격 증명을 가져와야 합니다. AWS SDK를 사용해 자격 증명을 관리하도록 함으로써 애플리케이션이 자격 증명을 새로 고치기 위해 추가적인 로직을 포함할 필요가 없게 할 수도 있습니다. 예를 들어 클라이언트를 인스턴스 프로필 자격 증명 공급자로 인스턴스화하면 됩니다. 그러나 애플리케이션이 인스턴스 메타데이터에서 임시 보안 자격 증명을 가져와 캐시한 경우, 현재 자격 증명이 만료되기 전에 한 시간 또는 최소 15분마다 갱신한 자격 증명을 가져와야 합니다. 만료 시간은 `iam/security-credentials/`*`role-name`* 카테고리에 반환되는 정보에 포함되어 있습니다.

Amazon EC2로 역할을 사용하는 데 필요한 권한

역할을 사용하여 인스턴스를 시작하려면 개발자에게 Amazon EC2 인스턴스를 시작할 수 있는 권한과 IAM 역할을 전달할 수 있는 권한이 있어야 합니다.

다음과 같은 샘플 정책은 사용자가 AWS Management Console을 사용하여 역할로 인스턴스를 시작할 수 있도록 허용합니다. 정책에 와일드카드(*)가 포함되어 있어 사용자가 어떤 역할이든 전달하고 나열된 Amazon EC2 작업을 수행할 수 있도록 허용합니다. `ListInstanceProfiles` 작업을 수행하면 사용자는 AWS 계정에서 제공되는 모든 역할을 볼 수 있습니다.

Example 사용자에게 Amazon EC2 콘솔을 사용하여 임의의 역할로 인스턴스를 시작할 권한을 부여하는 정책 예

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "IamPassRole",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ec2.amazonaws.com"
                }
            }
        },
        {
```

```
            "Sid": "ListEc2AndListInstanceProfiles",
            "Effect": "Allow",
            "Action": [
                "iam:ListInstanceProfiles",
                "ec2:Describe*",
                "ec2:Search*",
                "ec2:Get*"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon EC2 인스턴스로 전달할 수 있는 역할 제한(PassRole 사용)

PassRole 권한을 사용하여 사용자가 Amazon EC2 인스턴스를 시작할 때 이 인스턴스에 전달할 수 있는 역할을 제한할 수 있습니다. 이를 통해 사용자가 자신이 받은 권한보다 더 많은 권한이 있는 애플리케이션을 실행하지 않도록, 즉 높은 권한을 가져오지 않도록 할 수 있습니다. 예를 들어 사용자 Alice는 Amazon EC2 인스턴스를 시작하고 Amazon S3 버킷을 사용할 권한만 있지만, 그녀가 Amazon EC2 인스턴스에 전달하는 역할에는 IAM 및 Amazon DynamoDB를 사용할 권한이 있다고 가정합니다. 이 경우 Alice는 인스턴스를 시작하고 여기에 로그인하여 임시 보안 자격 증명을 가져온 다음 그녀에게 권한이 없는 IAM 또는 DynamoDB 작업을 수행할 수도 있습니다.

사용자가 Amazon EC2 인스턴스에 전달할 수 있는 역할 중 어떤 것을 제한하려면 PassRole 작업을 허용하는 정책을 생성합니다. 그런 다음 그 정책을 Amazon EC2 인스턴스를 시작할 사용자(또는 사용자가 속한 IAM 그룹)에게 연결합니다. 이 정책의 Resource 요소에서 사용자가 Amazon EC2 인스턴스에 전달할 수 있는 역할을 나열합니다. 사용자가 인스턴스를 시작하고 역할을 인스턴스에 연결하면 Amazon EC2에서 사용자가 해당 역할을 전달할 수 있는지 확인합니다. 물론 사용자가 전달할 수 있는 역할에 사용자가 보유하고 있을 것으로 추정되는 권한보다 더 많은 권한이 포함되어 있지 않은지도 확인해야 합니다.

> **Note**
>
> PassRole은 RunInstances 또는 ListInstanceProfiles와 동일한 방식의 API 작업이 아닙니다. 역할 ARN이 API에 대한 파라미터로 전달될 때마다 AWS에서 검사하는 권한입니다(또는 사용자 대신 콘솔이 이 기능을 수행). 관리자가 어느 사용자가 어느 역할을 전달할 수 있는지를 제어할 수 있습니다. 이 경우 사용자가 Amazon EC2 인스턴스에 특정 역할을 연결할 수 있습니다.

Example 사용자에게 특정 역할로 Amazon EC2 인스턴스를 시작할 권한을 부여하는 정책의 예

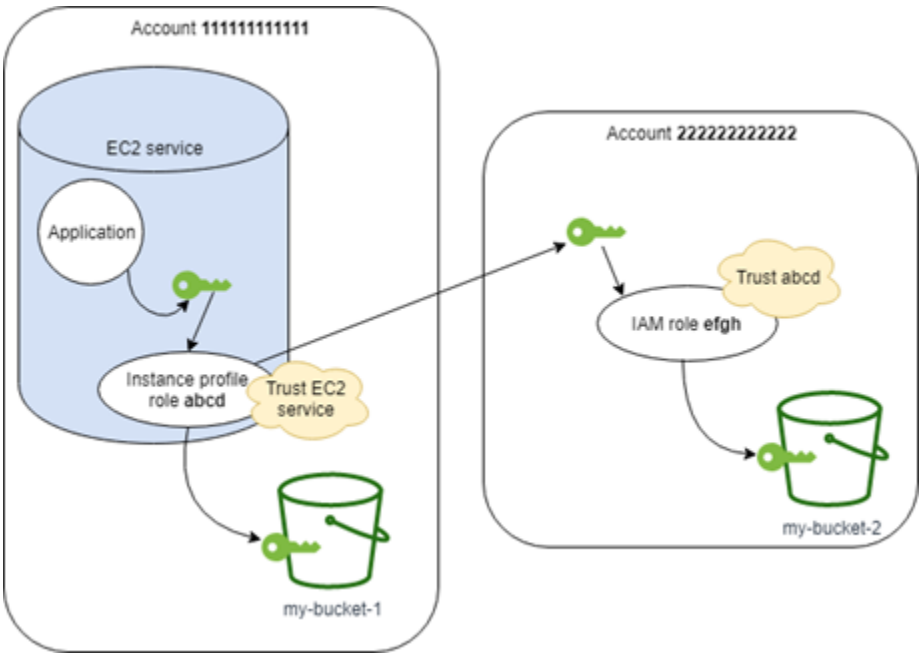
다음과 같은 샘플 정책은 사용자가 Amazon EC2 API를 사용하여 역할로 인스턴스를 시작할 수 있도록 허용합니다. `Resource` 요소는 역할의 Amazon 리소스 이름(ARN)을 지정합니다. ARN을 지정함으로써 정책은 사용자에게 `Get-pics` 역할만을 전달할 권한을 부여합니다. 사용자가 인스턴스를 시작할 때 다른 역할을 지정하려는 경우 작업이 실패합니다. 사용자는 역할을 전달하는지와 관계없이 모든 인스턴스를 실행할 권한이 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:RunInstances",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::account-id:role/Get-pics"
    }
  ]
}
```

인스턴스 프로파일 역할이 다른 계정의 역할로 전환하도록 허용

Amazon EC2 인스턴스에서 실행 중인 애플리케이션에서 다른 계정에 있는 명령을 실행하도록 허용할 수 있습니다. 이를 위해 첫 번째 계정에 있는 Amazon EC2 인스턴스 역할이 두 번째 계정의 역할로 전환하도록 허용해야 합니다.

두 개의 AWS 계정을 사용 중이고 Amazon EC2 인스턴스에서 실행 중인 특정 애플리케이션이 두 계정 모두에서 AWS CLI 명령을 실행하도록 허용하고자 한다고 가정합니다. Amazon EC2 인스턴스가 `111111111111` 계정에 존재한다고 가정합니다. 해당 인스턴스에는 애플리케이션이 동일한 `111111111111` 계정 내에 있는 `my-bucket-1` 버킷에서 읽기 전용 Amazon S3 작업을 수행하도록 허용하는 `abcd` 인스턴스 프로파일 역할이 포함되어 있습니다. 하지만 애플리케이션은 `efgh` 크로스 계정 역할을 수임하여 `222222222222` 계정의 `my-bucket-2` Amazon S3 버킷에 액세스하도록 허용되어야 합니다.



애플리케이션이 abcd Amazon S3 버킷에 액세스하도록 허용하려면 `my-bucket-1` Amazon EC2 인스턴스 프로파일 역할에 다음과 같은 권한 정책이 있어야 합니다.

계정 111111111111 ***abcd*** 역할 권한 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccountLevelS3Actions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowListAndReadS3ActionOnMyBucket",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::my-bucket-1/*",
                "arn:aws:s3:::my-bucket-1"
            ]
        },
        {
            "Sid": "AllowIPToAssumeCrossAccountRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::222222222222:role/efgh"
        }
    ]
}
```

abcd 역할은 Amazon EC2 서비스가 역할을 수임하도록 신뢰해야 합니다. 이를 위해 abcd 역할에 다음과 같은 신뢰 정책이 있어야 합니다.

계정 111111111111 ***abcd*** 역할 신뢰 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "abcdTrustPolicy",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"Service": "ec2.amazonaws.com"}
        }
    ]
}
```

efgh 크로스 계정 역할이 동일한 222222222222 계정 내에 있는 my-bucket-2 버킷에서 읽기 전용 Amazon S3 작업을 수행할 수 있다고 가정합니다. 이를 위해 efgh 크로스 계정 역할에 다음과 같은 권한 정책이 있어야 합니다.

계정 222222222222 ***efgh*** 역할 권한 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccountLevelS3Actions",
```

```
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowListAndReadS3ActionOnMyBucket",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket-2/*",
                "arn:aws:s3:::my-bucket-2"
            ]
        }
    ]
}
```

efgh 역할은 abcd 인스턴스 프로파일 역할이 이를 수임하도록 신뢰해야 합니다. 이를 위해 efgh 역할에 다음과 같은 신뢰 정책이 있어야 합니다.

계정 222222222222 ***efgh*** 역할 신뢰 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "efghTrustPolicy",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "arn:aws:iam::111111111111:role/abcd"}
        }
    ]
}
```

어떻게 시작할 수 있습니까?

역할이 Amazon EC2 인스턴스와 연동하는 방식을 이해하려면 IAM 콘솔을 사용하여 역할을 만들고 해당 역할을 사용하는 Amazon EC2 인스턴스를 시작한 다음에 실행 중인 인스턴스를 검사해야 합니다. 해당 역할의 임시 자격 증명이 인스턴스에서 사용되는 방식을 보기 위해 인스턴스 메타데이터를 검토할 수 있습니다. 또한, 인스턴스에서 실행되는 애플리케이션이 어떻게 역할을 사용하는지도 알 수 있습니다. 다음 리소스에서 자세히 알아보세요.

-
- SDK 설명입니다. AWS SDK 설명서에 Amazon S3 버킷을 읽기 위해 역할에 임시 자격 증명을 사용하는 Amazon EC2 인스턴스에서 실행되는 애플리케이션을 보여주는 자세한 안내가 포함되어 있습니다. 다음과 같은 각 설명에서는 여러 프로그래밍 언어를 사용하여 비슷한 절차를 제시합니다.
  - AWS SDK for Java 개발자 안내서의 SDK for Java를 사용하여 Amazon EC2용 IAM 역할 구성
  - AWS SDK for .NET 개발자 안내서의 SDK for .NET을 사용하여 Amazon EC2 인스턴스 시작
  - AWS SDK for Ruby 개발자 안내서의 SDK for Ruby를 사용하여 Amazon EC2 인스턴스 생성

관련 정보

역할 생성 및 Amazon EC2 인스턴스의 역할에 대한 자세한 내용은 다음 정보를 참조하세요.

- Amazon EC2 인스턴스에 IAM 역할을 사용하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서를 참조하세요.
- 역할을 만들려면 IAM 역할 생성 섹션을 참조하세요.
- 임시 보안 자격 증명의 사용에 관한 자세한 내용은 IAM의 임시 보안 자격 증명을 확인하세요.
- IAM API 또는 CLI를 사용하는 경우, IAM 인스턴스 프로파일을 생성 및 관리해야 합니다. 인스턴스 프로파일에 대한 자세한 내용은 섹션을 참조하세요인스턴스 프로파일 사용
- 인스턴스 메타데이터의 역할을 위한 임시 보안 자격 증명에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 메타데이터에서 보안 자격 증명 검색을 참조하세요.

인스턴스 프로파일 사용

인스턴스 프로파일을 사용하여 EC2 인스턴스에 IAM 역할을 전달합니다. 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2의 IAM 역할을 참조하세요.

인스턴스 프로파일 관리(콘솔)

AWS Management Console을 사용하여 Amazon EC2 역할을 생성하는 경우, 콘솔이 자동으로 인스턴스 프로파일을 생성하여 해당 역할과 동일한 이름을 부여합니다. 그런 다음 Amazon EC2 콘솔을 사용하여 IAM 역할로 인스턴스를 실행할 때는 인스턴스와 연결할 역할을 선택할 수 있습니다. 콘솔에 표시되는 목록이 실제로 인스턴스 프로파일 이름의 목록입니다. 콘솔은 Amazon EC2와 연결되지 않은 역할의 인스턴스 프로파일은 생성하지 않습니다.

역할과 인스턴스 프로파일의 이름이 같으면 AWS Management Console을 사용하여 Amazon EC2의 IAM 역할 및 인스턴스 프로파일을 삭제할 수 있습니다. 인스턴스 프로파일 삭제에 대한 자세한 내용은 역할 또는 인스턴스 프로파일 삭제 섹션을 참조하세요.

인스턴스 프로파일 관리(AWS CLI 또는 AWS API)

AWS CLI 또는 AWS API에서 역할을 관리할 경우 별도의 작업으로 역할 및 인스턴스 프로파일을 생성합니다. 역할 및 인스턴스 프로파일의 이름이 서로 다를 수 있으므로 인스턴스 프로파일 이름은 물론이고 프로파일이 속하는 역할 이름까지 알고 있어야 합니다. 그러면 EC2 인스턴스를 시작할 때 올바른 인스턴스 프로파일을 선택할 수 있습니다.

인스턴스 프로파일을 비롯한 IAM 리소스에 태그를 연결하여 해당 리소스에 대한 액세스를 식별, 구성 및 제어할 수 있습니다. AWS CLI 또는 AWS API를 사용하는 경우에만 인스턴스 프로파일을 태깅할 수 있습니다.

> **Note**
>
> 하나의 인스턴스 프로파일은 하나의 IAM 역할만 포함할 수 있습니다. 하지만 한 역할이 여러 인스턴스 프로파일에 포함될 수 있습니다. 이 인스턴스 프로파일당 역할 1개 제한은 늘릴 수 없습니다. 기존 역할을 제거하고 나서 인스턴스 프로파일에 다른 역할을 추가할 수 있습니다. 최종 일관성으로 인해 모든 AWS에 변경 사항이 적용될 때까지 기다려야 합니다. 변경을 적용하려면 인스턴스 프로파일 연결을 해제하고 나서 인스턴스 프로파일을 연결하거나, 인스턴스를 중지했다가 다시 시작합니다.

인스턴스 프로파일 관리(AWS CLI)

AWS 계정의 인스턴스 프로파일 작업을 할 때는 다음 AWS CLI 명령을 사용할 수 있습니다.

- 인스턴스 프로파일을 생성합니다: `aws iam create-instance-profile`
- 인스턴스 프로파일 태깅: `aws iam tag-instance-profile`

- 인스턴스 프로파일의 태그 나열: `aws iam list-instance-profile-tags`
- 인스턴스 프로파일 태깅 해제: `aws iam untag-instance-profile`
- 인스턴스 프로파일에 역할 추가: `aws iam add-role-to-instance-profile`
- 인스턴스 프로파일 표시: `aws iam list-instance-profiles`, `aws iam list-instance-profiles-for-role`
- 인스턴스 프로파일 정보 가져오기: `aws iam get-instance-profile`
- 인스턴스 프로파일에서 역할 제거: `aws iam remove-role-from-instance-profile`
- 인스턴스 프로파일 삭제: `aws iam delete-instance-profile`

다음 명령을 사용하여 이미 실행 중인 EC2 인스턴스에 역할을 연결할 수도 있습니다. 자세한 내용은 Amazon EC2의 IAM 역할을 참조하세요.

- 역할이 있는 인스턴스 프로파일을 중지되었거나 실행 중인 EC2 인스턴스에 연결: `aws ec2 associate-iam-instance-profile`
- EC2 인스턴스에 연결된 인스턴스 프로파일에 대한 정보 가져오기: `aws ec2 describe-iam-instance-profile-associations`
- 역할이 있는 인스턴스 프로파일을 중지되었거나 실행 중인 EC2 인스턴스에서 분리: `aws ec2 disassociate-iam-instance-profile`

인스턴스 프로파일 관리(AWS API)

AWS 계정의 인스턴스 프로파일 작업을 할 때는 다음 AWS API 연산을 호출할 수 있습니다.

- 인스턴스 프로파일을 생성합니다: `CreateInstanceProfile`
- 인스턴스 프로파일 태깅: `TagInstanceProfile`
- 인스턴스 프로필의 태그 나열: `ListInstanceProfileTags`
- 인스턴스 프로파일 태깅 해제: `UntagInstanceProfile`
- 인스턴스 프로파일에 역할 추가: `AddRoleToInstanceProfile`
- 인스턴스 프로파일 표시: `ListInstanceProfiles`, `ListInstanceProfilesForRole`
- 인스턴스 프로파일 정보 가져오기: `GetInstanceProfile`
- 인스턴스 프로파일에서 역할 제거: `RemoveRoleFromInstanceProfile`
- 인스턴스 프로파일 삭제: `DeleteInstanceProfile`

다음 연산을 호출하여 이미 실행 중인 EC2 인스턴스에 역할을 연결할 수도 있습니다. 자세한 내용은 Amazon EC2의 IAM 역할을 참조하세요.

- 역할이 있는 인스턴스 프로파일을 중지되었거나 실행 중인 EC2 인스턴스에 연결: `AssociateIamInstanceProfile`
- EC2 인스턴스에 연결된 인스턴스 프로파일에 대한 정보 가져오기: `DescribeIamInstanceProfileAssociations`
- 역할이 있는 인스턴스 프로파일을 중지되었거나 실행 중인 EC2 인스턴스에서 분리: `DisassociateIamInstanceProfile`

자격 증명 공급자 및 페더레이션

가장 좋은 방법은 AWS 계정에서 개별 IAM 사용자를 생성하는 대신 인간 사용자가 ID 제공업체와의 페더레이션을 사용하여 AWS 리소스에 액세스하는 것입니다. 자격 증명 공급자(IdP)를 사용하면 AWS 외부의 사용자 자격 증명을 관리할 수 있고 이 외부 사용자 자격 증명에 계정의 AWS 리소스에 대한 사용 권한을 부여할 수 있습니다. 기업 사용자 디렉터리처럼 조직 내에 이미 고유의 자격 증명 시스템이 있다면 이 방법이 유용합니다. 그 밖에 AWS 리소스에 액세스해야 하는 모바일 앱이나 웹 애플리케이션을 개발할 때도 효과적입니다.

> **Note**
>
> IAM에서 SAML 페더레이션을 사용하는 대신 외부 SAML ID 제공업체를 사용하여 IAM Identity Center에서 사용자 액세스를 관리할 수 있습니다. ID 제공업체와 IAM Identity Center 페더레이션을 통해 조직의 여러 AWS 계정과 여러 AWS 애플리케이션에 대한 액세스를 제공할 수 있습니다. IAM 사용자가 필요한 특정 상황에 대한 자세한 내용은 IAM 사용자(역할이 아님)를 생성해야 하는 경우를 참조하세요.

IAM Identity Center를 활성화하지 않고 단일 AWS 계정을 사용하려는 경우 OpenID Connect(OIDC) 또는 SAML 2.0(Security Assertion Markup Language 2.0)을 사용하여 ID 정보를 AWS에 제공하는 외부 IdP와 IAM을 함께 사용할 수 있습니다. OIDC는 AWS에서 실행되지 않는 GitHub Actions와 같은 애플리케이션을 AWS 리소스에 연결합니다. 잘 알려진 SAML ID 제공업체의 예로는 Shibboleth 및 Active Directory Federation Services가 있습니다.

자격 증명 공급자를 사용하면 사용자 지정 로그인 코드를 생성할 필요도, 그리고 자신의 사용자 자격 증명을 관리할 필요도 없습니다. IdP에서 이러한 작업을 대신 수행합니다. 외부 사용자는 IdP를 통해 로그인하고, 이러한 외부 자격 증명에 계정의 AWS 리소스를 사용할 권한을 제공할 수 있습니다. ID 제

공업체를 사용하면 애플리케이션에서 사용자 액세스 키 같은 장기 보안 인증을 배포하거나 포함할 필요가 없으므로 AWS 계정을 안전하게 보호할 수 있습니다.

다음 표를 검토하면 IAM, IAM Identity Center 또는 Amazon Cognito 중에서 사용 사례에 가장 적합한 IAM 페더레이션 유형이 무엇인지 파악하는 데 도움이 됩니다. 다음 요약 및 표는 사용자가 AWS 리소스에 대한 페더레이션 액세스를 얻기 위해 사용할 수 있는 방법에 대한 개요를 제공합니다.

IAM 페더레이션 유형	계정 유형	액세스 관리..	지원되는 ID 소스
IAM Identity Center와의 페더레이션	AWS Organizations에서 관리하는 여러 계정	인력의 인간 사용자	• SAML 2.0 • Managed Active Directory • Identity Center 디렉터리
IAM과의 페더레이션	단일, 독립 실행형 계정	• 단기적, 소규모 배포의 인간 사용자 • 기계 사용자	• SAML 2.0 • oidc
Amazon Cognito 자격 증명 풀과의 페더레이션	모두	리소스에 액세스하기 위해 IAM 인증이 필요한 앱 사용자	• SAML 2.0 • oidc • OAuth 2.0 소셜 ID 제공업체 선택

IAM Identity Center와의 페더레이션

인간 사용자의 중앙 액세스 관리를 위해 IAM Identity Center를 사용하여 계정에 대한 액세스 권한과 해당 계정 내 권한을 관리하는 것이 좋습니다. IAM Identity Center의 사용자에게는 AWS 리소스에 대한 단기 보안 인증이 부여됩니다. Active Directory, 외부 IdP(ID 제공업체) 또는 IAM Identity Center 디렉터리를 사용자 및 그룹의 ID 소스로 사용하여 AWS 리소스에 대한 액세스 권한을 할당할 수 있습니다.

IAM Identity Center는 SAML(Security Assertion Markup Language) 2.0과의 ID 페더레이션을 지원하여 AWS 액세스 포털 내에서 애플리케이션을 사용할 권한이 있는 사용자에게 페더레이션된 Single Sign-On 액세스를 제공합니다. 이후 사용자는 Microsoft 365, SAP Concur, Salesforce와 같은 AWS Management Console 및 타사 애플리케이션을 포함하여 SAML을 지원하는 서비스에 Single Sign-On을 할 수 있습니다.

IAM과의 페더레이션

IAM Identity Center에서 인간 사용자를 관리하는 것이 좋지만, 단기간의 소규모 배포에서는 인간 사용자를 위해 IAM과의 페더레이션 사용자 액세스를 활성화할 수 있습니다. IAM을 사용하면 별도의 SAML 2.0 및 OIDC(Open ID Connect) IdP를 사용하고 페더레이션 사용자 속성을 액세스 제어에 사용할 수 있습니다. IAM을 사용하면 IDP에서 AWS로 비용 센터, 제목 또는 로케일과 같은 사용자 속성을 전달하고, 이러한 속성을 기반으로 세분화된 액세스 권한을 구현할 수 있습니다.

워크로드란 애플리케이션이나 백엔드 프로세스같이 비즈니스 가치를 창출하는 리소스 및 코드 모음을 말합니다. 워크로드에서 AWS 서비스, 애플리케이션, 운영 도구 및 구성 요소에 요청을 보내려면 IAM ID가 필요할 수 있습니다. 이러한 자격 증명에는 AWS 환경에서 실행되는 컴퓨터가 포함됩니다(예: Amazon EC2 인스턴스 또는 AWS Lambda 함수).

또한 액세스 권한이 필요한 외부 당사자를 위해 시스템 자격 증명을 관리할 수도 있습니다. 시스템 자격 증명에 대한 액세스 권한을 부여하기 위해 IAM 역할을 사용할 수 있습니다. IAM 역할에는 특정 권한이 있으며 역할 세션과 함께 임시 보안 인증을 사용하여 AWS에 대한 액세스 방법을 제공합니다. 또한 AWS 외부에 AWS 환경에 대한 액세스 권한이 필요한 시스템이 있을 수 있습니다. AWS 외부에서 실행되는 시스템의 경우 IAM Roles Anywhere를 사용할 수 있습니다. 역할에 대한 자세한 내용은 IAM 역할 섹션을 참조하세요. 역할을 사용하여 AWS 계정 전체에서 액세스 권한을 위임하는 방법에 대한 자세한 내용은 튜토리얼: IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임을(를) 참조하세요.

IdP를 IAM과 직접 연결하려면 ID 제공업체 엔터티를 생성하여 AWS 계정과 IdP 간에 신뢰 관계를 설정합니다. IAM은 OpenID Connect(OIDC) 또는 SAML 2.0(Security Assertion Markup Language 2.0)과 호환되는 IdP를 지원합니다. AWS에서 해당 IdP 중 하나를 사용하는 것에 대한 자세한 정보는 다음을 참조하세요.

- OIDC 페더레이션
- SAML 2.0 연동

Amazon Cognito 자격 증명 풀과의 페더레이션

Amazon Cognito는 모바일 및 웹 앱에서 사용자를 인증하고 권한을 부여하고자 하는 개발자를 위해 설계되었습니다. Amazon Cognito 사용자 풀은 앱에 로그인 및 가입 기능을 추가하고, 자격 증명 풀은 AWS에서 관리하는 보호되는 리소스에 대한 액세스 권한을 사용자에게 부여하는 IAM 보안 인증 정보를 제공합니다. 자격 증명 풀은 `AssumeRoleWithWebIdentity` API 작업을 통해 임시 세션의 보안 인증 정보를 획득합니다.

Amazon Cognito는 SAML 및 OpenID Connect를 지원하는 외부 ID 제공업체, 그리고 Facebook, Google 및 Amazon과 소셜 ID 제공업체와 협력합니다. 앱은 사용자 풀 또는 외부 IdP로 사용자를 로그인한 다음, IAM 역할의 사용자 지정 임시 세션을 통해 리소스를 대신 검색할 수 있습니다.

추가 리소스

- 조직의 인증 시스템을 사용하여 AWS Management Console로 AWS Single Sign-On(SSO)을 활성화하는 사용자 지정 페더레이션 프록시를 생성하는 방법에 대한 데모를 보려면 사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기 단원을 참조하세요.

일반적인 시나리오

> **Note**
>
> 인간 사용자가 AWS에 액세스할 때 임시 보안 인증을 사용하도록 하는 것이 좋습니다. AWS IAM Identity Center 사용을 고려해 보셨나요? IAM Identity Center를 사용하여 여러 AWS 계정에 대한 액세스를 중앙에서 관리하고 사용자에게 한 곳에서 할당된 모든 계정에 대한 MFA 보호 Single Sign-On 액세스를 제공할 수 있습니다. IAM Identity Center를 사용하면 IAM Identity Center에서 사용자 자격 증명을 생성하고 관리하거나 기존 SAML 2.0 호환 자격 증명 제공업체에 쉽게 연결할 수 있습니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 IAM Identity Center란 무엇인가요? 섹션을 참조하세요.

외부 ID 제공업체(IdP)를 이용하여 AWS 외부 및 외부 IdP에서 사용자 ID를 관리할 수 있습니다. 외부 IdP는 OpenID Connect(OIDC) 또는 Security Assertion Markup Language(SAML)을 사용하여 AWS에 ID 정보를 제공할 수 있습니다. OIDC는 일반적으로 AWS에서 실행되지 않는 애플리케이션이 AWS 리소스에 액세스해야 할 때 사용됩니다.

외부 idP와의 페더레이션을 구성하려는 경우 IAM ID 제공업체를 생성하여 외부 IdP 및 구성에 대해 AWS에 알려줍니다. 이를 통해 AWS 계정과 외부 IdP 사이에 신뢰가 구축됩니다. 다음 항목에서는 IAM ID 공급자를 사용하는 일반적인 시나리오를 제공합니다.

주제

- 모바일 앱용 Amazon Cognito
- 모바일 앱용 OIDC 페더레이션

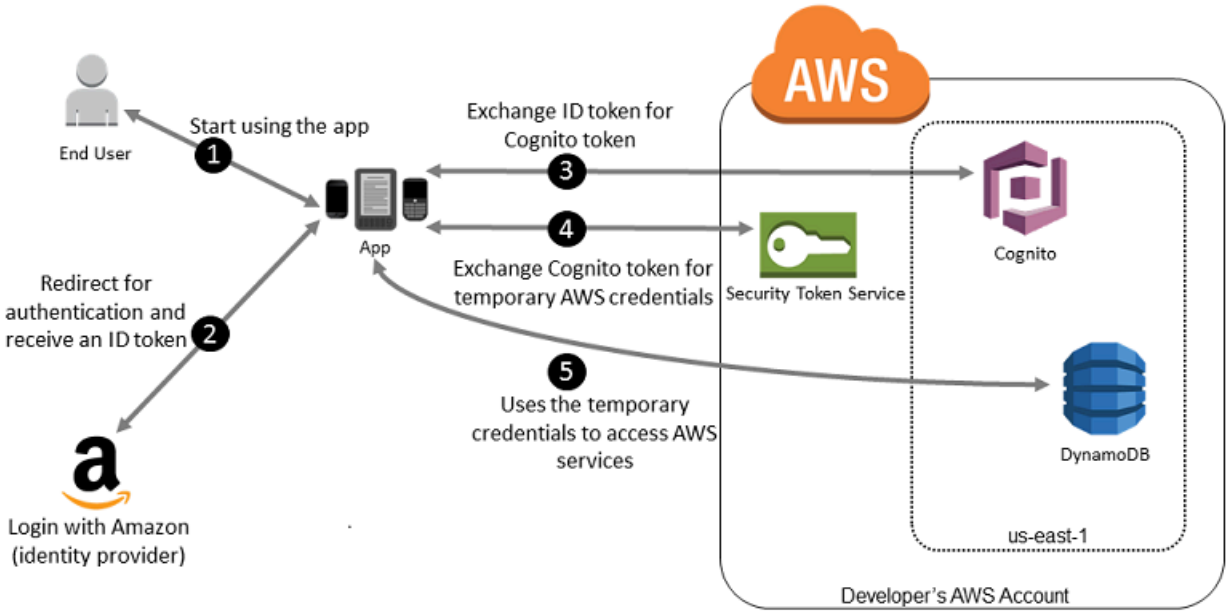
모바일 앱용 Amazon Cognito

OIDC 페더레이션을 사용하는 기본 방법은 Amazon Cognito를 사용하는 것입니다. 예를 들어 개발자 Adele이 점수, 프로필과 같은 사용자 데이터가 Amazon S3와 Amazon DynamoDB에 저장되는 모바일 디바이스를 위한 게임을 만들고 있다고 합시다. 또한 Adele은 이 데이터를 디바이스에 로컬로 저장하고 Amazon Cognito를 사용해 여러 디바이스 간에 동기화된 상태로 유지할 수 있습니다. Adele은 보안 및 유지 보수 상의 이유로 장기 AWS 보안 자격 증명은 게임과 함께 배포되어서는 안 된다는 것을 알고 있습니다. 또한, 게임 사용자가 아주 많을 수도 있다는 것을 알고 있습니다. 이 모든 이유로 Adele은 각 플레이어에 대해 IAM에서 새로운 사용자 자격 증명을 생성하길 원하지 않습니다. 대신에 사용자가 Login with Amazon, Facebook, Google 또는 OpenID Connect(OIDC) 호환 자격 증명 공급자(IdP)와 같은 널리 알려진 외부 IdP를 통해 이미 설정한 자격 증명을 사용해 로그인할 수 있도록 게임을 구축합니다. Adele의 게임은 이러한 공급자 중 하나의 인증 메커니즘을 이용해 사용자의 자격 증명을 확인할 수 있습니다.

모바일 앱이 자신의 AWS 리소스에 액세스할 수 있도록 하기 위해 Adele은 먼저 선택한 IdP에 개발자 ID를 등록합니다. Adele은 이들 각 공급자로 애플리케이션을 구성하기도 합니다. Adele은 게임용 Amazon S3 버킷 및 DynamoDB 테이블을 포함하는 AWS 계정에서 Amazon Cognito를 사용해 게임에 필요한 권한을 정확하게 정의하는 IAM 역할을 생성합니다. Adele이 OIDC IdP를 사용하는 경우 IAM OIDC ID 제공자 엔터티도 생성하여 AWS 계정의 Amazon Cognito 아이덴티티 풀과 IdP 간에 신뢰를 구축합니다.

앱의 코드에서 Adele은 자신이 이전에 구성한 IdP에 대한 로그인 인터페이스를 호출합니다. IdP는 사용자가 로그인하도록 허용하는 모든 세부 정보를 처리하고 앱은 공급자에게서 OAuth 액세스 토큰 또는 OIDC ID 토큰을 얻습니다. Adele의 앱은 이 인증 정보를 주고 AWS 액세스 키 ID, 보안 액세스 키 및 세션 토큰으로 구성된 임시 보안 자격 증명 집합을 얻을 수 있습니다. 그러면 앱은 이러한 자격 증명을 사용하여 AWS가 제공하는 웹 서비스에 액세스할 수 있습니다. 앱은 수임하는 역할에 정의된 권한으로 제한됩니다.

다음 그림은 Login with Amazon을 IdP로 사용하는 경우 이것이 어떻게 작동하는지 그 흐름을 단순화해 보여줍니다. 2단계에서 앱은 Facebook, Google 또는 OIDC 호환 IdP를 사용할 수도 있지만, 여기에서는 생략했습니다.



1. 고객은 모바일 디바이스에서 앱을 시작합니다. 앱은 사용자에게 로그인하도록 요청합니다.
2. 앱은 Login with Amazon 리소스를 사용해 사용자의 자격 증명을 수락합니다.
3. 앱이 Amazon Cognito API 작업 `GetId` 및 `GetCredentialsForIdentity`를 사용하여 Login with Amazon ID 토큰을 Amazon Cognito 토큰으로 교환합니다. Login with Amazon 프로젝트를 신뢰하도록 구성된 Amazon Cognito는 AWS STS와의 임시 세션 자격 증명과 교환하는 토큰을 생성합니다.
4. 앱은 Amazon Cognito에서 임시 보안 자격 증명을 받습니다. 또한 앱은 Amazon Cognito의 기본(클래식) 워크플로를 사용하여 `AssumeRoleWithWebIdentity`로 AWS STS에서 토큰을 검색할 수 있습니다. 자세한 내용은 Amazon Cognito 개발자 안내서의 자격 증명 풀(페더레이션 아이덴티티) 인증 흐름을 참조하세요.
5. 임시 보안 자격 증명은 앱에 의해 사용됨으로써 앱이 작동을 요청하는 어떤 AWS 리소스에도 액세스할 수 있습니다. 임시 보안 자격 증명과 연결된 역할 및 할당된 정책에 따라 액세스할 수 있는 항목이 결정됩니다.

다음 절차에 따라 Amazon Cognito를 사용해 사용자를 인증하도록 앱을 구성하고 앱에 AWS 리소스에 대한 액세스 권한을 부여합니다. 이 시나리오를 완수하기 위한 특정 단계에 대해서는 Amazon Cognito 설명서를 참조하세요.

1. (선택 사항) Login with Amazon, Facebook, Google 또는 기타 OpenID Connect(OIDC) 호환 IdP에 개발자로 가입하여 해당 공급자를 통해 1개 이상의 앱을 구성합니다. Amazon Cognito에서는 사용자의 인증되지 않은(게스트) 액세스도 지원하므로 이 단계는 선택 사항입니다.

2. AWS Management Console에서 Amazon Cognito로 이동합니다. Amazon Cognito 마법사를 사용해 자격 증명 풀을 생성합니다. 이 풀은 Amazon Cognito가 앱을 위해 최종 사용자 자격 증명을 정돈된 상태로 유지할 목적으로 사용하는 컨테이너입니다. 앱 간에 자격 증명 풀을 공유할 수 있습니다. 자격 증명 풀을 설정할 때 Amazon Cognito는 사용자에 대한 권한을 정의하는 1~2개의 IAM 역할(인증된 자격 증명용 한 개, 인증되지 않은 "게스트" 자격 증명용 한 개)을 생성합니다.
3. AWSAmplify를 앱과 통합하고 Amazon Cognito를 사용하는 데 필요한 파일을 가져옵니다.
4. 자격 증명 풀 ID, AWS 계정 번호 및 자격 증명 풀과 연결한 역할의 Amazon 리소스 이름(ARN)을 전달하여 Amazon Cognito 보안 인증 공급자의 인스턴스를 생성합니다. AWS Management Console의 Amazon Cognito 마법사에서 제공하는 샘플 코드를 사용하면 시작하는 데 도움이 됩니다.
5. 앱이 AWS 리소스에 액세스할 때 클라이언트 객체에 자격 증명 공급자 인스턴스를 전달합니다. 이렇게 하면 클라이언트에 임시 보안 자격 증명이 전달됩니다. 자격 증명에 대한 권한은 앞서 정의한 역할 또는 역할들에 기반을 두고 있습니다.

자세한 내용은 다음 자료를 참조하십시오.

- AWS Amplify Framework Documentation의 로그인(Android).
- AWS Amplify Framework Documentation(프레임워크 설명서)의 Sign in (iOS)(로그인(iOS))

모바일 앱용 OIDC 페더레이션

최상의 결과를 얻으려면 거의 모든 OIDC 페더레이션 시나리오에서 Amazon Cognito를 ID 브로커로 사용하는 것이 좋습니다. Amazon Cognito는 사용하기 쉽고 익명(인증되지 않은) 액세스, 디바이스 및 공급자 간에 사용자 데이터 동기화 등의 추가 기능을 제공합니다. 그러나 이미 수동으로 `AssumeRoleWithWebIdentity` API를 호출하여 OIDC 페더레이션을 사용하는 앱을 생성한 경우에는 계속 사용할 수 있으며 앱이 계속 정상적으로 작동합니다.

Amazon Cognito를 사용하지 않고 OIDC 페더레이션을 사용하는 프로세스의 경우 다음과 같은 일반적인 개요를 따릅니다

1. 외부 자격 증명 공급자(IdP)에서 개발자로 로그인하여 앱을 위한 고유 ID를 부여하는 IdP에서 앱을 구성합니다. (IdP마다 이 프로세스에 대해 다른 용어를 사용합니다. 이 개요에서는 IdP에 앱을 식별하는 프로세스를 구성이라는 용어로 설명합니다.) 각 IdP는 IdP 고유의 앱 ID를 제공함으로써, 동일한 앱을 다수의 IdP로 구성하는 경우 앱은 여러 개의 앱 ID를 갖게 됩니다. 각 공급자로 여러 개의 앱을 구성할 수 있습니다.

   다음 외부 링크는 흔히 사용되는 자격 증명 공급자(IdP) 중 일부를 사용하는 것에 대한 정보를 제공합니다.

- Login with Amazon 개발자 센터
- Facebook 개발자 사이트의 앱 또는 웹 사이트에 Facebook 로그인 추가하기
- Google 개발자 사이트의 OAuth 2.0을 사용한 로그인(OpenID Connect)

> **Important**
>
> Google, Facebook 또는 Amazon Cognito의 OIDC 자격 증명 공급자를 사용하는 경우 AWS Management Console에서 IAM 자격 증명 공급자를 별도로 생성하지 마세요. AWS에 내장된 OIDC 자격 증명 공급자를 사용하면 됩니다. 다음 단계를 건너뛰고 자격 증명 공급자를 사용하여 새 역할을 생성하는 단계로 바로 이동합니다.

2. Google, Facebook 또는 Amazon Cognito 외에 OIDC와 호환되는 다른 IdP를 사용하는 경우 IdP를 위한 IAM 자격 증명 공급자 엔터티를 생성합니다.
3. IAM에서 하나 이상의 역할을 생성합니다. 각 역할에 대해 그 역할을 위임할 수 있는 대상(신뢰 정책)과 앱 사용자가 가지는 권한(권한 정책)을 정의합니다. 일반적으로 앱이 지원하는 각 IdP마다 하나의 역할을 생성합니다. 예를 들면 사용자가 Login with Amazon을 통해 로그인할 때 앱이 수임할 수 있는 역할, 사용자가 Facebook을 통해 로그인하는 경우 동일한 앱의 두 번째 역할, 사용자가 Google을 통해 로그인하는 경우 앱의 세 번째 역할 등을 생성할 수 있습니다. 신뢰 관계를 위해서는 IdP(예: Amazon.com)를 `Principal`(신뢰받는 개체)로 지정하고 앱 ID에 할당된 IdP와 일치하는 `Condition`을 포함시키세요. 여러 공급자의 역할에 대한 예는 타사 ID 제공업체의 역할 생성(페더레이션)에 설명되어 있습니다.
4. 애플리케이션에서 IdP로 사용자를 인증하세요. 이렇게 하는 방법의 구체적인 내용은 사용 중인 IdP(Login with Amazon, Facebook 또는 Google)와 앱이 실행되는 플랫폼에 따라 달라집니다. 예를 들어 Android 앱의 인증 방법은 iOS 앱 또는 JavaScript 기반 웹 앱과 다를 수 있습니다.

   일반적으로 사용자가 아직 로그인하지 않은 경우 IdP가 로그인 페이지 표시를 처리합니다. IdP가 사용자를 인증한 후에 IdP는 사용자에 대한 정보가 담긴 인증 토큰을 앱에 반환합니다. 포함된 정보의 내용은 IdP가 노출하는 것과 사용자가 공유하고자 하는 정보가 무엇인지에 달려 있습니다. 앱에서 이 정보를 사용할 수 있습니다.
5. 앱에서 `AssumeRoleWithWebIdentity` 작업에 대한 서명되지 않은 호출을 수행하여 임시 보안 자격 증명을 요청합니다. 요청 시 IdP의 인증 토큰을 전달하고 해당 IdP에 대해 생성한 IAM 역할의 Amazon 리소스 이름(ARN)을 지정합니다. AWS는 그 토큰이 신뢰할 수 있고 유효한지 확인하여, 그럴 경우에는 요청에서 명명하는 역할에 대한 권한을 포함하는 임시 보안 자격 증명을 앱에 반환합니다. 그 응답에는 IdP가 사용자에게 연결하는 고유 사용자 ID와 같은, IdP에서 오는 사용자에 대한 메타데이터도 포함되어 있습니다.

6. `AssumeRoleWithWebIdentity` 응답의 임시 보안 자격 증명을 사용하여 앱에서 AWS API 작업에 대한 서명된 요청을 생성합니다. IdP의 사용자 ID 정보는 앱에서 사용자를 구분할 수 있습니다. 예를 들어 사용자 ID가 접두사 또는 접미사로 포함된 Amazon S3 폴더에 객체를 넣을 수 있습니다. 이렇게 함으로써 폴더를 잠그는 액세스 제어 정책을 생성해 그 ID를 지닌 사용자만 그 폴더에 액세스할 수 있게 됩니다. 자세한 내용은 AWS STS 페더레이션 사용자 세션 보안 주체 단원을 참조하십시오.
7. 앱은 AWS에 요청할 필요가 있을 때마다 새 임시 보안 자격 증명을 받지 않아도 되도록 임시 보안 자격 증명을 캐시해야 합니다. 기본적으로 자격 증명은 1시간 동안 유효합니다. 자격 증명이 만료되면 (또는 그 전에) `AssumeRoleWithWebIdentity`에 또 한 번 호출을 하여 새로운 임시 보안 자격 증명 집합을 얻으세요. IdP의 토큰 역시 보통 설정된 시간이 지나면 만료되기 때문에, IdP 및 IdP가 토큰을 어떻게 관리하느냐에 따라 `AssumeRoleWithWebIdentity`에 새로운 호출을 하기 전에 IdP의 토큰을 갱신해야 할 수도 있습니다. AWS SDK for iOS 또는 AWS SDK for Android를 사용하는 경우 AmazonSTSCredentialsProvider 작업을 사용해 IAM 임시 자격 증명을 필요에 따라 갱신하는 등 관리할 수 있습니다.

OIDC 페더레이션

워크플로를 사용하여 Amazon S3 및 DynamoDB에 액세스하는 GitHub Actions와 같은 AWS 리소스에 액세스하는 애플리케이션을 구축하는 상황을 가정해 보겠습니다.

이러한 워크플로를 사용할 때는 AWS 액세스 키로 서명해야 하는 AWS 서비스에 요청을 하게 됩니다. 하지만 AWS 자격 증명을 AWS 외부의 애플리케이션에 장기적으로 저장하지 않는 것을 강력하게 권장합니다. 대신 OIDC 페더레이션을 사용하여 필요할 때 임시 AWS 보안 자격 증명을 동적으로 요청하도록 애플리케이션을 구성합니다. 제공되는 임시 자격 증명은 애플리케이션에 필요한 작업을 수행하는 데 필요한 권한만 있는 AWS 역할에 매핑됩니다.

OIDC 페더레이션을 사용하면 사용자 정의 로그인 코드를 생성하거나 자신의 사용자 보안 인증을 관리할 필요가 없습니다. 대신, GitHub Actions 또는 기타 OIDC(OpenID Connect)와 호환되는 IdP와 같은 애플리케이션에서 OIDC를 사용하여 AWS에 인증할 수 있습니다. 이러한 애플리케이션은 JWT(JSON 웹 토큰)라는 인증 토큰을 수신한 후 이 토큰을 AWS에서 임시 보안 자격 증명으로 교환하여 AWS 계정의 특정 리소스를 사용할 수 있는 권한이 있는 IAM 역할에 매핑합니다. IdP를 사용하면 애플리케이션에 장기 보안 자격 증명을 포함하거나 배포할 필요가 없으므로 AWS 계정을(를) 안전하게 유지할 수 있습니다.

대부분의 시나리오에서 Amazon Cognito를 사용하는 것이 좋은데, Amazon Cognito가 자격 증명 브로커의 역할을 하고 페더레이션 작업을 대부분 수행하기 때문입니다. 자세한 정보는 모바일 앱용 Amazon Cognito 섹션을 참조하세요.

> **Note**
>
> OpenID Connect(OIDC) ID 제공업체에서 발행하는 JWT(JSON 웹 토큰)에는 토큰이 만료되는 시기를 지정하는 exp 클레임에 만료 시간이 포함되어 있습니다. IAM은 OpenID Connect(OIDC) Core 1.0 표준에서 허용하는 대로 클럭 스큐를 고려하여 JWT에 지정된 만료 시간 이후 5분의 기간을 제공합니다. 즉, 만료 시간 이후 5분 이내에 IAM이 수신한 OIDC JWT가 추가 평가 및 처리를 위해 승인됩니다.

주제

OIDC 페더레이션 관련 추가 리소스

다음 리소스는 OIDC 페더레이션에 대해 자세히 알아보는 데 도움이 됩니다.

- Amazon Web Services에서 OpenID Connect를 구성하여 GitHub 워크플로 내에서 OpenID Connect 사용
- Android용 Amplify 라이브러리 가이드의Amazon Cognito 아이덴티티와 Swift용 Amplify 라이브러리 가이드의 Amazon Cognito 아이덴티티를 참조하십시오.
- AWS Partner Network(APN) 블로그의 Microsoft Entra ID를 사용한 OpenID Connect 기반 AWS IAM 웹 ID 역할 자동화에서는 머신 간 OIDC 인증을 사용하여 자동화된 백그라운드 프로세스 또는 AWS 외부에서 실행되는 애플리케이션을 인증하는 방법을 안내합니다.
- 모바일 애플리케이션을 사용한 OIDC 페더레이션 문서에서는 웹 ID 페더레이션에 대해 설명하며 OIDC 페더레이션을 사용하여 Amazon S3의 콘텐츠에 액세스하는 방법의 예를 보여줍니다.

IAM에서 OIDC(OpenID Connect) ID 공급자 생성

IAM OIDC 자격 증명 공급자는 IAM의 엔터티로서 Google이나 Salesforce와 같은 OpenID Connect(OIDC) 표준을 지원하는 외부 자격 증명 공급자(IdP) 서비스를 설명합니다. IAM OIDC 자격 증명 공급자는 OIDC 호환 IdP와 AWS 계정 간에 신뢰를 구축하려 할 때 사용합니다. 예를 들어 AWS 리소스에 액세스하는 데 필요한 모바일 앱이나 웹 애플리케이션을 개발하면서 사용자 지정 로그인 코드를 생성하거나 자신의 사용자 자격 증명을 관리하지 않을 때 유용합니다. 이 시나리오에 대한 자세한 내용은 the section called “OIDC 페더레이션”를 참조하세요.

AWS Management Console, AWS Command Line Interface, Tools for Windows PowerShell 또는 IAM API 호출을 사용하여 IAM OIDC 자격 증명 공급자를 생성하고 관리할 수 있습니다.

IAM OIDC 자격 증명 공급자를 생성한 후 1개 이상의 IAM 역할을 생성해야 합니다. 역할은 AWS의 자격 증명으로서 자신만의 고유한 자격 증명이 없지만(사용자가 그러하듯이), 이 콘텍스트에서는 조직의 IdP에 의해 인증된 페더레이션 사용자에게 동적으로 할당됩니다. 그 역할은 조직의 IdP가 AWS에 액세스하기 위해 임시 보안 자격 증명을 요청할 수 있도록 허용합니다. 역할에 할당된 정책은 페더레이션 사용자가 AWS에서 하도록 허용된 것이 무엇인지 결정합니다. 서드 파티 자격 증명 공급자에 대한 역할을 생성하려면 타사 ID 제공업체의 역할 생성(페더레이션) 섹션을 참조하세요.

> **Important**
>
> `oidc-provider` 리소스를 지원하는 작업에 대해 ID 기반 정책을 구성하는 경우. IAM은 지정된 경로를 포함한 전체 OIDC ID 공급자 URL을 평가합니다. OIDC ID 공급자 URL에 경로가 있는 경우 그 경로를 `oidc-provider`ARN에 `Resource` 요소 값으로 포함해야 합니다. URL 도메인에 슬래시와 와일드카드(`/*`)를 추가할 수도 있고 URL 경로의 어느 지점에서든 와일드카드 문자(`*` 및?)를 사용할 수도 있습니다. 요청의 OIDC ID 공급자 URL이 정책의 `Resource` 요소에 설정된 값과 일치하지 않는 경우 요청은 실패합니다.

IAM OIDC 페더레이션과 관련된 일반적인 문제를 해결하려면 AWS re:Post의 OIDC 관련 오류 해결을 참조하세요.

주제

- 필수 조건: 자격 증명 공급자의 구성 검증
- OIDC 공급자 생성 및 관리(콘솔)
- IAM OIDC 자격 증명 공급자 생성 및 관리(AWS CLI)
- OIDC 자격 증명 공급자 생성 및 관리(AWS API)

필수 조건: 자격 증명 공급자의 구성 검증

IAM OIDC 자격 증명 공급자를 생성하려면 IdP에서 다음 정보를 가져와야 합니다. OIDC 공급자 구성 정보를 얻는 방법에 대한 자세한 정보는 해당 IdP 설명서를 참조하세요.

1. 공개적으로 사용 가능한 OIDC 자격 증명 공급자 URL을 확인하세요. URL은 https://로 시작해야 합니다. OIDC 표준에 따라 경로 구성 요소는 허용되지만 쿼리 파라미터는 허용되지 않습니다. 일반적으로 URL은 https://server.example.org 또는 https://example.com 같은 하나의 호스트 이름으로만 구성됩니다. URL은 포트 번호를 포함하지 않아야 합니다.

2. OIDC 자격 증명 공급자의 URL 끝에 /.well-known/openid-configuration을 추가하면 공개적으로 사용 가능한 공급자 구성 문서와 메타데이터를 확인할 수 있습니다. OpenID Connect 공급자 검색 엔드포인트 URL에서 검색할 수 있는 공급자의 구성 문서 및 메타데이터가 포함된 JSON 형식의 검색 문서가 있어야 합니다.
3. 공급자의 구성 정보에 다음 값이 포함되어 있는지 확인하세요. openid-configuration에 다음 필드 중 하나라도 없는 경우 검색 문서를 업데이트해야 합니다. 이 프로세스는 자격 증명 공급자에 따라 다를 수 있으므로 IdP의 설명서에 따라 이 작업을 완료하세요.

   - issuer: 도메인의 URL입니다.
   - jwks_uri: IAM이 퍼블릭 키를 가져오는 JSON Web Key Set(JWKS) 엔드포인트입니다. 자격 증명 공급자는 openid-configuration에 JSON Web Key Set(JWKS) 엔드포인트를 포함해야 합니다. 이 URI는 자격 증명 공급자의 서명된 토큰을 확인하는 데 사용되는 퍼블릭 키를 가져올 위치를 정의합니다.
   - claim s_supported: AWS가 IAM 정책에서 페더레이션 사용자의 권한 검사에 사용하는 필수 속성이 IdP의 OIDC 인증 응답에 포함되어 있는지 확인하도록 도와주는 사용자 관련 정보입니다. 클레임에 사용할 수 있는 IAM 조건 키 목록은 AWS OIDC 페더레이션에서 사용 가능한 키를 참조하세요.
     - aud: IdP가 JSON Web Token(JWT)으로 발급하는 audience 클레임 값을 확인해야 합니다. audience(aud) 클레임은 애플리케이션별로 다르며 토큰의 의도된 수신자를 식별합니다. 모바일 또는 웹 앱을 OpenID Connect 공급자에 등록하면 애플리케이션을 식별하는 클라이언트 ID가 설정됩니다. 클라이언트 ID는 인증을 위해 aud 클레임에 전달되는 앱의 고유 식별자입니다. IAM OIDC 자격 증명 공급자를 생성할 때 aud 클레임이 Audience 값과 일치해야 합니다.
     - iat: 클레임에는 ID 토큰이 발급된 시간을 나타내는 `iat` 값이 포함되어야 합니다.
     - iss: 자격 증명 공급자의 URL입니다. URL은 https://로 시작해야 하며 IAM에 제공된 공급자 URL과 일치해야 합니다. OIDC 표준에 따라 경로 구성 요소는 허용되지만 쿼리 파라미터는 허용되지 않습니다. 일반적으로 URL은 https://server.example.org 또는 https://example.com 같은 하나의 호스트 이름으로만 구성됩니다. URL은 포트 번호를 포함하지 않아야 합니다.
   - response_types_supported: id_token
   - subject_types_supported: public
   - id_token_signing_alg_values_supported: RS256

   > **Note**
   >
   > 아래 예제에서 custom과 같은 추가 클레임을 포함할 수 있지만 AWS STS는 이 클레임을 무시합니다.

```
{
  "issuer": "https://example-domain.com",
  "jwks_uri": "https://example-domain.com/jwks/keys",
  "claims_supported": [
    "aud",
    "iat",
    "iss",
    "name",
    "sub",
    "custom"
  ],
  "response_types_supported": [
    "id_token"
  ],
  "id_token_signing_alg_values_supported": [
    "RS256"
  ],
  "subject_types_supported": [
    "public"
  ]
}
```

OIDC 공급자 생성 및 관리(콘솔)

다음 지침에 따라 AWS Management Console에서 IAM OIDC 자격 증명 공급자를 생성 및 관리합니다.

> **Important**
>
> Google, Facebook 또는 Amazon Cognito의 OIDC 자격 증명 공급자를 사용하는 경우 이 절차에 따라 IAM 자격 증명 공급자를 별도로 생성하지 마세요. 이러한 OIDC 자격 증명 공급자는 이미 AWS에 내장되어 있고 용도에 맞게 사용할 수 있습니다. 대신 단계에 따라 자격 증명 공급자의 새 역할을 생성합니다. 자세한 내용은 OpenID Connect 페더레이션을 위한 역할 생성(콘솔) 섹션을 참조하세요.

IAM OIDC 자격 증명 공급자를 생성하려면(콘솔)

1. IAM OIDC 자격 증명 공급자를 생성하려면 먼저 애플리케이션을 IdP에 등록하여 클라이언트 ID를 받아야 합니다. 클라이언트 ID(사용자라고도 불림)는 앱을 IdP에 등록할 때 발급되는 고유의 앱 식

별자입니다. 클라이언트 ID를 얻는 방법에 대한 자세한 정보는 해당 IdP에 대한 설명서를 참조하세요.

> **Note**
>
> AWS는 신뢰할 수 있는 루트 CA(인증 기관) 라이브러리를 사용해 OIDC ID 공급자(IdP)와의 통신을 보호하여 JWKS(JSON 웹 키 세트) 엔드포인트의 TLS 인증서를 확인합니다. OIDC IdP가 이러한 신뢰할 수 있는 CA 중 하나에서 서명하지 않은 인증서를 사용하는 경우에만 IdP의 구성에 설정된 지문을 사용하여 통신을 보호합니다. AWS는 TLS 인증서를 검색할 수 없거나 TLS v1.3이 필요한 경우 지문 확인으로 대체합니다.

2. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
3. 탐색 창에서 [자격 증명 공급자(Identity providers)]를 선택한 다음 [공급자 추가(Add provider)]를 선택합니다.
4. [공급자 구성(Configure provider)]에 대해 [OpenID Connect]를 선택합니다.
5. Provider URL(공급자 URL)에서 IdP의 URL을 입력합니다. URL은 다음과 같은 제한을 준수해야 합니다.

   - URL은 대/소문자를 구분합니다.
   - URL은 **`https://`**로 시작해야 합니다.
   - URL은 포트 번호를 포함하지 않아야 합니다.
   - AWS 계정 내에서 각 IAM OIDC 자격 증명 공급자는 고유한 URL을 사용해야 합니다. AWS 계정에서 OpenID Connect 공급자에 이미 사용된 URL을 제출하려고 하면 오류가 발생합니다.

6. 대상(Audience) 필드에 IdP를 등록하고 Step 1에서 받은 애플리케이션의 클라이언트 ID를 입력하면 AWS에게도 요청됩니다. IdP에 등록한 클라이언트 ID(사용자들이라고도 불림)가 더 있는 경우 나중에 공급자 세부 정보 페이지에서 추가할 수 있습니다.

> **Note**
>
> IdP JWT 토큰에 `azp` 클레임이 포함된 경우 이 값을 Audience 값으로 입력하세요. OIDC ID 제공업체는 토큰에 `aud` 및 `azp` 클레임을 모두 설정하고, AWS STS에서는 `azp` 클레임의 값을 `aud` 클레임으로 사용합니다.

7. (선택 사항) 태그 추가(Add tags)에 키 값 페어를 추가하여 IdP를 식별하고 구성할 수 있습니다. 태그를 사용하여 AWS 리소스에 대한 액세스를 제어할 수도 있습니다. IAM OIDC 자격 증명 공급자

태깅에 대한 자세한 내용은 OpenID Connect(OIDC) ID 제공업체 태깅 섹션을 참조하세요. 태그 추가를 선택합니다. 각 태그 키 값 페어의 값을 입력합니다.

8. 제공한 정보를 확인합니다. 완료되면 [공급자 추가(Add provider)]를 선택합니다. IAM에서는 OIDC IdP 서버 인증서의 최상위 중간 CA 지문을 검색하고 사용하여 IAM OIDC ID 제공업체를 생성하려고 시도합니다.

> **Note**
>
> OIDC ID 제공업체의 인증서 체인은 도메인 또는 발급자 URL로 시작하고, 그다음으로 중간 인증서, 마지막으로는 루트 인증서여야 합니다. 인증서 체인 순서가 다른 경우 또는 중복되거나 추가 인증서가 포함된 경우 서명 불일치 오류가 발생하고 STS는 JSON 웹 토큰(JWT) 유효성 검사에 실패합니다. 서버에서 반환된 체인의 인증서 순서를 수정하여 오류를 해결하세요. 인증서 체인 표준에 대한 자세한 내용은 RFC 시리즈 웹 사이트에서 RFC 5246의 certificate_list를 참조하세요.

9. 자격 증명 공급자에 IAM 역할을 할당하여 자격 증명 공급자가 관리하는 외부 사용자 자격 증명에 계정의 AWS 리소스에 대한 액세스 권한을 부여합니다. ID 연동을 위한 역할을 생성하는 방법에 대한 자세한 내용은 타사 ID 제공업체의 역할 생성(페더레이션) 섹션을 참조하세요.

> **Note**
>
> 역할 신뢰 정책에 사용되는 OIDC IdP는 이를 신뢰하는 역할과 동일한 계정에 있어야 합니다.

IAM OIDC 자격 증명 공급자에 대한 지문을 추가하거나 제거하려면(콘솔)

> **Note**
>
> AWS는 신뢰할 수 있는 루트 CA(인증 기관) 라이브러리를 사용해 OIDC ID 공급자(IdP)와의 통신을 보호하여 JWKS(JSON 웹 키 세트) 엔드포인트의 TLS 인증서를 확인합니다. OIDC IdP가 이러한 신뢰할 수 있는 CA 중 하나에서 서명하지 않은 인증서를 사용하는 경우에만 IdP의 구성에 설정된 지문을 사용하여 통신을 보호합니다. AWS는 TLS 인증서를 검색할 수 없거나 TLS v1.3이 필요한 경우 지문 확인으로 대체합니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

2. 탐색 창에서 [자격 증명 공급자(Identity providers)]를 선택합니다. 그런 다음 업데이트할 IAM 자격 증명 공급자의 이름을 선택합니다.
3. 엔드포인트 확인 탭을 선택한 다음 지문 섹션에서 관리를 선택합니다. 새 지문 값을 입력하려면 [지문 추가(Add thumbprint)]를 선택합니다. 지문을 제거하려면 삭제할 지문 옆에 있는 [제거(Remove)]를 선택합니다.

> **Note**
>
> IAM OIDC 자격 증명 공급자마다 한 개 이상의 지문이 있어야 하며 최대 5개까지 가능합니다.

작업을 마쳤으면 [변경 사항 저장(Save changes)]을 선택합니다.

IAM OIDC 자격 증명 공급자에 대한 대상을 추가하려면(콘솔)

1. 탐색 창에서 자격 증명 공급자(Identity providers)를 선택한 다음 업데이트할 IAM 자격 증명 공급자의 이름을 선택합니다.
2. [대상(Audiences)] 섹션에서 [작업(Actions)]을 선택하고 [대상 추가(Add audience)]를 선택합니다.
3. IdP에 등록하고 Step 1에서 받은 애플리케이션의 클라이언트 ID를 입력하면 AWS로 요청이 전송됩니다. 그런 다음 [대상 추가(Add audience)]를 선택합니다.

> **Note**
>
> IAM OIDC 자격 증명 공급자마다 한 명 이상의 사용자가 있어야 하며 최대 100명까지 가능합니다.

IAM OIDC 자격 증명 공급자에 대한 대상을 제거하려면(콘솔)

1. 탐색 창에서 자격 증명 공급자(Identity providers)를 선택한 다음 업데이트할 IAM 자격 증명 공급자의 이름을 선택합니다.
2. [대상(Audiences)] 섹션에서 제거할 대상 옆에 있는 라디오 단추를 선택한 다음 [작업(Actions)]을 선택합니다.
3. [대상 제거(Remove audience)]를 선택합니다. 새 창이 열립니다.

4. 대상을 제거하면 해당 대상과 연동된 자격 증명에서 해당 대상에 연결된 역할을 수임할 수 없습니다. 창에서 경고를 읽고 필드에 `remove`라는 단어를 입력하여 대상 제거를 확인합니다.
5. [제거(Remove)]를 선택하여 대상을 제거합니다.

IAM OIDC 자격 증명 공급자를 삭제하려면(콘솔)

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 [자격 증명 공급자(Identity providers)]를 선택합니다.
3. 삭제할 IAM 자격 증명 공급자 옆의 확인란을 선택합니다. 새 창이 열립니다.
4. 필드에 단어 `delete`를 입력하여 공급자 삭제를 확인합니다. 그런 다음 삭제를 선택합니다.

IAM OIDC 자격 증명 공급자 생성 및 관리(AWS CLI)

다음 AWS CLI 명령을 사용하여 IAM OIDC 자격 증명 공급자를 생성하고 관리할 수 있습니다.

IAM OIDC 자격 증명 공급자를 생성하려면(AWS CLI)

1. (선택 사항) AWS 계정의 전체 IAM OIDC 자격 증명 공급자 목록을 가져오려면 다음 명령을 실행합니다.
   - `aws iam list-open-id-connect-providers`
2. 새 IAM OIDC 자격 증명 공급자를 생성하려면 다음 명령을 실행합니다.
   - `aws iam create-open-id-connect-provider`

기존 IAM OIDC 자격 증명 공급자의 서버 인증서 지문 목록을 업데이트하려면(AWS CLI)

- IAM OIDC 자격 증명 공급자의 서버 인증서 지문 목록을 업데이트하려면 다음 명령을 실행합니다.
  - `aws iam update-open-id-connect-provider-thumbprint`

기존 IAM OIDC 자격 증명 공급자를 태깅하려면(AWS CLI)

- 기존 IAM OIDC 자격 증명 공급자를 태깅하려면 다음 명령을 실행합니다.
  - `aws iam tag-open-id-connect-provider`

기존 IAM OIDC 자격 증명 공급자의 태그를 나열하려면(AWS CLI)

- 기존 IAM OIDC 자격 증명 공급자의 태그를 나열하려면 다음 명령을 실행합니다.
  - `aws iam list-open-id-connect-provider-tags`

IAM OIDC 자격 증명 공급자에서 태그를 제거하려면(AWS CLI)

- 기존 IAM OIDC 자격 증명 공급자에서 태그를 제거하려면 다음 명령을 실행합니다.
  - `aws iam untag-open-id-connect-provider`

기존 IAM OIDC 자격 증명 공급자에서 클라이언트 ID를 추가하거나 제거하려면(AWS CLI)

1. (선택 사항) AWS 계정의 전체 IAM OIDC 자격 증명 공급자 목록을 가져오려면 다음 명령을 실행합니다.
   - `aws iam list-open-id-connect-providers`
2. (선택 사항) IAM OIDC 자격 증명 공급자에 대한 자세한 정보를 보려면 다음 명령을 실행합니다.
   - `aws iam get-open-id-connect-provider`
3. 기존 IAM OIDC 자격 증명 공급자에 새로운 클라이언트 ID를 추가하려면 다음 명령을 실행합니다.
   - `aws iam add-client-id-to-open-id-connect-provider`
4. 기존 IAM OIDC 자격 증명 공급자에서 클라이언트를 제거하려면 다음 명령을 실행합니다.
   - `aws iam remove-client-id-from-open-id-connect-provider`

IAM OIDC 자격 증명 공급자를 삭제하려면(AWS CLI)

1. (선택 사항) AWS 계정의 전체 IAM OIDC 자격 증명 공급자 목록을 가져오려면 다음 명령을 실행합니다.
   - `aws iam list-open-id-connect-providers`
2. (선택 사항) IAM OIDC 자격 증명 공급자에 대한 자세한 정보를 보려면 다음 명령을 실행합니다.
   - `aws iam get-open-id-connect-provider`
3. IAM OIDC 자격 증명 공급자를 삭제하려면 다음 명령을 실행합니다.

- aws iam delete-open-id-connect-provider

OIDC 자격 증명 공급자 생성 및 관리(AWS API)

다음 IAM API 명령을 사용하여 OIDC 공급자를 생성하고 관리할 수 있습니다.

IAM OIDC 자격 증명 공급자를 생성하려면(AWS API)

1. (선택 사항) AWS 계정의 전체 IAM OIDC 자격 증명 공급자 목록을 가져오려면 다음 작업을 호출합니다.
   - ListOpenIDConnectProviders
2. 새로운 IAM OIDC 자격 증명 공급자를 생성하려면 다음 작업을 호출합니다.
   - CreateOpenIDConnectProvider

기존 IAM OIDC 자격 증명 공급자의 서버 인증서 지문 목록을 업데이트하려면(AWS API)

- IAM OIDC 자격 증명 공급자의 서버 인증서 지문 목록을 업데이트하려면 다음 작업을 호출합니다.
  - UpdateOpenIDConnectProviderThumbprint

기존 IAM OIDC 자격 증명 공급자를 태깅하려면(AWS API)

- 기존 IAM OIDC 자격 증명 공급자를 태깅하려면 다음 작업을 호출합니다.
  - TagOpenIDConnectProvider

기존 IAM OIDC 자격 증명 공급자의 태그를 나열하려면(AWS API)

- 기존 IAM OIDC 자격 증명 공급자의 태그를 나열하려면 다음 작업을 호출합니다.
  - ListOpenIDConnectProviderTags

기존 IAM OIDC 자격 증명 공급자에서 태그를 제거하려면(AWS API)

- 기존 IAM OIDC 자격 증명 공급자에서 태그를 제거하려면 다음 작업을 호출합니다.

- UntagOpenIDConnectProvider

기존 IAM OIDC 자격 증명 공급자에서 클라이언트 ID를 추가하거나 제거하려면(AWS API)

1. (선택 사항) AWS 계정의 전체 IAM OIDC 자격 증명 공급자 목록을 가져오려면 다음 작업을 호출합니다.
   - ListOpenIDConnectProviders
2. (선택 사항) IAM OIDC 자격 증명 공급자에 대한 자세한 정보를 보려면 다음 작업을 호출합니다.
   - GetOpenIDConnectProvider
3. 기존 IAM OIDC 자격 증명 공급자에 새로운 클라이언트 ID를 추가하려면 다음 작업을 호출합니다.
   - AddClientIDToOpenIDConnectProvider
4. 기존 IAM OIDC 자격 증명 공급자에서 클라이언트 ID를 제거하려면 다음 작업을 호출합니다.
   - RemoveClientIDFromOpenIDConnectProvider

IAM OIDC 자격 증명 공급자를 삭제하려면(AWS API)

1. (선택 사항) AWS 계정의 전체 IAM OIDC 자격 증명 공급자 목록을 가져오려면 다음 작업을 호출합니다.
   - ListOpenIDConnectProviders
2. (선택 사항) IAM OIDC 자격 증명 공급자에 대한 자세한 정보를 보려면 다음 작업을 호출합니다.
   - GetOpenIDConnectProvider
3. IAM OIDC 자격 증명 공급자를 삭제하려면 다음 작업을 호출합니다.
   - DeleteOpenIDConnectProvider

OpenID Connect ID 공급자에 대한 지문 얻기

IAM에서 OpenID Connect(OIDC) ID 제공업체를 생성할 때는 외부 ID 제공업체(idP)에서 사용한 인증서에 서명한 중간 인증 기관(CA)의 지문이 IAM에 필요합니다. 지문은 OIDC 호환 IdP에 대한 인증서 발급에 사용되는 CA에 대한 서명입니다. IAM OIDC 자격 증명 공급자를 생성하는 경우 해당 IdP에 의해 인증된 자격 증명이 자신의 AWS 계정에 액세스할 수 있도록 신뢰하는 것입니다. CA의 인증서 지문을

사용함으로써 등록된 것과 DNS 이름이 동일한 CA에서 발급한 모든 인증서를 신뢰하게 됩니다. 이를 통해 IdP의 서명 인증서를 갱신할 때 각 계정의 신뢰를 업데이트할 필요가 없습니다.

Important

대부분의 경우 연동 서버는 두 가지 다른 인증서를 사용합니다.

- 첫 번째는 AWS 및 IdP 사이의 HTTPS 연결을 설정합니다. 이는 잘 알려진 퍼블릭 루트 CA(예: AWS Certificate Manager)에서 발급해야 합니다. 이렇게 하면 클라이언트가 인증서의 안정성과 상태를 확인할 수 있습니다.
- 두 번째는 토큰을 암호화하는 데 사용되며 프라이빗 또는 퍼블릭 루트 CA가 서명해야 합니다.

AWS Command Line Interface, Tools for Windows PowerShell 또는 IAM API를 사용하여 IAM OIDC 자격 증명 공급자를 생성할 수 있습니다. 이러한 방법을 사용하면 수동으로 지문을 제공하는 옵션이 생깁니다. 지문을 포함하지 않기로 선택하면 IAM은 OIDC IdP 서버 인증서의 최상위 임시 CA 지문을 검색합니다. 지문을 포함하도록 선택한 경우 지문을 수동으로 획득하여 AWS에 제공해야 합니다.

IAM 콘솔을 사용하여 OIDC ID 제공업체를 생성하면 IAM에서는 자동으로 OIDC IdP 서버 인증서의 최상위 중간 CA 지문 검색을 시도합니다.

또한, OIDC IdP의 지문을 수동으로 가져와서 IAM에서 올바른 지문을 검색했는지 확인하는 것이 좋습니다. 인증서 지문을 얻는 방법에 대한 자세한 내용은 다음 섹션을 참조하세요.

Note

AWS는 신뢰할 수 있는 루트 CA(인증 기관) 라이브러리를 사용해 OIDC ID 공급자(IdP)와의 통신을 보호하여 JWKS(JSON 웹 키 세트) 엔드포인트의 TLS 인증서를 확인합니다. OIDC IdP가 이러한 신뢰할 수 있는 CA 중 하나에서 서명하지 않은 인증서를 사용하는 경우에만 IdP의 구성에 설정된 지문을 사용하여 통신을 보호합니다. AWS는 TLS 인증서를 검색할 수 없거나 TLS v1.3이 필요한 경우 지문 확인으로 대체합니다.

인증서 지문 얻기

웹 브라우저와 OpenSSL 명령줄 도구를 사용하여 OIDC 공급자의 인증서 지문을 얻습니다. 하지만 수동으로 인증서 지문을 얻어서 IAM OIDC ID 공급자를 생성할 필요는 없습니다. 다음 절차를 통해 OIDC 공급자의 인증서 지문을 얻을 수 있습니다.

OIDC IdP의 지문을 얻으려면

1. OIDC IdP의 지문을 얻으려면, 먼저 OpenSSL 명령줄 도구를 얻어야 합니다. 이 도구를 사용하여 OIDC IdP 인증서 체인을 다운로드하고 인증서 체인에 있는 마지막 인증서의 지문을 생성합니다. OpenSSL을 설치 및 구성해야 하는 경우 OpenSSL 설치 및 OpenSSL 구성의 지침을 따르세요.
2. OIDC IdP URL(예: `https://server.example.com`)로 시작한 다음 `/.well-known/openid-configuration`을 추가하여 다음과 같이 OIDC IdP의 구성 문서에 대한 URL을 만듭니다.

   `https://server.example.com/.well-known/openid-configuration`

   웹 브라우저에서 이 URL을 열 때 *`server.example.com`*을 IdP 서버 이름으로 바꾸어 엽니다.
3. 표시되는 문서에서 웹 브라우저 찾기 기능을 사용하여 `"jwks_uri"` 텍스트를 찾습니다. `"jwks_uri"` 텍스트 바로 뒤에 콜론(:)과 URL이 보입니다. 그 URL의 정규화된 도메인 이름을 복사합니다. `https://` 또는 최상위 도메인 다음에 오는 경로는 포함하지 마세요.

   ```
   {
    "issuer": "https://accounts.example.com",
    "authorization_endpoint": "https://accounts.example.com/o/oauth2/v2/auth",
    "device_authorization_endpoint": "https://oauth2.exampleapis.com/device/code",
    "token_endpoint": "https://oauth2.exampleapis.com/token",
    "userinfo_endpoint": "https://openidconnect.exampleapis.com/v1/userinfo",
    "revocation_endpoint": "https://oauth2.exampleapis.com/revoke",
    "jwks_uri": "https://www.exampleapis.com/oauth2/v3/certs",
   ...
   ```
4. OpenSSL 명령줄 도구를 사용하여 다음 명령을 실행합니다. 이때 *`keys.example.com`*을 Step 3에서 얻은 도메인 이름으로 바꿉니다.

   ```
   openssl s_client -servername keys.example.com -showcerts -
   connect keys.example.com:443
   ```
5. 명령 창에서 다음 예제와 비슷한 인증서가 보일 때까지 위로 스크롤합니다. 인증서가 두 개 이상 있을 경우 명령 출력의 하단에서 표시된 마지막 인증서를 찾습니다. 여기에는 인증 기관 체인의 최상위 중간 CA 인증서가 포함됩니다.

   ```
   -----BEGIN CERTIFICATE-----
    MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
    VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
   ```

```
b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----
```

인증서를 복사해(`-----BEGIN CERTIFICATE-----` 및 `-----END CERTIFICATE-----` 줄 포함) 텍스트 파일에 붙여 넣습니다. 그 다음에 그 파일을 **`certificate.crt`**라는 이름으로 저장합니다.

> **Note**
>
> OIDC ID 제공업체의 인증서 체인은 도메인 또는 발급자 URL로 시작하고, 그다음으로 중간 인증서, 마지막으로는 루트 인증서여야 합니다. 인증서 체인 순서가 다른 경우 또는 중복되거나 추가 인증서가 포함된 경우 서명 불일치 오류가 발생하고 STS는 JSON 웹 토큰(JWT) 유효성 검사에 실패합니다. 서버에서 반환된 체인의 인증서 순서를 수정하여 오류를 해결하세요. 인증서 체인 표준에 대한 자세한 내용은 RFC 시리즈 웹 사이트에서 RFC 5246의 certificate_list를 참조하세요.

6. OpenSSL 명령줄 도구를 사용하여 다음 명령을 실행합니다.

```
openssl x509 -in certificate.crt -fingerprint -sha1 -noout
```

다음 예제와 비슷한 인증서 지문이 명령 창에 표시됩니다.

```
SHA1 Fingerprint=99:0F:41:93:97:2F:2B:EC:F1:2D:DE:DA:52:37:F9:C9:52:F2:0D:9E
```

이 문자열에서 콜론 문자(:)를 제거하여 다음과 같은 최종 지문을 생성합니다.

```
990F4193972F2BECF12DDEDA5237F9C952F20D9E
```

7. AWS CLI, Tools for Windows PowerShell 또는 IAM API를 사용하여 IAM OIDC ID 제공업체를 생성하는 경우에는 지문 제공이 선택 사항입니다. 생성하는 동안 지문을 포함하지 않기로 선택하면 IAM에서는 OIDC IdP 서버 인증서의 최상위 임시 CA 지문을 검색합니다. IAM OIDC ID 제공업체가 생성되면 이 지문을 IAM에서 검색한 지문과 비교할 수 있습니다.

   IAM 콘솔에서 IAM OIDC ID 제공업체를 생성하면 콘솔에서는 자동으로 OIDC IdP 서버 인증서의 최상위 중간 CA 지문 검색을 시도합니다. 이 지문을 IAM에서 검색한 지문과 비교할 수 있습니다. IAM OIDC ID 제공업체가 생성되면 OIDC 제공업체 요약 콘솔 페이지의 엔드포인트 확인 탭에서 IAM OIDC ID 제공업체의 지문을 볼 수 있습니다.

   > **Important**
   >
   > 가져온 지문이 IAM OIDC ID 제공업체 지문 세부 정보에 표시되는 지문과 일치하지 않으면 OIDC 제공업체를 사용하지 않아야 합니다. 그 대신에, 생성된 OIDC 제공업체를 삭제한 다음에 시간이 조금 지난 후 OIDC 제공업체 생성을 다시 시도해야 합니다. 제공업체를 사용하기 전에 지문이 일치하는지 확인하세요. 두 번째 시도 후에도 지문이 여전히 일치하지 않을 경우에는 IAM 포럼을 통해 AWS에 문의하세요.

OpenSSL 설치

아직 OpenSSL을 설치하지 않았다면 이 단원에 나오는 지침을 따르세요.

Linux 또는 Unix에서 OpenSSL을 설치하려면

1. OpenSSL: Source, Tarballs(https://openssl.org/source/)로 이동합니다.
2. 최신 소스를 다운로드하여 패키지를 생성합니다.

Windows에서 OpenSSL을 설치하려면

1. Windows 버전을 설치할 수 있는 사이트 목록을 보려면 OpenSSL: Binary Distributions(https://wiki.openssl.org/index.php/Binaries)로 이동합니다.
2. 선택한 사이트의 지침을 따라 설치를 시작합니다.
3. Microsoft Visual C++ 2008 재배포 가능 패키지 설치를 묻는 메시지가 표시되고 아직 시스템에 설치되지 않았다면 환경에 적합한 다운로드 링크를 선택합니다. Microsoft Visual C++ 2008 재배포 가능 패키지 설치 마법사의 지시를 따릅니다.

> **Note**
>
> 시스템에 Microsoft Visual C++ 2008 Redistributables가 설치되어 있는지 알 수 없는 경우 OpenSSL을 먼저 설치합니다. Microsoft Visual C++ 2008 Redistributables가 설치되지 않은 경우에는 OpenSSL 설치 관리자에 알림이 표시됩니다. 설치할 OpenSSL 버전에 해당하는 아키텍처(32비트 또는 64비트)를 설치해야 합니다.

4. Microsoft Visual C++ 2008 Redistributables를 설치한 후에는 환경에 맞는 OpenSSL 바이너리를 선택하고 파일을 로컬 위치에 저장합니다. OpenSSL 설치 마법사를 시작합니다.
5. OpenSSL 설치 마법사의 지시에 따릅니다.

OpenSSL 구성

OpenSSL 명령을 사용하려면 OpenSSL이 설치된 위치 정보가 담기도록 운영 체제를 구성해야 합니다.

Linux 또는 Unix에서 OpenSSL을 구성하려면

1. 명령줄에서 `OpenSSL_HOME` 변수를 OpenSSL 설치 위치로 설정합니다.

```
$ export OpenSSL_HOME=path_to_your_OpenSSL_installation
```

2. OpenSSL 설치가 포함되도록 경로를 설정합니다.

```
$ export PATH=$PATH:$OpenSSL_HOME/bin
```

> **Note**
>
> `export` 명령을 사용하여 변경한 환경 변수는 현재 세션에만 유효합니다. 쉘 구성 파일에서 설정하면 환경 변수의 영구 변경이 가능합니다. 자세한 내용은 운영 체제 설명서를 참조하세요.

Windows에서 OpenSSL을 구성하려면

1. 명령 프롬프트 창을 엽니다.
2. `OpenSSL_HOME` 변수를 OpenSSL 설치 위치로 설정합니다.

```
C:\> set OpenSSL_HOME=path_to_your_OpenSSL_installation
```

3. OpenSSL_CONF 변수를 OpenSSL 설치에 있는 구성 파일 위치로 설정합니다.

```
C:\> set OpenSSL_CONF=path_to_your_OpenSSL_installation\bin\openssl.cfg
```

4. OpenSSL 설치가 포함되도록 경로를 설정합니다.

```
C:\> set Path=%Path%;%OpenSSL_HOME%\bin
```

> **Note**
>
> 명령 프롬프트 창에서 변경한 Windows 환경 변수는 현재 명령줄 세션에만 유효합니다. 환경 변수를 시스템 속성으로 설정하면 환경 변수의 영구 변경이 가능합니다. 정확한 절차는 사용 중인 Windows 버전에 따라 달라집니다. 예를 들어 Windows 7에서는 제어판, 시스템 및 보안, 시스템을 연 다음 고급 시스템 설정, 고급 탭, 환경 변수(Environment Variables)를 선택합니다. 자세한 내용은 Windows 설명서를 참조하세요.

SAML 2.0 연동

AWS는 많은 자격 증명 공급자(IdP)가 사용하는 개방형 표준인 SAML 2.0(Security Assertion Markup Language 2.0)이라는 아이덴티티 페더레이션을 지원합니다. 이 기능은 페더레이션 Single Sign-On(SSO)을 활성화하므로 조직의 모든 구성원에 대해 IAM 사용자를 생성하지 않아도 사용자가 AWS Management Console에 로그인하거나 AWS API 작업을 호출할 수 있습니다. SAML을 사용함으로써 AWS로 연동을 구성하는 과정을 단순화할 수 있는데, 이는 사용자 지정 자격 증명 프록시 코드를 작성하는 대신 IdP의 서비스를 사용할 수 있기 때문입니다.

IAM 페더레이션은 다음과 같은 사용 사례를 지원합니다.

- 조직의 사용자 또는 애플리케이션이 AWS API 작업을 호출할 수 있도록 허용하는 페더레이션 액세스. 이 사용 사례는 다음 섹션에서 설명합니다. 조직에서 생성되는 SAML 어설션(인증 응답의 일부)을 사용해 임시 보안 자격 증명을 얻습니다. 이 시나리오는 임시 보안 자격 증명 요청 및 OIDC 페더레이션에 기술된 것과 같이 IAM이 지원하는 다른 페더레이션 시나리오와 유사합니다. 그러나 조직의 SAML 2.0 기반 IdP가 인증 수행 및 권한 부여 확인을 위한 세부 사항을 런타임에 대부분 처리합니다.

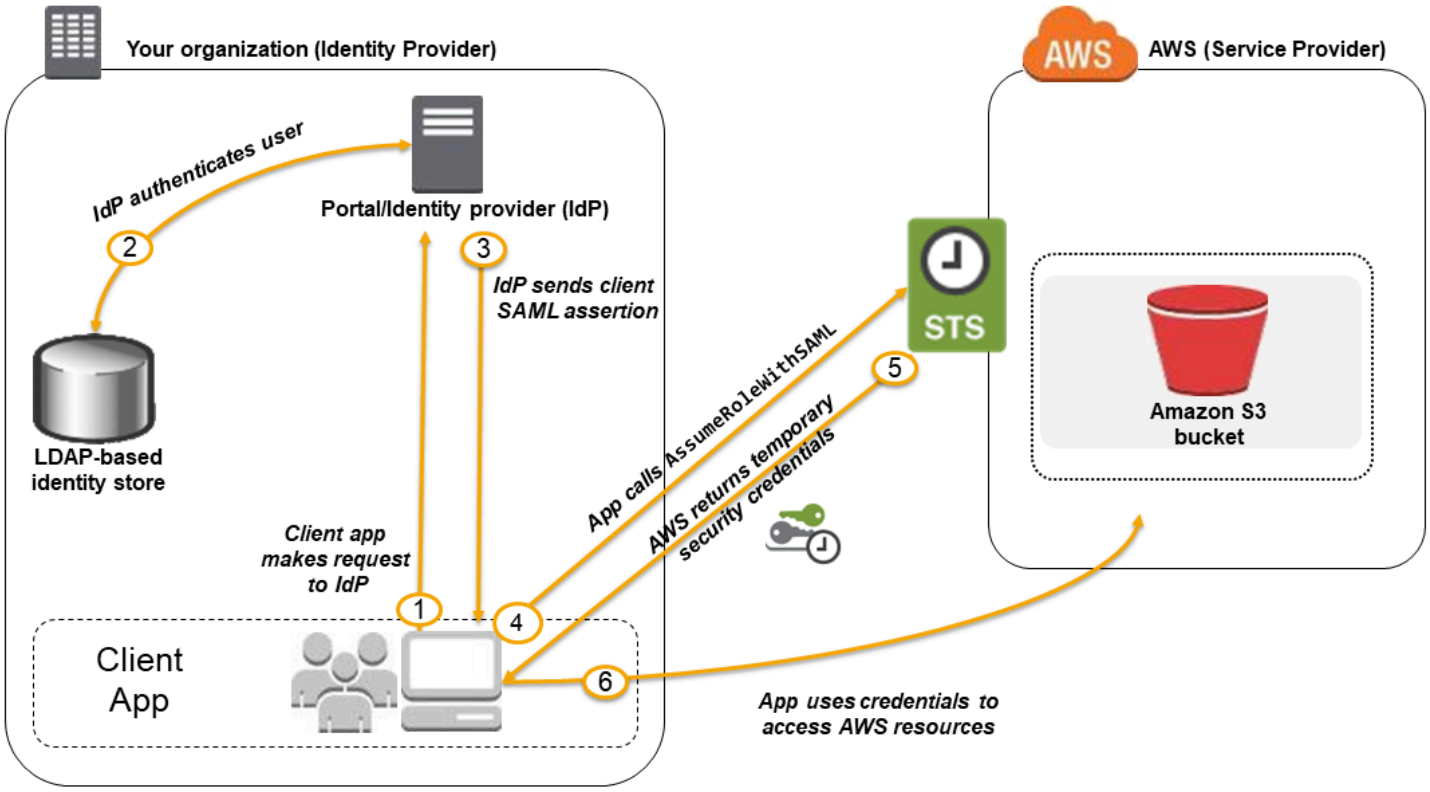
- 조직에서 AWS Management Console로 이루어지는 웹 기반 SSO(Single Sign-On). 사용자가 SAML 2.0 호환 IdP에서 호스팅하는 조직의 포털에 로그인하고 AWS로 이동하는 옵션을 선택하면 추가 로그인 정보를 제공하지 않고도 콘솔에 리디렉션될 수 있습니다. 타사 SAML IdP를 사용하여 콘솔에 SSO 액세스하거나 사용자 지정 IdP를 만들어 외부 사용자의 콘솔 액세스를 허용할 수 있습니다. 사용자 지정 IdP를 구축하는 방법에 대한 자세한 정보는 사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기를 참조하세요.

주제

- AWS에 대한 API 액세스를 위해 SAML 기반 페더레이션 사용
- SAML 2.0 기반 페더레이션 구성 개요
- AWS 리소스에 대한 SAML 페더레이션 액세스를 허용하는 역할 개요
- SAML 기반 페더레이션에서 사용자를 고유하게 식별
- IAM에서 SAML ID 공급자 생성
- 신뢰 당사자 신뢰 및 클레임 추가를 통해 SAML 2.0 IdP 구성
- 서드 파티 SAML 솔루션 공급자를 AWS와 통합
- 인증 응답에 대한 SAML 어설션 구성
- SAML 2.0 페더레이션 사용자가 AWS Management Console에 액세스할 수 있게 하기
- 브라우저에서 SAML 응답 보기

AWS에 대한 API 액세스를 위해 SAML 기반 페더레이션 사용

직원들에게 자신의 컴퓨터에서 백업 폴더로 데이터를 복사하는 방법을 제공하려 한다고 가정해 봅시다. 사용자가 컴퓨터에서 실행하는 애플리케이션을 구축합니다. 이 애플리케이션은 백엔드에서 Amazon S3 버킷에 있는 객체를 읽고 씁니다. 사용자는 AWS에 직접 액세스할 수 없습니다. 그 대신 다음 프로세스를 사용합니다.



1. 조직 내 사용자가 클라이언트 앱을 사용해 조직의 IdP로부터 인증을 요청합니다.
2. IdP가 조직의 자격 증명 스토어를 이용하여 사용자를 인증합니다.
3. IdP가 사용자에 대한 정보로 SAML 어설션을 만들어 클라이언트 앱으로 보냅니다.
4. 클라이언트 앱이 AWS STS`AssumeRoleWithSAML` API를 호출하면서 SAML 공급자의 ARN, 수임할 역할의 ARN, IdP로부터 받은 SAML 어설션을 전달합니다.
5. 클라이언트 앱에 대한 API 응답에는 임시 보안 자격 증명이 포함되어 있습니다.
6. 클라이언트 앱이 임시 보안 자격 증명을 사용하여 Amazon S3 API 작업을 호출합니다.

SAML 2.0 기반 페더레이션 구성 개요

앞의 시나리오와 다이어그램을 통해 설명한 대로 SAML 2.0 기반 페더레이션을 사용하기 전에, 서로를 신뢰하도록 조직의 IdP와 AWS 계정을 구성해야 합니다. 이 신뢰를 구성하는 일반적인 프로세스는 다음 단계에서 설명합니다. 조직 내에는 Microsoft Active Directory 연동 서비스(AD FS, Windows Server의 일부), Shibboleth 또는 기타 호환 가능한 SAML 2.0 공급자와 같이 SAML 2.0을 지원하는 IdP가 반드시 있어야 합니다.

> **Note**
>
> 페더레이션 복원력을 개선하려면 여러 SAML 로그인 엔드포인트를 지원하도록 IdP 및 AWS 페더레이션을 구성하는 것이 좋습니다. 자세한 내용은 AWS 보안 블로그 문서 How to use regional SAML endpoints for failover(장애 조치에 리전 SAML 엔드포인트를 사용하는 방법)를 참조하세요.

조직의 IdP와 AWS가 서로 신뢰하도록 구성

1. AWS를 조직의 IdP에 서비스 제공업체(SP)로 등록합니다. `https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` 의 SAML 메타데이터 문서를 사용합니다.

   가능한 *region-code* 값 목록은 AWS 로그인 엔드포인트의 리전(Region) 열을 참조하세요.

   필요에 따라 `https://signin.aws.amazon.com/static/saml-metadata.xml` 에서 SAML 메타데이터 문서를 사용할 수 있습니다.
2. 조직의 IdP를 사용하여 AWS에서 IdP를 IAM 자격 증명 공급자로 기술하는 동등한 메타데이터 XML 파일을 생성합니다. 그 파일에는 발급자 이름, 생성 일자, 만료 일자 및 AWS가 조직에서 오는 인증 응답의 유효성을 검증하는 데 사용할 수 있는 키가 포함되어 있어야 합니다.
3. IAM 콘솔에서 SAML 자격 증명 공급자를 생성합니다. 이 과정에서 SAML 메타데이터 문서와 조직의 IdP가 Step 2에서 생성한 를 업로드합니다. 자세한 내용은 IAM에서 SAML ID 공급자 생성 단원을 참조하십시오.
4. IAM에서 하나 이상의 IAM 역할을 생성합니다. 역할의 신뢰 정책에서 SAML 공급자를 보안 주체로 설정함으로써 조직과 AWS 사이에 신뢰 관계를 설정합니다. 역할의 권한 정책은 조직의 사용자가 AWS에서 하도록 허용된 것을 설정합니다. 자세한 내용은 타사 ID 제공업체의 역할 생성(페더레이션) 단원을 참조하십시오.

   > **Note**
   >
   > 역할 신뢰 정책에 사용되는 SAML IDP는 해당 역할이 속한 계정과 동일한 계정에 있어야 합니다.
5. 조직의 IdP에서 조직의 사용자 또는 그룹을 IAM 역할로 매핑하는 어설션을 정의합니다. 조직의 다양한 사용자 및 그룹은 서로 다른 IAM 역할에 매핑될 수 있다는 점에 유의하세요. 매핑 수행을 위한 정확한 절차는 사용하고 있는 IdP에 따라 다릅니다. 사용자의 경우 Amazon S3 폴더에 대한 앞

의 시나리오에서 모든 사용자가 Amazon S3 권한을 제공하는 동일한 역할에 매핑될 수 있습니다. 자세한 내용은 인증 응답에 대한 SAML 어설션 구성 단원을 참조하십시오.

IdP가 AWS 콘솔에 대한 SSO를 지원하는 경우, 콘솔 세션의 최대 지속 기간을 구성할 수 있습니다. 자세한 내용은 SAML 2.0 페더레이션 사용자가 AWS Management Console에 액세스할 수 있게 하기 단원을 참조하십시오.

6. 생성 중인 애플리케이션에서 AWS Security Token Service `AssumeRoleWithSAML` API를 호출해 그것을 Step 3 단계에서 생성한 SAML 공급자의 ARN, Step 4 단계에서 생성한 수임할 역할의 ARN 및 IdP에서 얻는 현재 사용자에 대한 SAML 어설션으로 전달합니다. AWS는 역할 수임 요청이 SAML 공급자에서 참조된 IdP로부터 오는지 확인합니다.

   자세한 내용은 AWS Security Token Service API 참조의 AssumeRoleWithSAML을 참조하세요.

7. 요청이 성공하면 API는 일련의 임시 보안 자격 증명을 반환하고 애플리케이션은 이를 사용해 AWS에 서명된 요청을 보냅니다. 애플리케이션은 현재 사용자에 대한 정보를 갖고 있어서 이전 시나리오에 설명된 대로 Amazon S3의 사용자별 폴더에 액세스할 수 있습니다.

AWS 리소스에 대한 SAML 페더레이션 액세스를 허용하는 역할 개요

IAM에서 생성하는 역할에서는 조직의 페더레이션 사용자가 AWS에서 무엇을 할 수 있는지를 정의합니다. 역할에 대한 신뢰 정책을 생성할 때 앞서 생성한 SAML 공급자를 `Principal`로 지정합니다. `Condition`으로 신뢰 정책을 추가로 자세히 살펴봄으로써 특정 SAML 속성과 일치하는 사용자만 그 역할에 액세스하도록 허용할 수 있습니다. 예를 들어 다음 샘플 정책에 설명되어 있듯이 SAML 소속이 `staff`(https://openidp.feide.no에 의해 어설션되듯이)인 사용자만이 그 역할에 액세스할 수 있도록 지정할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"Federated": "arn:aws:iam::account-id:saml-provider/
ExampleOrgSSOProvider"},
    "Action": "sts:AssumeRoleWithSAML",
    "Condition": {
      "StringEquals": {
        "saml:aud": "https://signin.aws.amazon.com/saml",
        "saml:iss": "https://openidp.feide.no"
      },
      "ForAllValues:StringLike": {"saml:edupersonaffiliation": ["staff"]}
    }
```

```
  }]
}
```

> **Note**
>
> 역할 신뢰 정책에 사용되는 SAML IDP는 해당 역할이 속한 계정과 동일한 계정에 있어야 합니다.

정책에서 확인할 수 있는 SAML 키에 대한 자세한 정보는 SAML 기반 AWS STS 연동에 사용할 수 있는 키 섹션을 참조하세요.

`https://region-code.signin.aws.amazon.com/static/saml-metadata.xml`에서 `saml:aud` 속성에 대한 리전 엔드포인트를 포함할 수 있습니다. 가능한 *region-code* 값 목록은 AWS 로그인 엔드포인트의 리전(Region) 열을 참조하세요.

해당 역할의 권한 정책에 대해서는, 역할에 사용하는 방식으로 권한을 지정합니다. 예를 들어 조직의 사용자가 Amazon Elastic Compute Cloud 인스턴스를 관리할 수 있는 경우 AmazonEC2FullAccess 관리형 정책에서처럼 권한 정책에서 Amazon EC2 작업을 명시적으로 허용해야 합니다.

SAML 기반 페더레이션에서 사용자를 고유하게 식별

IAM에서 액세스 정책을 생성할 때 사용자의 자격 증명을 기반으로 권한을 지정할 수 있으면 종종 쓸모가 있습니다. 예를 들어 SAML을 사용하여 페더레이션된 사용자의 경우 애플리케이션에서 Amazon S3에 정보를 다음과 같은 구조를 사용하여 저장할 수 있습니다.

```
myBucket/app1/user1
myBucket/app1/user2
myBucket/app1/user3
```

버킷(`myBucket`)과 폴더(`app1`)는 정적 값이므로 Amazon S3 콘솔 또는 AWS CLI를 통해 생성할 수 있습니다. 그러나 사용자 고유 폴더(*user1*, *user2*, *user3* 등)는 사용자가 연동 프로세스를 통해 최초로 로그인할 때까지 사용자를 식별하는 값이 알려지지 않기 때문에 코드를 사용해 런타임에 생성되어야 합니다.

사용자 고유의 세부 정보를 리소스 이름의 일부로 참조하는 정책을 작성하려면, 정책 조건에서 사용될 수 있는 SAML 키에서 사용자 자격 증명이 사용 가능해야 합니다. 다음 키는 IAM 정책에서 사용할 SAML 2.0 기반 페더레이션에 사용할 수 있습니다. 다음 키에서 반환되는 값을 사용하여 Amazon S3 폴더와 같은 리소스의 고유한 사용자 식별자를 생성할 수 있습니다.

- saml:namequalifier. Issuer 응답 값(saml:iss)과 AWS 계정 ID 및 IAM에서 SAML 공급자의 표시 이름(ARN의 마지막 부분)을 포함하는 문자열의 연결을 기반으로 하는 해시 값 계정 ID와 SAML 공급자 표시 이름의 연결을 IAM 정책에서 saml:doc 키로 사용할 수 있습니다. 계정 ID와 공급자 이름을 "/123456789012/provider_name"처럼 '/'로 구분해야 합니다. 자세한 정보는 saml:doc의 SAML 기반 AWS STS 연동에 사용할 수 있는 키 키를 참조하세요.

  NameQualifier와 Subject의 조합은 페더레이션 사용자를 고유한 이름으로 식별하는 데 사용할 수 있습니다. 다음 유사 코드는 이 값이 계산되는 방식을 보여줍니다. 이 유사 코드에서 +는 연결을 나타내고, SHA1는 SHA-1을 사용해 메시지 다이제스트를 생성하는 기능을 나타내며, Base64는 해시 출력의 Base-64 인코딩 버전을 생성하는 기능을 나타냅니다.

  ```
  Base64 ( SHA1 ( "https://example.com/saml" + "123456789012" + "/
  MySAMLIdP" ) )
  ```

  SAML 기반 연동에 사용 가능한 정책 키에 대한 자세한 정보는 다음(SAML 기반 AWS STS 연동에 사용할 수 있는 키)을 참조하세요.
- saml:sub (문자열). 이것은 클레임의 주체로서 여기에는 조직 내 사용자 개개인을 식별할 수 있는 고유 값이 포함됩니다(예: _cbb88bf52c2510eabe00c1642d4643f41430fe25e3).
- saml:sub_type (문자열). 이 키는 persistent, transient, 또는 SAML 어설션에서 사용되는 Format 및 Subject 요소의 전체 NameID URI일 수 있습니다. persistent라는 값은 saml:sub의 값이 모든 세션에 걸쳐 사용자에게 동일하다는 것을 나타냅니다. 값이 transient인 경우 각 세션마다 사용자의 saml:sub 값이 다릅니다. NameID 요소의 Format 속성에 대한 자세한 내용은 인증 응답에 대한 SAML 어설션 구성 섹션을 참조하세요.

다음 예에서는 앞의 키를 사용하여 Amazon S3의 사용자별 폴더에 대한 권한을 부여하는 권한 정책을 보여줍니다. 해당 정책에서는 saml:namequalifier 및 saml:sub를 모두 포함하는 접두사를 사용하여 Amazon S3 객체를 식별한다고 가정합니다. Condition 요소에는 saml:sub_type이 persistent로 설정되어 있는지 확인하는 테스트가 포함되어 있다는 것에 유의하세요. transient로 설정되어 있다면 사용자에 대한 saml:sub 값은 각 세션마다 다를 수 있고 값의 조합은 사용자 고유 폴더를 식별하는 데 사용되어서는 안 됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:PutObject",
```

```
      "s3:DeleteObject"
    ],
    "Resource": [
      "arn:aws:s3:::exampleorgBucket/backup/${saml:namequalifier}/${saml:sub}",
      "arn:aws:s3:::exampleorgBucket/backup/${saml:namequalifier}/${saml:sub}/*"
    ],
    "Condition": {"StringEquals": {"saml:sub_type": "persistent"}}
  }
}
```

IdP의 어설션을 정책 키에 매핑하는 것에 대한 자세한 정보는 인증 응답에 대한 SAML 어설션 구성 섹션을 참조하세요.

IAM에서 SAML ID 공급자 생성

IAM SAML 2.0 자격 증명 공급자는 SAML 2.0(Security Assertion Markup Language 2.0) 표준을 지원하는 외부 자격 증명 공급자(IdP) 서비스를 기술하는 IAM의 엔터티입니다. 조직의 사용자가 AWS 리소스에 액세스할 수 있도록 Shibboleth 또는 Active Directory 페더레이션 서비스와 같은 SAML 호환 IdP와 AWS 간에 신뢰를 구축하고자 할 때 IAM 자격 증명 공급자를 사용합니다. IAM SAML 자격 증명 공급자는 IAM 신뢰 정책에서 보안 주체로 사용됩니다.

이 시나리오에 대한 자세한 내용은 SAML 2.0 연동를 참조하세요.

AWS Management Console에서 또는 AWS CLI, Tools for Windows PowerShell 또는 AWS API 호출을 사용하여 IAM 자격 증명 공급자를 생성하고 관리할 수 있습니다.

SAML 공급자를 생성한 후에는 1개 이상의 IAM 역할을 생성해야 합니다. 역할은 AWS의 자격 증명으로서 자신만의 고유한 자격 증명이 없지만(사용자가 그러하듯이), 이 콘텍스트에서는 조직의 IdP에 의해 인증된 페더레이션 사용자에게 동적으로 할당됩니다. 그 역할은 조직의 IdP가 AWS에 액세스하기 위해 임시 보안 자격 증명을 요청할 수 있도록 허용합니다. 역할에 할당된 정책은 페더레이션 사용자가 AWS에서 하도록 허용된 것이 무엇인지 결정합니다. SAML 연동을 위한 역할을 생성하려면 타사 ID 제공업체의 역할 생성(페더레이션) 섹션을 참조하세요.

마지막으로 역할을 만든 후에는 AWS에 대한 정보와 페더레이션 사용자가 사용하도록 하고 싶은 역할(들)로 IdP를 구성하여 SAML 신뢰를 완료합니다. 이를 가리켜 IdP와 AWS 간 신뢰 당사자 신뢰 구성이라고 합니다. 신뢰 당사자 신뢰를 구성하려면 신뢰 당사자 신뢰 및 클레임 추가를 통해 SAML 2.0 IdP 구성 섹션을 참조하세요.

주제

- 사전 조건

- IAM SAML 자격 증명 공급자 생성 및 관리(콘솔)
- IAM SAML 자격 증명 공급자 생성(AWS CLI)
- IAM SAML 자격 증명 공급자 생성 및 관리(AWS API)
- 다음 단계

사전 조건

SAML 자격 증명 공급자를 생성하려면 IdP에서 가져온 다음 정보가 있어야 합니다.

- IdP에서 SAML 메타데이터 문서를 가져옵니다. 이 문서에는 발급자 이름, 만료 정보 및 IdP에서 가져온 SAML 인증 응답(어설션)을 확인하는 데 사용할 수 있는 키가 포함되어 있습니다. 메타데이터 문서를 생성하려면 외부 IdP가 제공하는 자격 증명 관리 소프트웨어를 사용하세요.

> **Important**
>
> 이 메타데이터 파일에는 발급자 이름, 만료 정보 및 IdP에서 가져온 SAML 인증 응답(어설션)을 확인하는 데 사용할 수 있는 키가 포함되어 있습니다. 메타데이터 파일은 바이트 순서 표시(BOM)가 없는 UTF-8 형식으로 인코딩되어야 합니다. BOM을 제거하려면 Notepad++와 같은 텍스트 편집 도구를 사용해 파일을 UTF-8로 인코딩합니다.
> SAML 메타데이터 문서의 일부로 포함된 x.509 인증서는 1,024비트 이상의 키를 사용해야 합니다. 또한 x.509 인증서에는 반복되는 확장이 없어야 합니다. 확장을 사용할 수 있지만 확장은 인증서에 한 번만 나타날 수 있습니다. x.509 인증서가 두 조건 중 하나를 충족하지 못하면 IdP 생성에 실패하고 “메타데이터를 구문 분석할 수 없음” 오류를 반환합니다.
> SAML V2.0 Metadata Interoperability Profile Version 1.0의 정의에 따라, IAM은 메타데이터 문서의 X.509 인증서 만료를 평가하거나 이에 대해 조치를 취하지 않습니다.

필요한 SAML 메타데이터 문서를 생성하는 방법을 비롯해, 사용 가능한 다수의 IdP를 구성하여 AWS에서 작동되도록 하는 방법에 대한 지침은 서드 파티 SAML 솔루션 공급자를 AWS와 통합 섹션을 참조하세요.

SAML 페더레이션에 대한 도움이 필요하면 SAML 페더레이션 문제 해결을 참조하세요.

IAM SAML 자격 증명 공급자 생성 및 관리(콘솔)

AWS Management Console을 사용하여 IAM SAML 자격 증명 공급자를 생성, 업데이트, 삭제할 수 있습니다. SAML 페더레이션에 대한 도움이 필요하면 SAML 페더레이션 문제 해결을 참조하세요.

IAM SAML 자격 증명 공급자를 생성하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 [자격 증명 공급자(Identity providers)]를 선택한 다음 [공급자 추가(Add provider)]를 선택합니다.
3. [공급자 구성(Configure provider)]에 대해 [SAML]을 선택합니다.
4. 자격 증명 공급자의 이름을 입력합니다.
5. [메타데이터 문서(Metadata document)에서 [파일 선택(Choose file)을 선택하고 the section called "사전 조건"에서 다운로드한 SAML 메터데이터 문서를 지정합니다.
6. (선택 사항) [태그 추가(Add tags)]에 키 값 페어를 추가하여 IdP를 식별하고 구성할 수 있습니다. 태그를 사용하여 AWS 리소스에 대한 액세스를 제어할 수도 있습니다. SAML 자격 증명 공급자의 태깅에 대한 자세한 내용은 IAM SAML ID 제공업체 태깅 섹션을 참조하세요.

   태그 추가를 선택합니다. 각 태그 키 값 페어의 값을 입력합니다.
7. 제공한 정보를 확인합니다. 완료되면 [공급자 추가(Add provider)]를 선택합니다.
8. 자격 증명 공급자에 IAM 역할을 할당하여 자격 증명 공급자가 관리하는 외부 사용자 자격 증명에 계정의 AWS 리소스에 대한 액세스 권한을 부여합니다. ID 연동을 위한 역할을 생성하는 방법에 대한 자세한 내용은 타사 ID 제공업체의 역할 생성(페더레이션) 섹션을 참조하세요.

> **Note**
>
> 역할 신뢰 정책에 사용되는 SAML IDP는 해당 역할이 속한 계정과 동일한 계정에 있어야 합니다.

SAML 공급자를 삭제하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 [자격 증명 공급자(Identity providers)]를 선택합니다.
3. 삭제할 자격 증명 공급자 옆의 라디오 버튼을 선택합니다.
4. Delete(삭제)를 선택합니다. 새 창이 열립니다.
5. 필드에 단어 `delete`를 입력하여 공급자 삭제를 확인합니다. 그런 다음 삭제를 선택합니다.

IAM SAML 자격 증명 공급자 생성(AWS CLI)

AWS CLI를 사용하여 SAML 공급자를 생성, 업데이트, 삭제할 수 있습니다. SAML 페더레이션에 대한 도움이 필요하면 SAML 페더레이션 문제 해결을 참조하세요.

IAM 자격 증명 공급자를 생성하고 메타데이터 문서를 업로드하려면(AWS CLI)

- 다음 명령을 실행합니다. `aws iam create-saml-provider`

IAM SAML 자격 증명 공급자를 업데이트하려면(AWS CLI)

- 다음 명령을 실행합니다. `aws iam update-saml-provider`

기존 IAM 자격 증명 공급자를 태깅하려면(AWS CLI)

- 다음 명령을 실행합니다. `aws iam tag-saml-provider`

기존 IAM 자격 증명 공급자의 태그를 나열하려면(AWS CLI)

- 다음 명령을 실행합니다. `aws iam list-saml-provider-tags`

기존 IAM 자격 증명 공급자에서 태그를 제거하려면(AWS CLI)

- 다음 명령을 실행합니다. `aws iam untag-saml-provider`

IAM SAML 자격 증명 공급자를 삭제하려면(AWS CLI)

1. (선택 사항) ARN, 생성 날짜, 만료 등 모든 공급자에 대한 정보를 나열하려면 다음 명령을 실행합니다.
   - `aws iam list-saml-providers`
2. (선택 사항) ARN, 생성 날짜, 만료 날짜, 암호화 설정, 프라이빗 키 정보 등 특정 공급자에 대한 정보를 얻으려면 다음 명령을 실행합니다.
   - `aws iam get-saml-provider`
3. IAM 자격 증명 공급자를 삭제하려면 다음 명령을 실행합니다.
   - `aws iam delete-saml-provider`

IAM SAML 자격 증명 공급자 생성 및 관리(AWS API)

AWS API를 사용하여 SAML 공급자를 생성, 업데이트, 삭제할 수 있습니다. SAML 페더레이션에 대한 도움이 필요하면 SAML 페더레이션 문제 해결을 참조하세요.

IAM 자격 증명 공급자를 생성하고 메타데이터 문서를 업로드하려면(AWS API)

- 다음 연산을 호출합니다. `CreateSAMLProvider`

IAM SAML 자격 증명 공급자를 업데이트하려면(AWS API)

- 다음 연산을 호출합니다. `UpdateSAMLProvider`

기존 IAM 자격 증명 공급자를 태깅하려면(AWS API)

- 다음 연산을 호출합니다. `TagSAMLProvider`

기존 IAM 자격 증명 공급자의 태그를 나열하려면(AWS API)

- 다음 연산을 호출합니다. `ListSAMLProviderTags`

기존 IAM 자격 증명 공급자에서 태그를 제거하려면(AWS API)

- 다음 연산을 호출합니다. `UntagSAMLProvider`

IAM 자격 증명 공급자를 삭제하려면(AWS API)

1. (선택 사항)ARN, 생성 날짜, 만료 등 모든 IdP에 대한 정보를 나열하려면 다음 연산을 호출합니다.
   - `ListSAMLProviders`
2. (선택 사항) ARN, 생성 날짜, 만료 날짜, 암호화 설정, 프라이빗 키 정보 등 특정 공급자에 대한 정보를 얻으려면 다음 연산을 호출합니다.
   - `GetSAMLProvider`
3. IdP를 삭제하려면 다음 연산을 호출합니다.
   - `DeleteSAMLProvider`

다음 단계

SAML ID 제공업체를 생성한 후 IdP와의 신뢰 당사자 트러스트를 설정합니다. IdP의 인증 응답에서 가져온 클레임을 정책에서 사용하여 역할에 대한 액세스를 제어할 수도 있습니다.

- IdP에 서비스 공급자로서 AWS의 정보를 알려야 합니다. 이를 가리켜 IdP와 AWS 간 신뢰 당사자 트러스트 추가라고 합니다. 신뢰 당사자 신뢰를 추가하기 위한 정확한 프로세스는 사용 중인 IdP에 따라 달라집니다. 세부 정보는 신뢰 당사자 신뢰 및 클레임 추가를 통해 SAML 2.0 IdP 구성을 참조하세요.
- IdP가 AWS에 클레임이 포함된 리소스를 전송하는 경우 수신 클레임 중 다수가 AWS 콘텍스트 키에 매핑됩니다. Condition 요소를 사용하여 IAM 정책에서 이러한 컨텍스트 키를 사용하고 역할에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 인증 응답에 대한 SAML 어설션 구성을 참조하세요.

신뢰 당사자 신뢰 및 클레임 추가를 통해 SAML 2.0 IdP 구성

SAML 액세스를 위한 IAM 자격 증명 공급자 및 역할을 생성한다는 것은 외부 자격 증명 공급자(IdP)와 관련 사용자에게 허용된 작업을 AWS에 알려주는 것입니다. 그 다음 단계는 IdP에게 서비스 공급자인 AWS에 대해 알려주는 것입니다. 이를 가리켜 IdP와 AWS 간 신뢰 당사자 신뢰 추가라고 합니다. 신뢰 당사자 신뢰를 추가하기 위한 정확한 프로세스는 사용 중인 IdP에 따라 달라집니다. 자세한 정보는 자격 증명 관리 소프트웨어의 설명서를 참조하세요.

오늘날 IdP는 신뢰 당사자 정보와 인증서가 저장된 XML 문서를 IdP가 읽을 수 있도록 URL 지정을 허용하는 곳이 많습니다. AWS에 `https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` 또는 `https://signin.aws.amazon.com/static/saml-metadata.xml`을 사용합니다. 가능한 *region-code* 값 목록은 AWS 로그인 엔드포인트의 리전(Region) 열을 참조하세요.

URL을 직접 지정할 수 없는 경우 위 URL에서 XML 문서를 다운로드하여 IdP 소프트웨어로 가져오면 됩니다.

또한, AWS를 신뢰 당사자로 지정하는 IdP에서는 적절한 클레임 규칙을 생성해야 합니다. IdP가 AWS 엔드포인트에 보내는 SAML 응답에는 클레임이 하나 이상 있는 SAML 어설션이 포함됩니다. 클레임은 사용자 및 사용자 소속 그룹에 대한 정보입니다. 클레임 규칙은 그 정보를 SAML 속성에 매핑합니다. 이는 IAM 정책에서 AWS가 페더레이션 사용자의 권한을 검사하는 데 필요한 속성이 IdP의 SAML 인증 응답에 저장되어 있는지 확인하도록 해줍니다. 자세한 정보는 다음 주제를 참조하십시오.

- AWS 리소스에 대한 SAML 페더레이션 액세스를 허용하는 역할 개요. 이 주제에서는 IAM 정책의 SAML별 키 사용을 비롯해 이 키를 사용하여 SAML 페더레이션 사용자의 권한을 제한하는 방법에 대해 살펴봅니다.

- 인증 응답에 대한 SAML 어설션 구성. 이 주제에서는 사용자에 대한 정보가 포함된 SAML 클레임을 구성하는 방법에 대해 살펴봅니다. 그 클레임은 SAML 어설션에 번들링되어 있으며 AWS로 전동되는 SAML 응답에 포함되어 있습니다. AWS 정책에 필요한 그 정보가 AWS가 인식하고 사용할 수 있는 형식으로 SAML 어설션에 반드시 포함되도록 해야 합니다.
- 서드 파티 SAML 솔루션 공급자를 AWS와 통합. 이 주제에서는 아이덴티티 솔루션과 AWS의 통합 방법에 대한 타사 설명서 링크를 제공합니다.

> **Note**
>
> 페더레이션 복원력을 개선하려면 여러 SAML 로그인 엔드포인트를 지원하도록 IdP 및 AWS 페더레이션을 구성하는 것이 좋습니다. 자세한 내용은 AWS 보안 블로그 문서 How to use regional SAML endpoints for failover(장애 조치에 리전 SAML 엔드포인트를 사용하는 방법)를 참조하세요.

서드 파티 SAML 솔루션 공급자를 AWS와 통합

> **Note**
>
> 인간 사용자가 AWS에 액세스할 때 임시 보안 인증을 사용하도록 하는 것이 좋습니다. AWS IAM Identity Center 사용을 고려해 보셨나요? IAM Identity Center를 사용하여 여러 AWS 계정에 대한 액세스를 중앙에서 관리하고 사용자에게 한 곳에서 할당된 모든 계정에 대한 MFA 보호 Single Sign-On 액세스를 제공할 수 있습니다. IAM Identity Center를 사용하면 IAM Identity Center에서 사용자 자격 증명을 생성하고 관리하거나 기존 SAML 2.0 호환 자격 증명 제공업체에 쉽게 연결할 수 있습니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 IAM Identity Center란 무엇인가요? 섹션을 참조하세요.

다음 링크는 AWS 연동을 처리할 타사 SAML 2.0 자격 증명 공급자(IdP) 솔루션을 구성하는 데 도움이 됩니다.

> **Tip**
>
> AWS Support 엔지니어는 타사 소프트웨어를 사용하는 몇 가지 통합 작업과 함께 비즈니스 및 엔터프라이즈 지원 계획이 있는 고객을 지원할 수 있습니다. 지원되는 플랫폼 및 애플리케이션

의 최신 목록은 AWS Support FAQ에서 지원되는 타사 소프트웨어는 무엇입니까?를 참조하세요.

Solution	추가 정보
Auth0	Amazon Web Services와 통합 - Auth0 설명서 웹 사이트의 이 페이지에는 AWS Management Console에서 통합 인증(SSO)을 설정하는 방법을 설명하는 리소스에 대한 링크가 있으며, JavaScript 예제가 포함되어 있습니다. 세션 태그를 전달하도록 Auth0을 구성할 수 있습니다. 자세한 내용은 Auth0, IAM 세션 태그에 관한 AWS와의 파트너십 발표를 참조하세요.
Microsoft Entra	자습서: AWS 단일 계정 액세스와 Microsoft Entra SSO 통합 - Microsoft 웹 사이트의 이 자습서에서는 SAML 페더레이션을 사용하여 Microsoft Entra(이전 Azure AD)를 ID 제공업체(IdP)로 설정하는 방법을 설명합니다.
Centrify	Centrify 구성 및 AWS에 SSO용 SAML 사용 - Centrify 웹 사이트의 이 페이지는 Centrify를 구성해 SSO를 위한 SAML을 AWS에 사용하는 방법에 대해 설명합니다.
CyberArk	CyberArk를 구성하여 CyberArk 사용자 포털에서 SAML 통합 인증(SSO)을 통해 로그인하는 사용자에게 Amazon Web Services(AWS) 액세스를 제공합니다.
ForgeRock	ForgeRock Identity Platform이 AWS와 통합됩니다. 세션 태그를 전달하도록 ForgeRock을 구성할 수 있습니다. 자세한 내용은 Amazon Web Services의 속성 기반 액세스 제어를 참조하세요.
Google Workspace	Amazon Web Services 클라우드 애플리케이션 - Google Workspace Admin Help 사이트에 있는 이 문서는 AWS를 서비스 공급자로 하여 Google Workspace를 SAML 2.0 IdP로 구성하는 방법을 설명합니다.

Solution	추가 정보
IBM	세션 태그를 전달하도록 IBM을 구성할 수 있습니다. 자세한 내용은 AWS 세션 태그를 지원하는 최초 제품 중 하나인 IBM Cloud Identity IDaaS를 참조하세요.
JumpCloud	Amazon AWS에서 Single Sign On(SSO)을 위한 IAM 역할을 통해 액세스 권한 부여 - JumpCloud 웹 사이트의 이 문서에서는 AWS의 IAM 역할을 기반으로 SSO를 설정하고 활성화하는 방법을 설명합니다.
Matrix42	MyWorkspace 시작 안내서 - 이 안내서에서는 AWS Identity 서비스를 Matrix42 MyWorkspace와 통합하는 방법에 대해 설명합니다.
Microsoft AD FS(Active Directory Federation Services)	Field Notes: Integrating Active Directory Federation Service with AWS IAM Identity Center(필드 노트: Active Directory Federation Service와 SSO 통합) - AWS 아키텍처 블로그의 이 게시물에서는 AD FS와 AWS IAM Identity Center(IAM Identity Center) 간의 인증 흐름을 설명합니다. IAM Identity Center는 SAML 2.0을 통한 아이덴티티 페더레이션을 지원하므로 AD FS 솔루션과의 통합이 가능합니다. 사용자는 회사 보안 인증을 사용하여 IAM Identity Center 포털에 로그인할 수 있으므로 IAM Identity Center에서 별도의 보안 인증을 유지 관리하는 관리 오버헤드를 줄일 수 있습니다. 세션 태그를 전달하도록 AD FS를 구성할 수도 있습니다. 자세한 내용은 AD FS에서 속성 기반 액세스 제어를 사용하여 IAM 권한 관리 간소화를 참조하세요.
miniOrange	AWS용 - miniOrange 웹 사이트의 이 페이지에서는 기업용 AWS에 대한 보안 액세스 및 AWS 애플리케이션에 대한 완전한 액세스 제어를 설정하는 방법을 설명합니다.

Solution	추가 정보
Okta	Okta를 사용하여 Amazon Web Services 명령줄 인터페이스 통합 - Okta 지원 사이트의 이 페이지에서는 Okta를 AWS와 함께 사용하도록 구성하는 방법을 알아볼 수 있습니다. 세션 태그를 전달하도록 Okta를 구성할 수 있습니다. 자세한 내용은 Okta 및 AWS 파트너, 세션 태그를 통해 액세스 간소화를 참조하세요.
Okta	AWS 계정 페더레이션 – Okta 웹사이트의 이 섹션은 AWS에 대해 IAM Identity Center를 설정하고 지원하는 방법을 설명합니다.
OneLogin	OneLogin Knowledgebase에서 **SAML AWS**라는 검색어를 입력하여 단일 역할 및 다중 역할 시나리오를 위해 OneLogin과 AWS 사이에 IAM Identity Center 기능을 설정하는 방법이 설명된 일련의 문서를 찾습니다. 세션 태그를 전달하도록 OneLogin을 구성할 수 있습니다. 자세한 내용은 OneLogin 및 세션 태그: AWS 리소스에 대한 속성 기반 액세스 제어를 참조하세요.
Ping Identity	PingFederate AWS 커넥터 - 통합 인증(SSO) 및 프로비저닝 연결을 쉽게 설정할 수 있는 빠른 연결 템플릿인 PingFederate AWS 커넥터에 대한 세부 정보를 봅니다. 설명서를 읽고 AWS와의 통합을 위한 최신 PingFederate AWS Connector를 다운로드합니다. 세션 태그를 전달하도록 Ping ID를 구성할 수 있습니다. 자세한 내용은 AWS에서 속성 기반 액세스 제어를 위한 Ping Identity 지원 발표를 참조하세요.
RadiantLogic	Radiant Logic 기술 파트너 - Radiant Logic의 RadiantOne 페더레이션 자격 증명 서비스를 AWS와 통합하여 SAML 기반의 SSO를 위한 자격 증명 허브를 구축할 수 있습니다.
RSA	Amazon Web Services - RSA Ready Implementation Guide에서는 AWS 및 RSA 구현 안내를 제공합니다. SAML 구성에 대한 자세한 내용은 Amazon Web Services - SAML My Page SSO Configuration - RSA Ready Implementation Guide를 참조하세요.

Solution	추가 정보
Salesforce.com	Salesforce에서 AWS와의 SSO를 구성하는 방법 - Salesforce.com 개발자 사이트에 실린 이 사용 방법 설명서에서는 Salesforce에 IdP(자격 증명 공급자)를 설정하고 AWS를 서비스 공급자로 구성하는 방법을 설명합니다.
SecureAuth	AWS - SecureAuth SAML SSO - SecureAuth 웹 사이트 관련에 관한 이 문서는 SecureAuth 어플라이언스를 위해 SAML과 AWS의 통합을 설정하는 방법을 설명합니다.
Shibboleth	AWS Management Console에 대한 SSO를 위해 Shibboleth를 사용하는 방법 - AWS 보안 블로그의 이 항목에서는 Shibboleth를 설정하고 이를 AWS에 대한 자격 증명 공급자로 구성하는 방법을 단계별로 안내합니다. 세션 태그를 전달하도록 Shibboleth를 구성할 수 있습니다.

자세한 정보는 AWS 웹 사이트의 IAM 파트너 페이지를 참조하세요.

인증 응답에 대한 SAML 어설션 구성

조직에서 사용자의 아이덴티티를 확인한 후 외부 ID 제공자(IdP)가 `https://region-code.signin.aws.amazon.com/saml`의 AWS SAML 엔드포인트로 인증 응답을 보냅니다. 잠재적 *region-code* 대체 목록은 AWS 로그인 엔드포인트의 리전(Region) 열을 참조하세요. 이 응답은 SAML 2.0을 위한 HTTP POST 바인딩 표준을 준수하고 다음 요소 또는 클레임이 저장된 SAML 토큰을 포함하는 POST 요청입니다. SAML 호환 IdP에서 이 클레임들을 구성합니다. 이 클레임들을 입력하는 방법에 대한 지침에 대해서는 귀하의 IdP를 위한 문서를 참고하세요.

IdP가 AWS에 클레임이 포함된 리소스를 전송하는 경우 수신 클레임 중 다수가 AWS 콘텍스트 키에 매핑됩니다. 이러한 컨텍스트 키는 `Condition` 요소를 사용하여 IAM 정책에서 확인할 수 있습니다. 사용 가능한 매핑 목록은 SAML 속성을 AWS 신뢰 정책 컨텍스트 키에 매핑 섹션에 나와 있습니다.

`Subject` 및 `NameID`

다음 발췌문은 한 가지 예를 보여줍니다. 자신의 값을 표시된 것으로 대치합니다. `SubjectConfirmation` 속성과 `SubjectConfirmationData` 속성을 둘 다 포함하는 `NotOnOrAfter` 요소와 함께 정확하게 `Recipient` 요소가 하나 있어야 합니다. 이러한 속성에는 AWS 엔드포인트 `https://region-code.signin.aws.amazon.com/saml`과 일치해야 하는 값

이 포함되어 있습니다. 가능한 *region-code* 값 목록은 AWS 로그인 엔드포인트의 리전(Region) 열을 참조하세요. AWS 값의 경우 다음 예제와 같이 `https://signin.aws.amazon.com/static/saml`을 사용할 수도 있습니다.

`NameID` 요소는 영구 값, 임시 값을 갖거나 IdP 솔루션에서 제공한 전체 형식 URI로 구성될 수 있습니다. 영구 값은 `NameID`의 값이 세션 간 사용자에서도 동일하다는 것을 나타냅니다. 임시 값인 경우 각 세션마다 사용자의 `NameID` 값이 다릅니다. 싱글 사인온 상호작용은 다음과 같은 유형의 식별자를 지원합니다.

- `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`

```
<Subject>
  <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent">_cbb88bf52c2510eabe00c1642d4643f41430fe25e3</NameID>
  <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
    <SubjectConfirmationData NotOnOrAfter="2013-11-05T02:06:42.876Z"
 Recipient="https://signin.aws.amazon.com/saml"/>
  </SubjectConfirmation>
</Subject>
```

> **Important**
>
> `saml:aud` 콘텍스트 키는 SAML 수신자 속성에서 온 것으로, 그 이유는 이 속성이 `accounts.google.com:aud`와 같은 OIDC 대상 필드와 동일한 SAML이기 때문입니다.

`PrincipalTag` SAML 속성

(선택 사항) `Name` 속성이 `https://aws.amazon.com/SAML/Attributes/PrincipalTag:{TagKey}`로 설정된 `Attribute` 요소를 사용할 수 있습니다. 이 요소를 사용하면

속성을 SAML 어설션에 세션 태그로 전달할 수 있습니다. 세션 태그에 대한 자세한 내용은 AWS STS에서 세션 태그 전달 섹션을 참조하세요.

속성을 세션 태그로 전달하려면 태그 값을 지정하는 `AttributeValue` 요소를 포함합니다. 예를 들어, 태그 키 값 페어 `Project = Marketing` 및 `CostCenter = 12345`를 전달하려면 다음 속성을 사용합니다. 각 태그에 대해 별도의 `Attribute` 요소를 포함합니다.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Project">
  <AttributeValue>Marketing</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:CostCenter">
  <AttributeValue>12345</AttributeValue>
</Attribute>
```

위의 태그를 전이적으로 설정하려면 `Name` 속성이 `https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys`로 설정된 다른 `Attribute` 요소를 포함합니다. 세션 태그를 전이적으로 설정하는 선택적 다중 값 속성입니다. 전이적 태그는 SAML 세션을 사용하여 AWS에서 다른 역할을 맡을 때 유지됩니다. 이를 역할 체인이라고 합니다. 예를 들어, `Principal` 및 `CostCenter` 태그를 전이적으로 설정하려면 다음 속성을 사용하여 키를 지정합니다.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys">
  <AttributeValue>Project</AttributeValue>
  <AttributeValue>CostCenter</AttributeValue>
</Attribute>
```

`Role` SAML 속성

`Name` 속성이 `https://aws.amazon.com/SAML/Attributes/Role`로 설정된 `Attribute` 요소를 사용할 수 있습니다. 이 요소는 IdP에 의해 사용자가 매핑되는 IAM 자격 증명 공급자 및 역할을 나열하는 `AttributeValue` 요소를 한 개 이상 포함합니다. IAM 역할 및 IAM 자격 증명 공급자는 AssumeRoleWithSAML에 전달되는 `RoleArn` 및 `PrincipalArn` 파라미터와 동일한 형식의 쉼표로 구분된 ARN 쌍으로 지정됩니다. 이 요소는 하나 이상의 역할 공급자 페어(`AttributeValue` 요소)를 포함해야 하며 여러 페어를 포함할 수 있습니다. 요소가 다수의 페어를 포함하는 경우 사용자가 WebSSO를 사용하여 AWS Management Console에 로그인할 때 어떤 역할을 수임할지 선택하라는 메시지가 표시됩니다.

> **Important**
>
> Name 태그의 Attribute 속성 값은 대/소문자를 구분합니다. 정확하게 https://aws.amazon.com/SAML/Attributes/Role로 설정해야 합니다.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/Role">
  <AttributeValue>arn:aws:iam::account-number:role/role-name1,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
  <AttributeValue>arn:aws:iam::account-number:role/role-name2,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
  <AttributeValue>arn:aws:iam::account-number:role/role-name3,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
</Attribute>
```

RoleSessionName SAML 속성

Name 속성이 https://aws.amazon.com/SAML/Attributes/RoleSessionName로 설정된 Attribute 요소를 사용할 수 있습니다. 이 요소에는 역할을 수임할 때 발급된 임시 자격 증명의 식별자를 제공하는 AttributeValue 요소가 하나 포함되어 있습니다. 이 요소를 사용하여 임시 자격 증명을 애플리케이션을 사용하는 사용자와 연결할 수 있습니다. 이 요소는 AWS Management Console 콘솔에서 사용자 정보를 표시하는 데 사용됩니다. AttributeValue 요소의 값은 2~64자여야 하며 영숫자, 밑줄 및 다음 문자만 포함할 수 있습니다. . , + = @ -(하이픈). 공백은 포함할 수 없습니다. 값은 일반적으로 사용자 ID(johndoe) 또는 이메일 주소(johndoe@example.com)입니다. 사용자의 표시 이름(John Doe)과 같이 값이 공백을 포함하면 안 됩니다.

> **Important**
>
> Name 태그의 Attribute 속성 값은 대/소문자를 구분합니다. 정확하게 https://aws.amazon.com/SAML/Attributes/RoleSessionName로 설정해야 합니다.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/RoleSessionName">
  <AttributeValue>user-id-name</AttributeValue>
</Attribute>
```

SessionDuration SAML 속성

(선택 사항) Name 속성이 https://aws.amazon.com/SAML/Attributes/SessionDuration"로 설정된 Attribute 요소를 사용할 수 있습니다. 이 요소에는 사용자가 AWS Management Console에 액세스할 수 있는 기간을 지정하는 AttributeValue 요소가 한 개 포함되어 있습니다. 이 시간이 지나면 새로운 임시 자격 증명을 요청해야 합니다. 이 값은 세션에 대한 기간(초)을 나타내는 정수입니다. 이 값은 900초(15분)~43200초(12시간)일 수 있습니다. 이 속성이 없으면 자격 증명은 한 시간 동안 지속됩니다(DurationSeconds API의 AssumeRoleWithSAML 파라미터 기본값).

이 속성을 사용하려면 https://*region-code*.signin.aws.amazon.com/saml에서 콘솔 로그인 웹 엔드포인트를 통해 AWS Management Console에 대한 SSO(Single Sign-On) 액세스를 제공하도록 SAML 공급자를 구성해야 합니다. 가능한 *region-code* 값 목록은 AWS 로그인 엔드포인트의 리전(Region) 열을 참조하세요. URL https://signin.aws.amazon.com/static/saml을 사용할 수 있습니다. 이 속성은 AWS Management Console에 대해서만 세션을 연장할 수 있습니다. 다른 자격 증명의 수명을 늘릴 수는 없습니다. 그러나 AssumeRoleWithSAML API 호출에 존재하는 경우 세션 기간을 단축하는 데 사용할 수 있습니다. 호출에 의해 반환되는 자격 증명의 기본 수명은 60분입니다.

SessionNotOnOrAfter 속성도 정의된 경우 콘솔 세션 중 두 속성 SessionDuration 또는 SessionNotOnOrAfter 중 더 작은 값으로 최대값이 설정됩니다.

지속 기간을 더 늘려 콘솔 세션을 활성화하면 자격 증명이 손상될 위험이 높아집니다. 이러한 위험을 줄이려면 IAM 콘솔의 역할 요약 페이지에서 세션 취소(Revoke Sessions)를 선택하여 원하는 역할의 활성 콘솔 세션을 즉시 비활성화하면 됩니다. 자세한 내용은 IAM 역할 임시 보안 자격 증명 취소 단원을 참조하십시오.

> **Important**
>
> Name 태그의 Attribute 속성 값은 대/소문자를 구분합니다. 정확하게 https://aws.amazon.com/SAML/Attributes/SessionDuration로 설정해야 합니다.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SessionDuration">
  <AttributeValue>1800</AttributeValue>
</Attribute>
```

`SourceIdentity` SAML 속성

(선택 사항) Name 속성이 `https://aws.amazon.com/SAML/Attributes/SourceIdentity`로 설정된 `Attribute` 요소를 사용할 수 있습니다. 이 요소에는 하나의 IAM 역할을 사용하는 사용자 또는 애플리케이션에 식별자를 제공하는 `AttributeValue` 요소가 포함됩니다. 소스 자격 증명의 값은 SAML 세션을 사용하여 AWS에서 다른 역할을 수임할 때(역할 체인이라고 함) 유지됩니다. 소스 자격 증명 값은 역할 세션 중에 수행된 모든 작업의 요청에 표시됩니다. 역할 세션 중에는 설정된 값을 변경할 수 없습니다. 그런 다음 관리자는 AWS CloudTrail 로그를 통해 소스 자격 증명 정보를 모니터링하고 감사하여 공유 역할로 작업을 수행한 사용자를 결정합니다.

`AttributeValue` 요소의 값은 2~64자여야 하며 영숫자, 밑줄 및 다음 문자만 포함할 수 있습니다. . , + = @ -(하이픈). 공백은 포함할 수 없습니다. 이 값은 일반적으로 사용자 ID(johndoe) 또는 이메일 주소(johndoe@example.com) 등 사용자와 연결된 속성입니다. 사용자의 표시 이름(John Doe)과 같이 값이 공백을 포함하면 안 됩니다. 소스 자격 증명 사용에 대한 자세한 내용은 위임된 역할로 수행한 작업 모니터링 및 제어 섹션을 참조하세요.

> **Important**
>
> `SourceIdentity` 속성을 사용하도록 SAML 어설션이 구성된 경우 역할 신뢰 정책에도 `sts:SetSourceIdentity` 작업이 포함되어야 합니다. 그러지 않으면 수임 역할 작업이 실패합니다. 소스 자격 증명 사용에 대한 자세한 내용은 위임된 역할로 수행한 작업 모니터링 및 제어 섹션을 참조하세요.

소스 자격 증명 속성을 전달하려면 소스 자격 증명의 값을 지정하는 `AttributeValue` 요소를 포함합니다. 예를 들어, 소스 자격 증명 `DiegoRamirez`를 전달하려면 다음 속성을 사용합니다.

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SourceIdentity">
  <AttributeValue>DiegoRamirez</AttributeValue>
```

SAML 속성을 AWS 신뢰 정책 컨텍스트 키에 매핑

이 섹션의 표들은 흔히 사용되는 SAML 속성들을 나열하고, 그 속성들이 AWS에서 신뢰 정책 조건 콘텍스트 키에 어떻게 매핑되는지 보여줍니다. 이러한 키를 사용하여 역할에 대한 액세스를 제어할 수 있습니다. 이렇게 하려면 키를 SAML 액세스 요청과 함께 제공되는 어설션에 포함된 값과 비교합니다.

> **Important**
>
> 이런 키는 IAM 신뢰 정책(역할을 수임할 수 있는 사용자를 결정하는 정책)에서만 사용할 수 있고, 권한 정책에는 적용할 수 없습니다.

eduPerson 및 eduOrg 속성 표에서 값은 문자열 또는 문자열 목록의 형태로 입력됩니다. 문자열 값의 경우, `StringEquals` 또는 `StringLike` 조건을 이용해 IAM 신뢰 정책에서 이러한 값을 테스트할 수 있습니다. 문자열 목록이 포함된 값의 경우에는 `ForAnyValue` 및 `ForAllValues` 정책 설정 연산자를 사용해 신뢰 정책에서 값을 테스트합니다.

> **Note**
>
> AWS 컨텍스트 키당 하나의 클레임만을 포함해야 합니다. 하나 이상의 클레임을 포함하는 경우, 하나의 클레임만 매핑됩니다.

다음 표는 eduPerson 및 eduOrg 속성을 보여줍니다.

eduPerson 또는 eduOrg 속성(**`Name`** 키)	이 AWS 콘텍스트 키(**`FriendlyName`** 키)에 매핑	유형
`urn:oid:1.3.6.1.4.1.5923.1.1.1.1`	`eduPerson Affiliation`	문자열 목록
`urn:oid:1.3.6.1.4.1.5923.1.1.1.2`	`eduPersonNickname`	문자열 목록
`urn:oid:1.3.6.1.4.1.5923.1.1.1.3`	`eduPersonOrgDN`	String
`urn:oid:1.3.6.1.4.1.5923.1.1.1.4`	`eduPerson OrgUnitDN`	문자열 목록
`urn:oid:1.3.6.1.4.1.5923.1.1.1.5`	`eduPerson PrimaryAf filiation`	String

eduPerson 또는 eduOrg 속성(**Name** 키)	이 AWS 콘텍스트 키 (**FriendlyName** 키)에 매핑	유형
urn:oid:1.3.6.1.4.1.5923.1.1.1.6	eduPerson PrincipalName	String
urn:oid:1.3.6.1.4.1.5923.1.1.1.7	eduPerson Entitlement	문자열 목록
urn:oid:1.3.6.1.4.1.5923.1.1.1.8	eduPerson PrimaryOrgUnitDN	String
urn:oid:1.3.6.1.4.1.5923.1.1.1.9	eduPerson ScopedAff iliation	문자열 목록
urn:oid:1.3.6.1.4.1.5923.1. 1.1.10	eduPerson TargetedID	문자열 목록
urn:oid:1.3.6.1.4.1.5923.1. 1.1.11	eduPerson Assurance	문자열 목록
urn:oid:1.3.6.1.4.1.5923.1.2.1.2	eduOrgHomePageURI	문자열 목록
urn:oid:1.3.6.1.4.1.5923.1.2.1.3	eduOrgIde ntityAuth NPolicyURI	문자열 목록
urn:oid:1.3.6.1.4.1.5923.1.2.1.4	eduOrgLegalName	문자열 목록
urn:oid:1.3.6.1.4.1.5923.1.2.1.5	eduOrgSuperiorURI	문자열 목록
urn:oid:1.3.6.1.4.1.5923.1.2.1.6	eduOrgWhi tePagesURI	문자열 목록
urn:oid:2.5.4.3	cn	문자열 목록

다음 표는 Active Directory 속성을 보여줍니다.

AD 속성	이 AWS 컨텍스트 키에 매핑	유형
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name	name	문자열
http://schemas.xmlsoap.org/claims/CommonName	commonName	문자열
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname	givenName	문자열
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname	surname	문자열
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress	mail	문자열
http://schemas.microsoft.com/ws/2008/06/identity/claims/primarygroupsid	uid	String

다음 표는 X.500 속성을 보여줍니다.

X.500 속성	이 AWS 컨텍스트 키에 매핑	유형
2.5.4.3	commonName	문자열
2.5.4.4	surname	문자열
2.4.5.42	givenName	문자열
2.5.4.45	x500UniqueIdentifier	문자열
0.9.2342.19200300100.1.1	uid	문자열
0.9.2342.19200300100.1.3	mail	문자열

X.500 속성	이 AWS 컨텍스트 키에 매핑	유형
`0.9.2342.19200300.100.1.45`	`organizationStatus`	String

SAML 2.0 페더레이션 사용자가 AWS Management Console에 액세스할 수 있게 하기

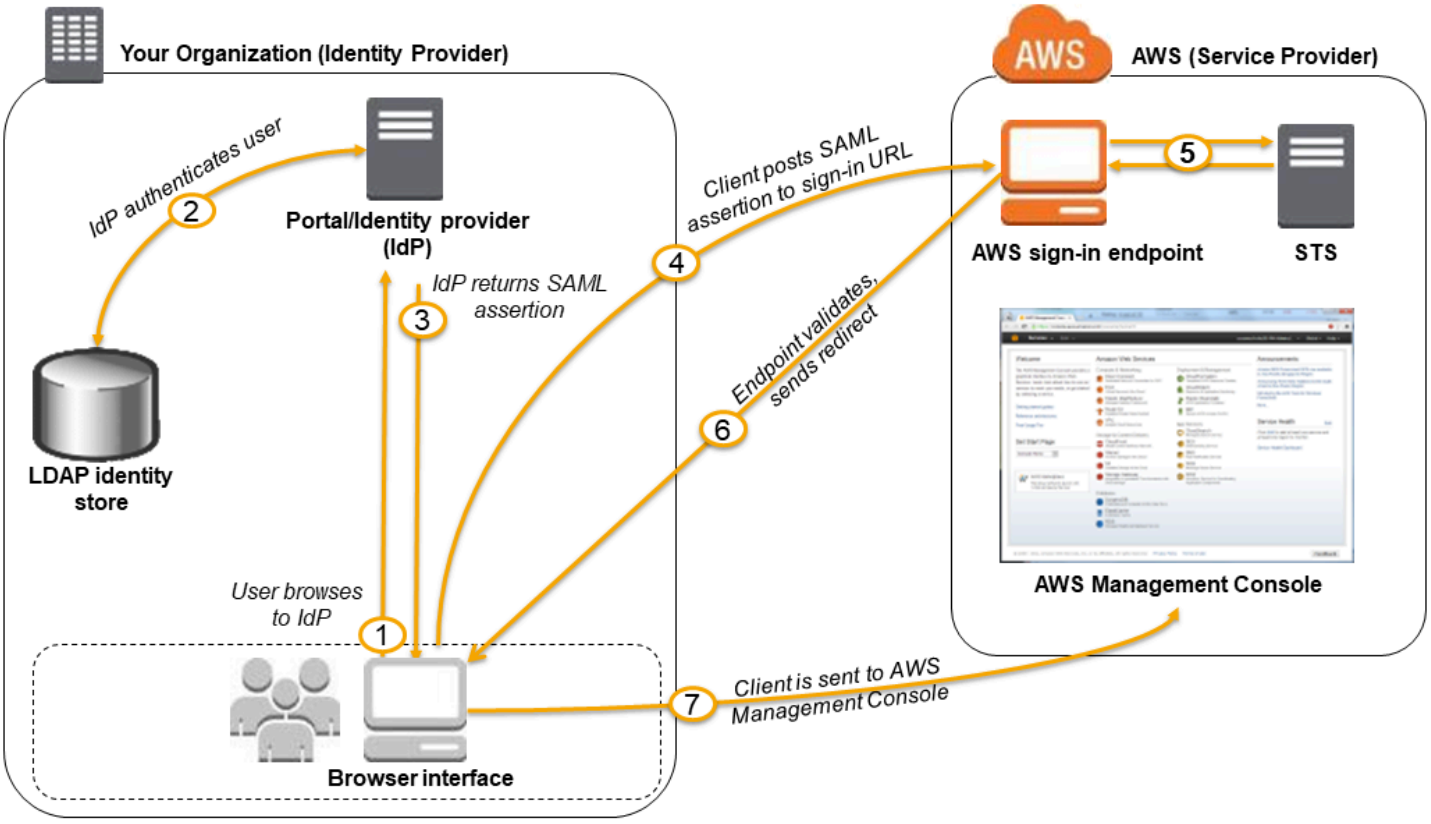
역할을 사용해 SAML 2.0 호환 자격 증명 공급자(IdP) 및 AWS를 구성하여 페더레이션 사용자가 AWS Management Console에 액세스하도록 허용할 수 있습니다. 역할은 콘솔에서 작업을 수행할 수 있는 권한을 사용자에게 부여합니다. SAML 페더레이션 사용자가 다른 방법으로 AWS에 액세스할 수 있게 하려면 다음 주제 중 하나를 참조하세요.

- AWS CLI: IAM 역할로 전환(AWS CLI)
- Tools for Windows PowerShell: IAM 역할로 전환(Tools for Windows PowerShell)
- AWS API: IAM 역할로 전환(AWS API)

개요

다음 다이어그램은 SAML 지원 Single Sign-On의 흐름을 보여줍니다.

> **Note**
>
> 이와 같은 SAML의 특수한 사용이 SAML 2.0 연동에 설명된 더 일반적인 사용과 차이가 나는 이유는, 이 워크플로우가 사용자를 대신해 AWS Management Console을 열기 때문입니다. 이를 위해서는 `AssumeRoleWithSAML` API를 직접 호출하는 대신 AWS 로그인 엔드포인트를 사용해야 합니다. 엔드포인트는 사용자를 위해 API를 호출하고 사용자의 브라우저를 AWS Management Console로 자동 리디렉션하는 URL을 반환합니다.



다이어그램은 다음 단계들을 보여줍니다.

1. 사용자는 검색을 통해 조직의 포털에 이르러 옵션을 선택해 AWS Management Console로 갑니다. 조직에서 포털은 일반적으로, 조직과 AWS 간의 신뢰 교환을 처리하는 IdP 기능입니다. 예를 들어 Active Directory Federation Services에서 포털 URL은 `https://ADFSServiceName/adfs/ls/IdpInitiatedSignOn.aspx`입니다.
2. 포털은 사용자의 조직 내 자격 증명을 확인합니다.
3. 포털은 사용자를 식별하고 사용자에 대한 속성을 포함하는 어설션이 포함된 SAML 인증 응답을 생성합니다. 콘솔 세션의 유효 기간을 지정하는 `SessionDuration`이라는 SAML 어설션 속성을 포함하여 IdP를 구성할 수도 있습니다. 속성을 세션 태그로 전달하도록 IdP를 구성할 수도 있습니다. 포털은 이 응답을 클라이언트 브라우저로 전송합니다.
4. 클라이언트 브라우저는 AWS Single Sign-On 엔드포인트로 리디렉션되고 SAML 어설션을 게시합니다.
5. 엔드포인트는 사용자 대신 임시 보안 자격 증명을 요청하고 그 자격 증명을 사용하는 콘솔 로그인 URL을 생성합니다.
6. AWS는 리디렉션으로 클라이언트에게 로그인 URL을 반송합니다.

7. 클라이언트 브라우저는 AWS Management Console로 리디렉션됩니다. SAML 인증 응답이 여러 개의 IAM 역할에 매핑되는 속성을 포함하는 경우 사용자는 콘솔에 액세스하는 데 사용할 역할을 선택하라는 메시지를 먼저 받습니다.

사용자의 시점에서는 그 과정을 투명하게 들여다볼 수 있습니다. 사용자는 조직의 내부 포털에서 시작하여 AWS 자격 증명을 제공할 필요 없이 AWS Management Console에서 마칩니다.

세부 단계들에 대한 링크를 따라 이 행동을 구성하는 방법을 개관하시려면 다음 섹션들을 참조하세요.

AWS에 대한 SAML 공급자로 네트워크 구성하기

귀하의 조직 네트워크의 내부에서 자격 증명 스토어(Windows Active Directory 등)를 구성해 Windows Active Directory Federation Services, Shibboleth와 같은 SAML 기반 IdP로 작업합니다. IdP를 사용하여 귀하의 조직을 IdP로 기술하고 인증 키를 포함하는 메타데이터 문서를 생성합니다. 또한 조직의 포털을 구성해, AWS Management Console에 대한 사용자 요청을 SAML 어설션을 이용한 인증을 위해 AWS SAML 엔드포인트로 라우팅합니다. metadata.xml 파일을 생성하기 위해 IdP를 어떻게 구성하는가는 IdP에 따라 다릅니다. 지침을 보시려면 IdP의 문서를 참고하시거나 지원되는 SAML 공급자들 중 다수의 웹 문서 링크가 있는 서드 파티 SAML 솔루션 공급자를 AWS와 통합를 참조하세요.

IAM에서 SAML 공급자 생성

그 다음에는 AWS Management Console에 로그인하여 IAM 콘솔로 이동합니다. 그곳에서 새로운 SAML 공급자를 생성합니다. 그 공급자는 조직의 Idp에 대한 정보를 담고 있는 IAM의 엔터티입니다. 이 과정의 일부로 이전 섹션에서 조직의 IdP 소프트웨어가 생성한 메타데이터 문서를 업로드합니다. 자세한 내용은 IAM에서 SAML ID 공급자 생성 섹션을 참조하세요.

페더레이션 사용자들을 위해 AWS에서 권한 구성하기

다음 단계는 조직의 IdP와 IAM 간에 신뢰 관계를 수립하는 IAM 역할을 생성하는 것입니다. 이 역할은 연동을 위해 IdP를 보안 주체(신뢰할 수 있는 엔터티)로 식별해야 합니다. 그 역할은 조직의 IdP에 의해 인증된 사용자들이 AWS에서 할 수 있도록 허용되는 것이 무엇인지 정의하기도 합니다. IAM 콘솔을 사용하여 이 역할을 생성할 수 있습니다. 역할을 수임할 수 있는 사용자를 나타내는 신뢰 정책을 생성할 때 이전에 IAM에서 생성한 SAML 공급자를 지정합니다. 또한 역할을 맡을 수 있도록 사용자가 일치해야 하는 SAML 속성을 하나 이상 지정합니다. 예를 들어 SAML `eduPersonOrgDN` 값이 `ExampleOrg`인 사용자에게만 로그인을 허용하도록 구성할 수 있습니다. 그 역할 마법사는 조건을 자동으로 추가해 `saml:aud` 속성을 테스트함으로써 그 역할이 AWS Management Console에 로그인하는 것을 위해서만 위임되는 것인지 확인합니다. 그 역할을 위한 신뢰 정책은 다음과 같을 수 있습니다.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"Federated": "arn:aws:iam::account-id:saml-provider/
ExampleOrgSSOProvider"},
    "Action": "sts:AssumeRoleWithSAML",
    "Condition": {"StringEquals": {
      "saml:edupersonorgdn": "ExampleOrg",
      "saml:aud": "https://signin.aws.amazon.com/saml"
    }}
  }]
}
```

> **Note**
>
> 역할 신뢰 정책에 사용되는 SAML IdP는 해당 역할이 속한 계정과 동일한 계정에 있어야 합니다.

`https://region-code.signin.aws.amazon.com/static/saml-metadata.xml`에서 `saml:aud` 속성에 대한 리전 엔드포인트를 포함할 수 있습니다. 가능한 *region-code* 값 목록은 AWS 로그인 엔드포인트의 리전(Region) 열을 참조하세요.

역할의 권한 정책에 대해 어떤 역할, 사용자, 또는 그룹에 사용하는 방식으로 권한을 지정합니다. 예를 들어, 조직의 사용자가 Amazon EC2 인스턴스를 관리하도록 허용될 경우 권한 정책에서 명시적으로 Amazon EC2 작업을 허용합니다. 이렇게 하려면 Amazon EC2 모든 액세스 관리형 정책과 같은 관리형 정책을 할당합니다.

SAML IdP를 위한 역할 생성에 관한 자세한 정보는 SAML 2.0 페더레이션을 위한 역할 생성(콘솔)을 참조하세요.

구성 완료 및 SAML 어설션 생성

`https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` 또는 `https://signin.aws.amazon.com/static/saml-metadata.xml`에 있는 `saml-metadata.xml` 파일을 설치하여 AWS가 서비스 제공업체임을 SAML IdP에 알립니다. 가능한 *region-code* 값 목록은 AWS 로그인 엔드포인트의 리전(Region) 열을 참조하세요.

그 파일의 설치 방법은 IdP에 따라 다릅니다. 어떤 IdP는 URL을 입력할 수 있는 옵션을 제공하고, 그 결과 IdP가 그 파일을 획득하고 설치해 줍니다. 다른 IdP들의 경우에는 URL에서 파일을 내려받은 다음

로컬 파일로 제공해야 합니다. 세부 정보를 보시려면 IdP의 문서를 참고하시거나 지원되는 SAML 공급자들 중 다수의 웹 문서 링크가 있는 서드 파티 SAML 솔루션 공급자를 AWS와 통합를 참조하세요.

또한, IdP가 인증 응답의 일부로 AWS에 SAML 속성으로 전달하기 원하는 정보를 구성합니다. 이 정보의 대부분은 정책에서 평가할 수 있는 조건 컨텍스트 키로 AWS에 나타납니다. 이러한 조건 키를 사용하면 올바른 컨텍스트의 승인된 사용자에게만 AWS 리소스에 액세스할 수 있는 권한이 부여됩니다. 콘솔을 사용할 수 있는 시간을 제한하는 시간 창을 지정할 수 있습니다. 또한 사용자가 콘솔에 액세스할 수 있는 최대 시간(최대 12시간)을 지정할 수 있습니다. 사용자는 이 시간 이후에 자신의 자격 증명을 새로 고쳐야 합니다. 자세한 내용은 인증 응답에 대한 SAML 어설션 구성 섹션을 참조하세요.

브라우저에서 SAML 응답 보기

다음 절차에서는 SAML 2.0 관련 문제를 해결할 때 브라우저에서 서비스 공급자의 SAML 응답을 보는 방법에 대해 설명합니다.

브라우저에서 문제를 재현할 수 있는 페이지로 이동합니다. 그런 다음 해당 브라우저의 단계를 따릅니다.

주제

- Google Chrome
- Mozilla Firefox
- Apple Safari
- Base64 인코딩 SAML 응답에 대해 해야 할 작업

Google Chrome

Chrome에서 SAML 응답을 보려면

이 단계는 Google Chrome의 버전 106.0.5249.103(공식 빌드)(arm64)을 사용하여 테스트되었습니다. 다른 버전을 사용할 경우 그에 맞게 단계를 적용해야 할 수 있습니다.

1. F12를 눌러 Developer Tools(개발자 도구) 콘솔을 시작합니다.
2. Network(네트워크) 탭을 선택한 다음 Developer Tools(개발자 도구) 창의 왼쪽 상단에서 Preserve log(로그 보존)을 선택합니다.
3. 문제를 재현합니다.
4. (선택 사항) Developer Tools(개발자 도구) Network(네트워크) 로그 창에 Method(메서드) 열이 보이지 않는 경우 아무 열 레이블에서 마우스 오른쪽 버튼으로 클릭하고 Method(메서드)를 선택하여 열을 추가할 수 있습니다.

5. Developer Tools(개발자 도구) Network(네트워크) 로그 창에서 SAML Post(SAML 게시물)을 찾습니다. 해당 행을 선택하고 상단에서 Payload(페이로드) 탭을 봅니다. 인코딩된 요청을 포함하는 SAMLResponse 요소를 확인합니다. 연결된 값은 Base64 인코딩 응답입니다.

Mozilla Firefox

Firefox에서 SAML 응답을 보려면

이 절차는 Mozilla Firefox 버전 105.0.3(64-bit)에서 테스트했습니다. 다른 버전을 사용할 경우 그에 맞게 단계를 적용해야 할 수 있습니다.

1. F12를 눌러 Web Developer Tools(웹 개발자 도구) 콘솔을 시작합니다.
2. 네트워크 탭을 선택합니다.
3. Web Developer Tools(웹 개발자 도구) 창 상단 오른쪽에서 옵션(작은 기어 모양 아이콘)을 클릭합니다. Persist logs(로그 유지)를 선택합니다.
4. 문제를 재현합니다.
5. (선택 사항) Web Developer Tools(웹 개발자 도구) Network(네트워크) 로그 창에 Method(메서드) 열이 보이지 않는 경우 아무 열 레이블에서 마우스 오른쪽 버튼으로 클릭하고 Method(메서드)를 선택하여 열을 추가할 수 있습니다.
6. 테이블에서 POST SAML을 확인합니다. 해당 행을 선택한 다음 Request(요청) 탭을 보고 SAMLResponse 요소를 찾습니다. 연결된 값은 Base64 인코딩 응답입니다.

Apple Safari

Safari에서 SAML 응답을 보려면

이 단계는 Apple Safari 버전 16.0(17614.1.25.9.10, 17614)을 사용하여 테스트되었습니다. 다른 버전을 사용할 경우 그에 맞게 단계를 적용해야 할 수 있습니다.

1. Safari에서 Web Inspector를 사용하도록 설정합니다. 기본 설정 창을 열고 고급 탭을 선택한 후 메뉴 표시줄에 Develop 메뉴 표시(Show Develop menu in the menu bar)를 선택합니다.
2. 이제 Web Inspector를 열 수 있습니다. 메뉴 막대에서 Develop(개발)을 선택한 다음 Show Web Inspector(Web Inspector 표시)를 선택합니다.
3. 네트워크 탭을 선택합니다.
4. Web Inspector 창의 왼쪽 상단에서 옵션(세 개의 수평선이 포함된 작은 원형 아이콘)을 선택합니다. Preserve Log(로그 보존)을 선택합니다.

5. (선택 사항) Web Inspector Network(네트워크) 로그 창에 Method(메서드) 열이 보이지 않는 경우 아무 열 레이블에서 마우스 오른쪽 버튼으로 클릭하고 Method(메서드)를 선택하여 열을 추가할 수 있습니다.
6. 문제를 재현합니다.
7. 테이블에서 POST SAML을 확인합니다. 해당 행을 선택하고 헤더(Headers) 탭을 봅니다.
8. 인코딩된 요청을 포함하는 SAMLResponse 요소를 확인합니다. 아래로 스크롤하여 `Request Data`라는 `SAMLResponse`를 확인합니다. 연결된 값은 Base64 인코딩 응답입니다.

Base64 인코딩 SAML 응답에 대해 해야 할 작업

브라우저에서 Base64 인코딩 SAML 응답 요소를 확인했으면 복사한 후 Base-64 디코딩 도구에서 사용하여 XML 태그 응답을 추출합니다.

> **보안 팁**
>
> 표시되는 SAML 응답 데이터에는 중요한 보안 데이터가 포함되어 있을 수 있으므로 온라인 base64 디코더를 사용하지 않을 것을 권장합니다. 대신 로컬 컴퓨터에 설치된 도구를 사용하십시오. 로컬 컴퓨터에 설치된 도구는 네트워크를 통해 SAML 데이터를 전송하지 않습니다.

Windows 시스템용 내장 옵션(PowerShell):

```
PS C:
\> [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("base64encodedtext
```

MacOS 및 Linux 시스템용 내장 옵션:

```
$ echo "base64encodedtext" | base64 --decode
```

디코딩된 파일의 값 검토

디코딩된 SAML 응답 파일의 값을 검토합니다.

- saml:NameID 속성 값이 인증된 사용자의 사용자 이름과 일치하는지 확인합니다.
- https://aws.amazon.com/SAML/Attributes/Role 값을 검토합니다. ARN과 SAML 공급자는 대소문자를 구분하며 ARN은 계정의 리소스와 일치해야 합니다.
- https://aws.amazon.com/SAML/Attributes/RoleSessionName 값을 검토합니다. 값은 클레임 규칙에 있는 값과 일치해야 합니다.

- 이메일 주소 또는 계정 이름의 속성 값을 구성하는 경우 값이 올바른지 확인합니다. 값은 인증된 사용자의 이메일 주소 또는 계정 이름과 일치해야 합니다.

오류 확인 및 구성 확인

값에 오류가 있는지 확인하고 다음 구성이 올바른지 확인합니다.

- 클레임 규칙이 필수 요소를 충족하고 모든 ARN이 정확합니다. 자세한 내용은 신뢰 당사자 신뢰 및 클레임 추가를 통해 SAML 2.0 IdP 구성 단원을 참조하십시오.
- SAML 공급자에서 IdP의 최신 메타데이터 파일을 AWS에 업로드했습니다. 자세한 내용은 SAML 2.0 페더레이션 사용자가 AWS Management Console에 액세스할 수 있게 하기 단원을 참조하십시오.
- IAM 역할의 신뢰 정책을 올바르게 구성했습니다. 자세한 내용은 역할 수임 방법 단원을 참조하십시오.

IAM의 임시 보안 자격 증명

AWS Security Token Service(AWS STS)를 사용하면 AWS 리소스에 대한 액세스를 제어할 수 있는 임시 보안 자격 증명을 생성하여 신뢰받는 사용자에게 제공할 수 있습니다. 임시 보안 인증은 다음과 같은 차이점을 제외하고는 장기 액세스 키 보안 인증과 거의 동일한 효력을 지닙니다.

- 임시 보안 자격 증명은 그 이름이 암시하듯 단기적입니다. 이 자격 증명은 몇 분에서 몇 시간까지 지속되도록 구성할 수 있습니다. 자격 증명이 만료된 후 AWS는 더는 그 자격 증명을 인식하지 못하거나 그 자격 증명을 사용한 API 요청으로부터 이루어지는 어떤 종류의 액세스도 허용하지 않습니다.
- 임시 보안 자격 증명은 사용자와 함께 저장되지 않지만 동적으로 생성되어 요청시 사용자에게 제공됩니다. 임시 보안 자격 증명이 만료되었을 때(심지어는 만료 전이라도) 사용자는 새 자격 증명을 요청할 수 있습니다. 단, 자격 증명을 요청하는 해당 사용자에게 그렇게 할 수 있는 권한이 있어야 합니다.

따라서 임시 보안 인증은 장기 보안 인증보다 다음과 같은 이점이 있습니다.

- 애플리케이션으로 장기 AWS 보안 자격 증명을 배포 또는 포함할 필요가 없습니다.
- 사용자에 대한 AWS 자격 증명을 정의하지 않고도 AWS 리소스에 대한 액세스 권한을 사용자에게 제공할 수 있습니다. 임시 자격 증명은 역할 및 ID 페더레이션의 기반입니다.
- 임시 보안 인증은 수명이 제한되어 있어서, 더 이상 필요하지 않을 때 업데이트하거나 명시적으로 취소할 필요가 없습니다. 임시 보안 자격 증명이 만료된 후에는 다시 사용할 수 없습니다. 그 자격 증명에 대해 유효 기간을 최대 한계까지 지정할 수 있습니다.

AWS STS 및 AWS 리전

임시 보안 자격 증명은 AWS STS에 의해 생성됩니다. 기본적으로 AWS STS는 `https://sts.amazonaws.com`에 단일 엔드포인트가 있는 전역적 서비스입니다. 그러나 지원되는 기타 다른 리전에서 엔드포인트에 대한 AWS STS API 호출을 할 수도 있습니다. 이렇게 지리적으로 더 가까운 리전에 있는 서버로 요청을 전송함으로써 지연 시간(서버 랙)을 단축할 수 있습니다. 자격 증명은 어떤 리전에서 오는지 상관없이 전역적으로 유효합니다. 자세한 내용은 AWS STS에서 AWS 리전 관리 단원을 참조하십시오.

임시 자격 증명과 관련된 일반적인 시나리오

임시 자격 증명은 아이덴티티 페더레이션, 위임, 크로스 계정 액세스, IAM 역할 등의 시나리오에서 유용합니다.

아이덴티티 페더레이션

AWS 밖의 외부 시스템에서 사용자 자격 증명을 관리할 수 있고 해당 시스템을 통해 로그인하는 사용자에게 액세스 권한을 부여하여 AWS 작업을 수행하고 AWS 리소스에 액세스할 수 있습니다. IAM은 두 가지 아이덴티티 페더레이션 유형을 지원합니다. 두 경우 모두 자격 증명은 AWS 외부에 저장됩니다. 외부 시스템이 상주하는 위치, 즉 데이터 센터 또는 웹의 외부 타사에 따라 구분됩니다. 외부 자격 증명 공급자에 대한 자세한 내용은 자격 증명 공급자 및 페더레이션 섹션을 참조하세요.

- SAML 페더레이션 - 조직의 네트워크에서 사용자를 인증한 다음, 해당 사용자에게 새로운 AWS ID를 만들거나 다른 로그인 자격 증명을 사용하여 로그인하도록 요구하지 않고도 해당 사용자에게 AWS에 대한 액세스 권한을 제공할 수 있습니다. 이는 임시 액세스 권한에 대한 Single Sign-On 접근 방식으로 알려져 있습니다. AWS STS는 SAML 2.0(Security Assertion Markup Language 2.0)과 같은 개방형 표준을 지원합니다. 이를 통해 Microsoft AD FS를 사용해 Microsoft Active Directory를 최대한 활용할 수 있습니다. 또한, SAML 2.0을 사용해 사용자 자격 증명 연동을 위한 자신만의 솔루션을 관리할 수 있습니다. 자세한 내용은 SAML 2.0 연동 단원을 참조하십시오.
  - 사용자 지정 페더레이션 브로커 - 조직의 인증 시스템을 사용해 AWS 리소스에 대한 액세스 권한을 부여할 수 있습니다. 시나리오 예시는 사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기 섹션을 참조하세요.
  - SAML 2.0을 사용한 페더레이션 - 조직의 인증 시스템과 SAML을 사용해 AWS 리소스에 대한 액세스 권한을 부여할 수 있습니다. 자세한 내용과 시나리오 예시는 SAML 2.0 연동 섹션을 참조하세요.
- OIDC(OpenID Connect) 페더레이션 - 사용자가 Amazon, Facebook 또는 Google로 로그인 또는 모바일 또는 웹 애플리케이션용 OIDC 2.0 호환 공급자와 같은 잘 알려진 타사 ID 공급자를 사용하여 로그인하도록 할 수 있으므로 사용자 지정 로그인 코드를 만들거나 자체 사용자 ID를 관리할 필요가

없습니다. OIDC 페더레이션을 사용하면 IAM 사용자 액세스 키와 같은 장기 보안 자격 증명을 애플리케이션에 배포할 필요가 없으므로 AWS 계정의 보안을 유지하는 데 도움이 됩니다. 자세한 내용은 OIDC 페더레이션 단원을 참조하십시오.

AWS STS OIDC 페더레이션은 Amazon, Facebook 및 Google로 로그인과 모든 OIDC(OpenID Connect) 호환 ID 공급자를 지원합니다.

> **Note**
>
> 모바일 애플리케이션의 경우 Amazon Cognito를 사용하는 것이 좋습니다. 이 서비스와 함께 모바일 개발용 AWS SDK를 사용하면 사용자의 고유 자격 증명을 생성하고 인증하여 AWS 리소스에 안전하게 액세스하도록 할 수 있습니다. Amazon Cognito는 AWS STS와 동일한 자격 증명 공급자를 지원하고 인증되지 않은(게스트) 액세스도 지원하며 사용자가 로그인할 때 사용자 데이터를 마이그레이션할 수 있습니다. 또한, Amazon Cognito는 사용자가 디바이스 간에 이동할 때 데이터가 보존되도록 사용자 데이터를 동기화하기 위한 API 작업도 제공합니다. 자세한 내용은 Amplify 설명서의 Amplify를 사용한 인증을 참조하세요.

크로스 계정 액세스를 위한 역할

많은 조직이 1개 이상의 AWS 계정을 유지합니다. 역할 및 크로스 계정 액세스를 사용하면 하나의 계정에서 사용자 자격 증명을 정의하고 그 자격 증명을 사용해 조직에 속한 다른 계정의 AWS 리소스에 액세스할 수 있습니다. 이는 임시 액세스 권한에 대한 위임 접근 방식으로 알려져 있습니다. 크로스 계정 역할 생성에 대한 자세한 내용은 IAM 사용자에게 권한을 위임할 역할 생성 섹션을 참조하세요. 해당 신뢰 영역(신뢰할 수 있는 조직 또는 계정) 외의 계정 내 보안 주체가 역할을 수임하는 권한이 있는지 자세히 알고 싶다면, IAM Access Analyzer란 무엇일까요?를 참조하세요.

Amazon EC2의 역할

Amazon EC2 인스턴스에서 애플리케이션을 실행할 때 그 애플리케이션이 AWS 리소스에 대한 액세스 권한이 필요한 경우 인스턴스 시작 시 인스턴스에 대한 임시 보안 자격 증명을 제공할 수 있습니다. 이 임시 보안 자격 증명은 인스턴스에서 실행되는 모든 애플리케이션에서 사용 가능하므로 그 인스턴스에 어떤 장기 자격 증명도 저장할 필요가 없습니다. 자세한 내용은 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여 단원을 참조하십시오.

기타 AWS 서비스

임시 보안 보안 인증을 사용해 대부분의 AWS 서비스에 액세스할 수 있습니다. 임시 보안 자격 증명을 수락하는 서비스의 목록은 AWS IAM으로 작업하는 서비스 섹션을 참조하세요.

임시 보안 자격 증명 요청

임시 보안 자격 증명을 요청하려면 AWS API에서 AWS Security Token Service(AWS STS) 작업을 사용할 수 있습니다. 이러한 작업에는 AWS 리소스에 대한 액세스를 제어할 수 있는 임시 보안 자격 증명을 생성하여 신뢰할 수 있는 사용자에게 제공할 수 있습니다. 에 대한 자세한 내용은 AWS STS 섹션을 참조하세요.IAM의 임시 보안 자격 증명. 역할을 수임해 임시 보안 자격 증명을 요청하는 데 사용할 수 있는 여러 방법을 알아보려면 역할 수임 방법 섹션을 참조하세요.

API 작업을 호출하려면 AWS SDK 중 하나를 사용할 수 있습니다. SDK는 Java, .NET, Python, Ruby, Android 및 iOS 등 다양한 프로그래밍 언어 및 환경에서 사용할 수 있습니다. SDK는 요청에 암호화 방식으로 서명, 필요한 경우 요청 재시도, 오류 응답 처리와 같은 작업들을 다룹니다. AWS Security Token Service API 참조에 설명된 AWS STS 쿼리 API를 사용할 수도 있습니다. 마지막으로, AWS Command Line Interface 및 AWS Tools for Windows PowerShell의 두 가지 명령 줄 도구에서 AWS STS 명령을 지원합니다.

AWS STS API 작업은 액세스 키 페어 및 세션 토큰을 포함하는 임시 보안 자격 증명을 사용하여 새 세션을 생성합니다. 액세스 키 페어는 액세스 키 ID와 보안 키로 구성되어 있습니다. 사용자(또는 사용자가 실행하는 애플리케이션)는 이 자격 증명을 사용해 리소스에 액세스할 수 있습니다. AWS STS API 작업을 사용하여 프로그래밍 방식으로 역할 세션을 생성하고 세션 정책 및 세션 태그를 전달할 수 있습니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 자격 증명 기반 정책의 교집합과 세션 정책입니다. 세션 정책에 대한 자세한 정보는 세션 정책 섹션을 참조하세요. 세션 태그에 대한 자세한 내용은 AWS STS에서 세션 태그 전달 섹션을 참조하세요.

> **Note**
>
> AWS STS API 작업이 반환하는 세션 토큰의 크기는 고정적이지 않습니다. 따라서 최대 크기를 가정하지 않는 것이 좋습니다. 일반적인 토큰 크기는 4096바이트 미만이나 경우에 따라 다를 수 있습니다.

AWS 리전에서 AWS STS 사용하기

전역 엔드포인트 또는 리전 엔드포인트 중 하나에 AWS STS API 호출을 전송할 수 있습니다. 더 가까이 있는 엔드포인트를 선택하면 지연 시간을 단축해 API 호출의 성능을 향상시킬 수 있습니다. 또한 원래 엔드포인트와 더 이상 통신하지 할 수 없는 경우 대체 리전 엔드포인트에 호출을 직접 보내는 방법을 선택할 수 있습니다. 다양한 AWS SDK 중 하나를 사용하고 있다면 API 호출 전에 해당 SDK 방법을 사용해 리전을 지정합니다. HTTP API 요청을 수동으로 구축하는 경우 그 요청을 정확한 엔드포인트에

직접 전송해야 합니다. 자세한 내용은 리전 및 엔드포인트의 AWS STS 섹션 및 AWS STS에서 AWS 리전 관리 섹션을 참조하세요.

다음은 AWS 환경 및 애플리케이션에서 사용할 임시 자격 증명을 획득하는 데 사용할 수 있는 API 작업입니다.

AssumeRole - 사용자 지정 아이덴티티 브로커를 통한 크로스 계정 위임과 페더레이션

`AssumeRole` API 작업은 기존 IAM 사용자가 아직 액세스 권한이 없는 AWS 리소스에 액세스할 수 있도록 허용하는 데 유용합니다. 예를 들어 사용자가 다른 AWS 계정의 리소스에 액세스해야 할 수 있습니다. 또한 멀티 팩터 인증(MFA)을 제공하는 것과 같이 일시적으로 액세스 권한을 얻는 수단으로도 유용합니다. 활성 자격 증명을 사용해 이 API를 호출해야 합니다. 이 작업을 호출할 수 있는 사용자를 알아보려면 AWS STS API 작업 비교을(를) 참조하세요. 자세한 내용은 IAM 사용자에게 권한을 위임할 역할 생성 및 MFA를 통한 보안 API 액세스을(를) 참조하세요.

이 호출에는 반드시 유효한 AWS 보안 자격 증명을 사용해야 합니다. 이 호출을 할 때 다음과 같은 정보를 전달하게 됩니다.

- 앱이 수임해야 하는 역할의 Amazon 리소스 이름(ARN)
- (선택 사항) 기간. 임시 보안 자격 증명의 기간을 지정합니다. `DurationSeconds` 파라미터를 사용하여 역할 세션 기간을 900초(15분)에서 해당 역할에 대한 최대 세션 기간 설정까지 지정합니다. 역할에 대한 최댓값을 확인하는 방법을 알아보려면 역할의 최대 세션 기간 업데이트 섹션을 참조하세요. 이 파라미터를 전달하지 않으면 임시 자격 증명이 한 시간 내에 만료됩니다. 이 API의 `DurationSeconds` 파라미터는 콘솔 세션의 기간을 지정하는 데 사용하는 `SessionDuration` HTTP 파라미터와 다릅니다. 콘솔 로그인 토큰의 연동 엔드포인트에 대한 요청에는 `SessionDuration` HTTP 파라미터를 사용하세요. 자세한 내용은 사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기 단원을 참조하십시오.
- 역할 세션 이름. 여러 보안 주체가 하나의 역할을 사용하는 경우 이 문자열 값을 사용하여 세션을 식별합니다. 보안을 위해 관리자는 AWS CloudTrail 로그의 이 필드를 검토하여 AWS에서 누가 작업을 수행했는지 확인할 수 있습니다. 관리자는 사용자가 역할을 수임할 때 IAM 사용자 이름을 세션 이름으로 지정하도록 요구할 수 있습니다. 자세한 내용은 `sts:RoleSessionName` 단원을 참조하십시오.
- (선택 사항) 소스 자격 증명. 사용자가 역할을 수임할 때 소스 자격 증명을 지정하도록 요구할 수 있습니다. 소스 자격 증명은 설정한 후에는 값을 변경할 수 없으며 역할 세션 중에 수행되는 모든 작업의 요청에 있습니다. 소스 자격 증명 값은 역할 체인 세션 간에 유지됩니다. AWS CloudTrail 로그의 소스 자격 증명 정보를 사용하여 역할로 작업을 수행한 사용자를 확인할 수 있습니다. 소스 자격 증명 사용에 대한 자세한 내용은 위임된 역할로 수행한 작업 모니터링 및 제어 섹션을 참조하세요.

- (선택 사항) 인라인 또는 관리형 세션 정책. 이러한 정책은 역할 세션에 할당된 역할 자격 증명 기반 정책의 권한을 제한합니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 자격 증명 기반 정책의 교집합과 세션 정책입니다. 세션 정책을 사용하여 수임된 역할의 자격 증명 기반 정책에서 허용되는 것보다 더 많은 권한을 부여할 수는 없습니다. 역할 세션 권한에 대한 자세한 정보는 세션 정책 섹션을 참조하세요.
- (선택 사항) 세션 태그. 역할을 맡은 다음 임시 자격 증명을 사용하여 요청할 수 있습니다. 이렇게 하면 세션의 보안 주체 태그에 역할의 태그와 전달된 세션 태그가 포함됩니다. 임시 자격 증명을 사용하여 이렇게 호출하는 경우 새 세션은 호출 세션의 전이적 세션 태그도 상속합니다. 세션 태그에 대한 자세한 내용은 AWS STS에서 세션 태그 전달 섹션을 참조하세요.
- (선택 사항) MFA 정보. MFA(멀티 팩터 인증)를 사용하도록 구성한 경우, MFA 디바이스의 식별자와 해당 디바이스에서 제공한 일회용 코드를 포함시켜야 합니다.
- (선택 사항) 계정에 대한 액세스 권한을 타사에 위임할 때 사용할 수 있는 `ExternalId` 값입니다. 이 값은 지정된 타사만 역할에 액세스할 수 있도록 하는 데 도움이 됩니다. 자세한 내용은 타사가 소유한 AWS 계정에 액세스 단원을 참조하십시오.

다음 예제에서는 `AssumeRole`을 사용한 샘플 요청 및 응답을 보여줍니다. 이 예제 요청에서는 지정된 기간 동안 세션 정책, 세션 태그, 외부 ID 및소스 자격 증명이 포함된 demo 역할을 가정합니다. 결과 세션의 이름이 `John-session`으로 지정됩니다.

Example 요청 예제

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=AssumeRole
&RoleSessionName=John-session
&RoleArn=arn:aws::iam::123456789012:role/demo
&Policy=%7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Sid%22%3A
%20%22Stmt1%22%2C%22Effect%22%3A%20%22Allow%22%2C%22Action%22%3A%20%22s3%3A*%22%2C
%22Resource%22%3A%20%22*%22%7D%5D%7D
&DurationSeconds=1800
&Tags.member.1.Key=Project
&Tags.member.1.Value=Pegasus
&Tags.member.2.Key=Cost-Center
&Tags.member.2.Value=12345
&ExternalId=123ABC
&SourceIdentity=DevUser123
&AUTHPARAMS
```

이전 예제의 정책 값은 다음 정책을 URL로 인코딩한 버전입니다.

```
{"Version":"2012-10-17","Statement":
[{"Sid":"Stmt1","Effect":"Allow","Action":"s3:*","Resource":"*"}]}
```

예시의 AUTHPARAMS 파라미터는 서명에 대한 자리 표시자입니다. 서명은 AWS HTTP API 요청에 포함해야 하는 인증 정보입니다. AWS SDK를 사용하여 API 요청을 생성하는 것이 좋습니다. 이렇게 하면 SDK가 요청 서명을 대신 처리한다는 장점이 있습니다. API 요청을 수동으로 생성하고 서명해야 하는 경우 요청에 서명하는 방법을 알아보려면 Amazon Web Services 일반 참조의 서명된 AWS API 요청 생성을 참조하세요.

그 응답에는 임시 보안 자격 증명뿐만 아니라 페더레이션 사용자 및 자격 증명 만료 시간에 대한 Amazon 리소스 이름(ARN)이 포함되어 있습니다.

Example 응답의 예

```
<AssumeRoleResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
<AssumeRoleResult>
<SourceIdentity>DevUser123</SourceIdentity>
<Credentials>
  <SessionToken>
   AQoDYXdzEPT//////////wEXAMPLEtc764bNrC9SAPBSM22wDOk4x4HIZ8j4FZTwdQW
   LWsKWHGBuFqwAeMicRXmxfpSPfIeoIYRqTflfKD8YUuwthAx7mSEI/qkPpKPi/kMcGd
   QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPkyQDYwT7WZ0wq5VSXDvp75YU
   9HFvlRd8Tx6q6fE8YQcHNVXAkiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
   +scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSYvKTr4rvx3iSIlTJabIQwj2ICCR/oLxBA==
  </SessionToken>
  <SecretAccessKey>
   wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY
  </SecretAccessKey>
  <Expiration>2019-07-15T23:28:33.359Z</Expiration>
  <AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
</Credentials>
<AssumedRoleUser>
  <Arn>arn:aws:sts::123456789012:assumed-role/demo/John</Arn>
  <AssumedRoleId>ARO123EXAMPLE123:John</AssumedRoleId>
</AssumedRoleUser>
<PackedPolicySize>8</PackedPolicySize>
</AssumeRoleResult>
<ResponseMetadata>
<RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
</ResponseMetadata>
</AssumeRoleResponse>
```

> **Note**
>
> AWS 변환은 전달된 세션 정책과 세션 태그를 별도의 제한이 있는 압축된 이진 형식으로 압축합니다. 일반 텍스트가 다른 요구 사항을 충족하는 경우에도 이 제한으로 인해 요청이 실패할 수 있습니다. `PackedPolicySize` 응답 요소는 요청에 대한 정책 및 태그가 상위 크기 제한과 얼마나 가까운지를 백분율로 나타냅니다.

AssumeRoleWithWebIdentity - 웹 기반 ID 제공자를 통한 페더레이션

`AssumeRoleWithWebIdentity` API 작업은 퍼블릭 자격 증명 공급자를 통해 인증된 페더레이션 사용자의 임시 보안 자격 증명 세트를 반환합니다. 퍼블릭 보안 인증 공급자의 예에는 Login with Amazon, Facebook, Google 또는 OpenID Connect(OIDC)호환 보안 인증 공급자 등이 있습니다. 이 작업은 AWS에 대한 액세스가 필요한 모바일 애플리케이션 또는 클라이언트 기반 애플리케이션을 생성하는 데 유용합니다. 이 작업을 사용하는 경우 사용자가 자체 AWS 또는 IAM 자격 증명을 필요로 하지 않습니다. 자세한 내용은 OIDC 페더레이션 단원을 참조하십시오.

`AssumeRoleWithWebIdentity`를 직접 호출하는 대신 모바일 개발을 위한 AWS SDK에서 Amazon Cognito 및 Amazon Cognito 자격 증명 공급자를 사용할 것을 권장합니다. 자세한 내용은 Amplify 설명서의 Amplify를 사용한 인증을 참조하세요.

Amazon Cognito를 사용하고 있지 않다면 `AssumeRoleWithWebIdentity`의 AWS STS 작업을 호출합니다. 이것은 서명되지 않은 호출로서 앱이 이 호출을 하기 위해 어떤 AWS 보안 자격 증명에도 액세스할 필요가 없음을 뜻합니다. 이 호출을 할 때 다음과 같은 정보를 전달하게 됩니다.

- 앱이 수임해야 하는 역할의 Amazon 리소스 이름(ARN) 앱이 사용자가 로그인하는 여러 가지 방식을 지원하는 경우 다양한 역할, 즉 자격 증명 공급자당 하나의 역할을 정의해야 합니다. `AssumeRoleWithWebIdentity`에 대한 호출에는 사용자가 로그인할 때 사용한 공급자에 특정된 역할의 ARN이 포함되어야 합니다.
- 앱이 사용자를 인증한 후에 IdP로부터 얻는 토큰
- 속성을 토큰에 세션 태그로 전달하도록 IdP를 구성할 수 있습니다.
- (선택 사항) 기간. 임시 보안 자격 증명의 기간을 지정합니다. `DurationSeconds` 파라미터를 사용하여 역할 세션 기간을 900초(15분)에서 해당 역할에 대한 최대 세션 기간 설정까지 지정합니다. 역할에 대한 최댓값을 확인하는 방법을 알아보려면 역할의 최대 세션 기간 업데이트 섹션을 참조하세요. 이 파라미터를 전달하지 않으면 임시 자격 증명이 한 시간 내에 만료됩니다. 이 API의 `DurationSeconds` 파라미터는 콘솔 세션의 기간을 지정하는 데 사용하는 `SessionDuration` HTTP 파라미터와 다릅니다. 콘솔 로그인 토큰의 연동 엔드포인트에 대한 요청

에는 SessionDuration HTTP 파라미터를 사용하세요. 자세한 내용은 사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기 단원을 참조하십시오.

- 역할 세션 이름. 여러 보안 주체가 하나의 역할을 사용하는 경우 이 문자열 값을 사용하여 세션을 식별합니다. 보안을 위해 관리자는 AWS CloudTrail 로그의 이 필드를 검토하여 AWS에서 누가 작업을 수행했는지 확인할 수 있습니다. 관리자는 사용자가 역할을 수임할 때 세션 이름에 특정 값을 지정하도록 요구할 수 있습니다. 자세한 내용은 sts:RoleSessionName 단원을 참조하십시오.
- (선택 사항) 소스 자격 증명. 페더레이션 사용자가 역할을 수임할 때 소스 자격 증명을 지정하도록 요구할 수 있습니다. 소스 자격 증명은 설정한 후에는 값을 변경할 수 없으며 역할 세션 중에 수행되는 모든 작업의 요청에 있습니다. 소스 자격 증명 값은 역할 체인 세션 간에 유지됩니다. AWS CloudTrail 로그의 소스 자격 증명 정보를 사용하여 역할로 작업을 수행한 사용자를 확인할 수 있습니다. 소스 자격 증명 사용에 대한 자세한 내용은 위임된 역할로 수행한 작업 모니터링 및 제어 섹션을 참조하세요.
- (선택 사항) 인라인 또는 관리형 세션 정책. 이러한 정책은 역할 세션에 할당된 역할 자격 증명 기반 정책의 권한을 제한합니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 자격 증명 기반 정책의 교집합과 세션 정책입니다. 세션 정책을 사용하여 수임된 역할의 자격 증명 기반 정책에서 허용되는 것보다 더 많은 권한을 부여할 수는 없습니다. 역할 세션 권한에 대한 자세한 정보는 세션 정책 섹션을 참조하세요.

> **Note**
>
> AssumeRoleWithWebIdentity에 대한 호출이 서명(암호화)되지 않았습니다. 따라서 요청이 신뢰할 수 있는 중개자를 통해 전송된 경우에만 선택적 세션 정책을 포함해야 합니다. 이러한 경우 누군가가 정책을 변경해 제한을 제거할 수 있습니다.

AssumeRoleWithWebIdentity를 호출하면 AWS가 토큰의 신뢰성을 확인합니다. 예를 들어 공급자에 따라 AWS는 해당 공급자를 호출해 앱이 전달한 토큰을 포함할 수 있습니다. 자격 증명 공급자가 토큰을 확인한다고 가정하면, AWS는 다음 정보를 반환합니다.

- 일련의 임시 보안 자격 증명 이러한 임시 보안 자격 증명은 액세스 키 ID, 보안 액세스 키 및 세션 토큰으로 이루어져 있습니다.
- 위임된 역할의 역할 ID 및 ARN
- 고유한 사용자 ID를 포함하는 SubjectFromWebIdentityToken 값

임시 보안 자격 증명이 있으면 AWS API 호출에 사용할 수 있습니다. 이는 장기 보안 자격 증명을 사용한 AWS API 호출과 동일한 프로세스입니다. 차이점은 AWS에서 임시 보안 자격 증명이 유효한지 확인하도록 하는 세션 토큰을 포함해야 한다는 점입니다.

앱은 자격 증명을 캐싱해야 합니다. 언급한 바와 같이, 자격 증명은 한 시간 후에 만료되도록 기본 설정되어 있습니다. AWS SDK에서 AmazonSTSCredentialsProvider 작업을 사용하지 않는 경우 사용자 및 사용자 앱에서 `AssumeRoleWithWebIdentity`을(를) 다시 호출해야 합니다. 이전 자격 증명이 만료되기 전에 이 작업을 호출하여 임시 보안 자격 증명 세트를 새로 받으세요.

AssumeRoleWithSAML - SAML 2.0과 호환되는 엔터프라이즈 ID 제공자를 통한 페더레이션

`AssumeRoleWithSAML` API 작업은 조직의 기존 자격 증명 시스템을 통해 인증된 페더레이션 사용자의 임시 보안 자격 증명 세트를 반환합니다. 또한 사용자는 SAML 2.0(Security Assertion Markup Language)을 사용하여 AWS에 인증 및 권한 부여 정보를 전달해야 합니다. 이 API 작업은 자격 증명 시스템(예: Windows Active Directory 또는 OpenLDAP)을 SAML 어설션을 생성할 수 있는 소프트웨어와 통합한 조직에 유용합니다. 이러한 통합은 사용자 자격 증명 및 권한에 대한 정보를 제공합니다(예: Active Directory Federation Services 또는 Shibboleth). 자세한 내용은 SAML 2.0 연동 단원을 참조하십시오.

> **Note**
>
> `AssumeRoleWithSAML`에 대한 호출이 서명(암호화)되지 않았습니다. 따라서 요청이 신뢰할 수 있는 중개자를 통해 전송된 경우에만 선택적 세션 정책을 포함해야 합니다. 이러한 경우 누군가가 정책을 변경해 제한을 제거할 수 있습니다.

이것은 서명되지 않은 호출로서 앱이 이 호출을 하기 위해 어떤 AWS 보안 자격 증명에도 액세스할 필요가 없음을 뜻합니다. 이 호출을 할 때 다음과 같은 정보를 전달하게 됩니다.

- 앱이 수임해야 하는 역할의 Amazon 리소스 이름(ARN)
- SAML 공급자의 ARN(자격 증명 공급자에 대해 설명하는 IAM에서 생성됨)
- 앱의 로그인 요청에 대한 인증 응답 시 SAML 자격 증명 공급자가 제공한 base-64 인코딩 SAML 어설션
- 속성을 SAML 어설션에 세션 태그로 전달하도록 IdP를 구성할 수 있습니다.
- (선택 사항) 기간. 임시 보안 자격 증명의 기간을 지정합니다. `DurationSeconds` 파라미터를 사용하여 역할 세션 기간을 900초(15분)에서 해당 역할에 대한 최대 세션 기간 설정까지 지

정합니다. 역할에 대한 최댓값을 확인하는 방법을 알아보려면 역할의 최대 세션 기간 업데이트 섹션을 참조하세요. 이 파라미터를 전달하지 않으면 임시 자격 증명이 한 시간 내에 만료됩니다. 이 API의 `DurationSeconds` 파라미터는 콘솔 세션의 기간을 지정하는 데 사용하는 `SessionDuration` HTTP 파라미터와 다릅니다. 콘솔 로그인 토큰의 연동 엔드포인트에 대한 요청에는 `SessionDuration` HTTP 파라미터를 사용하세요. 자세한 내용은 사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기 단원을 참조하십시오.

- (선택 사항) 인라인 또는 관리형 세션 정책. 이러한 정책은 역할 세션에 할당된 역할 자격 증명 기반 정책의 권한을 제한합니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 자격 증명 기반 정책의 교집합과 세션 정책입니다. 세션 정책을 사용하여 수임된 역할의 자격 증명 기반 정책에서 허용되는 것보다 더 많은 권한을 부여할 수는 없습니다. 역할 세션 권한에 대한 자세한 정보는 세션 정책 섹션을 참조하세요.
- 역할 세션 이름. 여러 보안 주체가 하나의 역할을 사용하는 경우 이 문자열 값을 사용하여 세션을 식별합니다. 보안을 위해 관리자는 AWS CloudTrail 로그의 이 필드를 검토하여 AWS에서 누가 작업을 수행했는지 확인할 수 있습니다. 관리자는 사용자가 역할을 수임할 때 세션 이름에 특정 값을 지정하도록 요구할 수 있습니다. 자세한 내용은 `sts:RoleSessionName` 단원을 참조하십시오.
- (선택 사항) 소스 자격 증명. 페더레이션 사용자가 역할을 수임할 때 소스 자격 증명을 지정하도록 요구할 수 있습니다. 소스 자격 증명은 설정한 후에는 값을 변경할 수 없으며 역할 세션 중에 수행되는 모든 작업의 요청에 있습니다. 소스 자격 증명 값은 역할 체인 세션 간에 유지됩니다. AWS CloudTrail 로그의 소스 자격 증명 정보를 사용하여 역할로 작업을 수행한 사용자를 확인할 수 있습니다. 소스 자격 증명 사용에 대한 자세한 내용은 위임된 역할로 수행한 작업 모니터링 및 제어 섹션을 참조하세요.

`AssumeRoleWithSAML`을 호출하면 AWS가 SAML 어설션의 신뢰성을 확인합니다. 자격 증명 공급자가 어설션을 확인한다고 가정하면, AWS는 다음 정보를 반환합니다.

- 일련의 임시 보안 자격 증명 이러한 임시 보안 자격 증명은 액세스 키 ID, 보안 액세스 키 및 세션 토큰으로 이루어져 있습니다.
- 위임된 역할의 역할 ID 및 ARN
- SAML 어설션의 `Audience` 요소의 `Recipient` 속성 값을 포함하는 `SubjectConfirmationData` 값
- SAML 어설션의 `Issuer` 요소 값을 포함하는 `Issuer` 값
- `Issuer` 값, AWS 계정 ID, SAML 공급자의 표시 이름으로 구축된 해시 값을 포함하는 `NameQualifier` 요소 `Subject` 요소와 결합되면 페더레이션 사용자를 고유한 이름으로 식별할 수 있습니다.
- SAML 어설션의 `Subject` 요소에 있는 `NameID` 요소의 값을 포함하는 `Subject` 요소

- SubjectType 요소의 형식을 나타내는 Subject 요소 그 값은 persistent, transient, 또는 SAML 어설션에서 사용되는 Format 및 Subject 요소의 전체 NameID URI일 수 있습니다. NameID 요소의 Format 속성에 대한 자세한 내용은 인증 응답에 대한 SAML 어설션 구성 섹션을 참조하세요.

임시 보안 자격 증명이 있으면 AWS API 호출에 사용할 수 있습니다. 이는 장기 보안 자격 증명을 사용한 AWS API 호출과 동일한 프로세스입니다. 차이점은 AWS에서 임시 보안 자격 증명이 유효한지 확인하도록 하는 세션 토큰을 포함해야 한다는 점입니다.

앱은 자격 증명을 캐싱해야 합니다. 자격 증명은 한 시간 후에 만료되도록 기본 설정되어 있습니다. AWS SDK에서 AmazonSTSCredentialsProvider 작업을 사용하지 않는 경우 사용자 및 사용자 앱에서 AssumeRoleWithSAML을 다시 호출해야 합니다. 이전 자격 증명이 만료되기 전에 이 작업을 호출하여 임시 보안 자격 증명 세트를 새로 받으세요.

GetFederationToken - 사용자 지정 아이덴티티 브로커를 통한 페더레이션

GetFederationToken API 작업은 페더레이션 사용자에게 일련의 임시 보안 자격 증명을 반환합니다. 이 API는 기본 만료 기간이 상당히 길다는 점이(1시간이 아니라12시간) AssumeRole과 다릅니다. 또한 DurationSeconds 파라미터를 사용하여 임시 보안 자격 증명이 유효하게 남아 있을 기간을 지정할 수 있습니다. 결과 보안 인증 정보는 900초(15분)~129,600초(36시간)의 지정된 기간 동안 유효합니다. 만료 기간이 길어지면 새 보안 인증 정보를 자주 받을 필요가 없으므로 AWS에 대한 호출 수를 줄이는 데 도움이 될 수 있습니다.

이 요청을 할 때 특정 IAM 사용자의 자격 증명을 사용합니다. 임시 보안 자격 증명에 대한 권한은 GetFederationToken을 호출할 때 전달하는 세션 정책에 의해 결정됩니다. 결과적으로 얻는 세션의 권한은 IAM 사용자 정책 또는 전달한 세션 정책의 교집합입니다. 세션 정책은 페더레이션을 요청하는 IAM 사용자의 자격 증명 기반 정책에서 허용되는 권한보다 많은 권한을 부여하는 데 사용할 수 없습니다. 역할 세션 권한에 대한 자세한 정보는 세션 정책 섹션을 참조하세요.

GetFederationToken 작업에서 반환되는 임시 자격 증명을 사용하는 경우 세션의 보안 주체 태그에 사용자의 태그와 전달된 세션 태그가 포함됩니다. 세션 태그에 대한 자세한 내용은 AWS STS에서 세션 태그 전달 섹션을 참조하세요.

GetFederationToken 호출은 세션 토큰, 액세스 키, 보안 키, 만료로 구성된 임시 보안 자격 증명을 반환합니다. 조직 내에서 권한을 관리하고 싶다면 GetFederationToken을 사용할 수 있습니다(예: 프록시 애플리케이션을 사용할 권한 할당).

다음 예에서는 GetFederationToken을 사용한 샘플 요청 및 응답을 보여줍니다. 이 요청 예제에서는 지정된 기간 동안 호출 사용자를 세션 정책 ARN 및 세션 태그와 연동합니다. 결과 세션의 이름이 Jane-session으로 지정됩니다.

Example 요청 예제

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=GetFederationToken
&Name=Jane-session
&PolicyArns.member.1.arn==arn%3Aaws%3Aiam%3A%3A123456789012%3Apolicy%2FRole1policy
&DurationSeconds=1800
&Tags.member.1.Key=Project
&Tags.member.1.Value=Pegasus
&Tags.member.2.Key=Cost-Center
&Tags.member.2.Value=12345
&AUTHPARAMS
```

앞의 예시에서 표시된 정책 ARN에는 다음과 같은 URL 인코딩 ARN이 포함되어 있습니다.

```
arn:aws:iam::123456789012:policy/Role1policy
```

또한 이 예제의 &AUTHPARAMS 파라미터는 인증 정보의 자리 표시자로 사용됩니다. 이는 서명이며 AWS HTTP API 요청과 함께 포함되어야 합니다. AWS SDK를 사용하여 API 요청을 생성하는 것이 좋습니다. 이렇게 하면 SDK가 요청 서명을 대신 처리한다는 장점이 있습니다. API 요청을 수동으로 생성하고 서명해야 하는 경우 요청에 서명하는 방법을 알아보려면 Amazon Web Services 일반 참조의 서명된 AWS API 요청 생성으로 이동하세요.

그 응답에는 임시 보안 자격 증명뿐만 아니라 페더레이션 사용자 및 자격 증명 만료 시간에 대한 Amazon 리소스 이름(ARN)이 포함되어 있습니다.

Example 응답의 예

```
<GetFederationTokenResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
<GetFederationTokenResult>
<Credentials>
  <SessionToken>
   AQoDYXdzEPT//////////wEXAMPLEtc764bNrC9SAPBSM22wDOk4x4HIZ8j4FZTwdQW
   LWsKWHGBuFqwAeMicRXmxfpSPfIeoIYRqTflfKD8YUuwthAx7mSEI/qkPpKPi/kMcGd
   QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPkyQDYwT7WZ0wq5VSXDvp75YU
   9HFvlRd8Tx6q6fE8YQcHNVXAkiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
```

```
    +scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSYvKTr4rvx3iSIlTJabIQwj2ICCEXAMPLE==
  </SessionToken>
  <SecretAccessKey>
  wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY
  </SecretAccessKey>
  <Expiration>2019-04-15T23:28:33.359Z</Expiration>
  <AccessKeyId>AKIAIOSFODNN7EXAMPLE;</AccessKeyId>
</Credentials>
<FederatedUser>
  <Arn>arn:aws:sts::123456789012:federated-user/Jean</Arn>
  <FederatedUserId>123456789012:Jean</FederatedUserId>
</FederatedUser>
<PackedPolicySize>4</PackedPolicySize>
</GetFederationTokenResult>
<ResponseMetadata>
<RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
</ResponseMetadata>
</GetFederationTokenResponse>
```

> **Note**
>
> AWS 변환은 전달된 세션 정책과 세션 태그를 별도의 제한이 있는 압축된 이진 형식으로 압축합니다. 일반 텍스트가 다른 요구 사항을 충족하는 경우에도 이 제한으로 인해 요청이 실패할 수 있습니다. `PackedPolicySize` 응답 요소는 요청에 대한 정책 및 태그가 상위 크기 제한과 얼마나 가까운지를 백분율로 나타냅니다.

AWS는 리소스 수준에서 권한을 부여할 것을 권장합니다(예: Amazon S3 버킷에 리소스 기반 정책 연결). `Policy` 파라미터는 생략할 수 있습니다. 그러나 페더레이션 사용자에 대한 정책을 포함하지 않으면, 임시 보안 자격 증명은 어떤 권한도 부여하지 않을 것입니다. 이 경우 반드시 리소스 정책을 사용해 페더레이션 사용자에게 AWS 리소스에 대한 액세스 권한을 부여해야 합니다.

예를 들어, 내 AWS 계정 번호가 111122223333이고 Susan이 액세스하도록 허용하려는 Amazon S3 버킷을 내가 가지고 있다고 가정해 보겠습니다. Susan의 임시 보안 자격 증명에는 버킷에 대한 정책은 포함되어 있지 않습니다. 이러한 경우 버킷에 Susan의 ARN과 일치하는 ARN과 관련된 정책이 있는지 확인해야 합니다(예: `arn:aws:sts::111122223333:federated-user/Susan`).

GetSessionToken - 신뢰할 수 없는 환경에 있는 사용자를 위한 임시 자격 증명

`GetSessionToken` API 작업은 기존 IAM 사용자에게 일련의 임시 보안 자격 증명을 반환합니다. 예를 들어 MFA가 IAM 사용자에 대해 활성화된 경우에만 AWS 요청을 허용하면 보안을 강화하는 데 유

용합니다. 자격 증명은 일시적이므로 덜 안전한 환경을 통해 리소스에 액세스하는 IAM 사용자가 있을 때 보안을 강화하는 역할을 합니다. 덜 안전한 환경의 예시로는 모바일 디바이스 또는 웹 브라우저가 있습니다. 자세한 내용은 AWS Security Token Service API 참조의 임시 보안 자격 증명 요청 또는 GetSessionToken을 참조하세요.

기본적으로 IAM 사용자에 대한 임시 보안 자격 증명은 최대 12시간 동안 유효합니다. 그러나 DurationSeconds 파라미터를 사용하여 이 기간을 15분만큼 짧게 또는 36시간만큼 길게 요청할 수 있습니다. 보안상의 이유로 AWS 계정 루트 사용자의 토큰은 1시간의 유효 기간으로 제한됩니다.

GetSessionToken은 세션 토큰, 액세스 키 ID 및 비밀 액세스 키로 구성된 임시 보안 자격 증명을 반환합니다. 다음 예제에서는 GetSessionToken을 사용한 샘플 요청 및 응답을 보여줍니다. 응답에는 임시 보안 자격 증명의 만료 시간도 포함되어 있습니다.

Example 요청 예제

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=GetSessionToken
&DurationSeconds=1800
&AUTHPARAMS
```

예시의 AUTHPARAMS 파라미터는 서명에 대한 자리 표시자입니다. 서명은 AWS HTTP API 요청에 포함해야 하는 인증 정보입니다. AWS SDK를 사용하여 API 요청을 생성하는 것이 좋습니다. 이렇게 하면 SDK가 요청 서명을 대신 처리한다는 장점이 있습니다. API 요청을 수동으로 생성하고 서명해야 하는 경우 요청에 서명하는 방법을 알아보려면 Amazon Web Services 일반 참조의 서명된 AWS API 요청 생성으로 이동하세요.

Example 응답의 예

```
<GetSessionTokenResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
<GetSessionTokenResult>
<Credentials>
  <SessionToken>
   AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrRh3c/L
   To6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/IvU1dYUg2RVAJBanLiHb4IgRmpRV3z
   rkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/AXlzBBko7b15fjrBs2+cTQtp
   Z3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE
  </SessionToken>
  <SecretAccessKey>
  wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY
  </SecretAccessKey>
```

```
  <Expiration>2011-07-11T19:55:29.611Z</Expiration>
  <AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
</Credentials>
</GetSessionTokenResult>
<ResponseMetadata>
<RequestId>58c5dbae-abef-11e0-8cfe-09039844ac7d</RequestId>
</ResponseMetadata>
</GetSessionTokenResponse>
```

선택 사항으로 GetSessionToken 요청은 AWS 멀티 팩터 인증(MFA) 확인에 대한 SerialNumber 및 TokenCode 값을 포함할 수 있습니다. 제공한 값이 유효하면 AWS STS에서는 MFA 인증 상태가 포함된 임시 보안 자격 증명을 제공합니다. 그런 다음 임시 보안 자격 증명은 MFA 인증이 유효한 동안 MFA로 보호되는 API 작업 또는 AWS 웹 사이트에 액세스하는 데 사용할 수 있습니다.

다음 예는 MFA 확인 코드 및 디바이스 일련 번호를 포함하는 GetSessionToken 요청을 보여줍니다.

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=GetSessionToken
&DurationSeconds=7200
&SerialNumber=YourMFADeviceSerialNumber
&TokenCode=123456
&AUTHPARAMS
```

> **Note**
>
> AWS STS에 대한 호출은 전역 엔드포인트 또는 AWS 계정이 활성화된 리전 엔드포인트 어느 곳으로도 이루어질 수 있습니다. 자세한 내용은 리전 및 엔드포인트의 AWS STS 섹션을 참조하세요.
> 예시의 AUTHPARAMS 파라미터는 서명에 대한 자리 표시자입니다. 서명은 AWS HTTP API 요청에 포함해야 하는 인증 정보입니다. AWS SDK를 사용하여 API 요청을 생성하는 것이 좋습니다. 이렇게 하면 SDK가 요청 서명을 대신 처리한다는 장점이 있습니다. API 요청을 수동으로 생성하고 서명해야 하는 경우 요청에 서명하는 방법을 알아보려면 Amazon Web Services 일반 참조의 서명된 AWS API 요청 생성을 참조하세요.

AWS STS API 작업 비교

다음 표는 임시 보안 자격 증명을 반환하는 AWS STS의 API 작업이 수행하는 기능을 비교해 보여줍니다. 역할을 수임해 임시 보안 자격 증명을 요청하는 데 사용할 수 있는 여러 방법을 알아보려면 역할 수

임 방법 섹션을 참조하세요. 세션 태그를 전달할 수 있는 다양한 AWS STS API 작업에 대한 자세한 내용은 AWS STS에서 세션 태그 전달 섹션을 참조하세요.

AWS STS API	호출할 수 있는 사용자	자격 증명의 수명(최소 | 최대 | 기본)	MFA 지원[1]	세션 정책 지원[2]	결과로 얻은 임시 자격 증명에 대한 제한
AssumeRole	IAM 사용자 또는 기존 임시 보안 자격 증명이 있는 IAM 역할	15분 | 최대 세션 기간 설정[3] | 1시간	예	예	`GetFederationToken` 또는 `GetSessionToken` 호출 불가
AssumeRoleWithSAML	어떤 사용자나 호출자도 잘 알려진 자격 증명 공급자의 인증을 나타내는 SAML 인증 응답을 반드시 전달해야 합니다.	15분 | 최대 세션 기간 설정[3] | 1시간	아니요	예	`GetFederationToken` 또는 `GetSessionToken` 호출 불가
AssumeRoleWithWebIdentity	모든 사용자, 호출자는 알려진 ID 공급자의 인증을 나타내는 OIDC 호환 JWT 토큰을 반드시 전달해야 합니다.	15분 | 최대 세션 기간 설정[3] | 1시간	아니요	예	`GetFederationToken` 또는 `GetSessionToken` 호출 불가
GetFederationToken	IAM 사용자 또는 AWS 계정 루트 사용자	IAM 사용자: 15분 | 36시간 | 12시간 루트 사용자: 15	아니요	예	AWS CLI 또는 AWS API를 사용하여 IAM 작업을 호출할 수 없습니다. 이 제한은 콘솔 세션에는 적용되지 않습니다.

AWS STS API	호출할 수 있는 사용자	자격 증명의 수명(최소 | 최대 | 기본)	MFA 지원[1]	세션 정책 지원[2]	결과로 얻은 임시 자격 증명에 대한 제한
		분 | 1시간 | 1시간			`GetCallerIdentity` 를 제외한 AWS STS 작업을 호출할 수 없습니다.[4] 콘솔로의 SSO가 허용됩니다.[5]
GetSessionToken	IAM 사용자 또는 AWS 계정 루트 사용자	IAM 사용자: 15분 | 36시간 | 12시간 루트 사용자: 15분 | 1시간 | 1시간	예	아니요	요청에 MFA 정보를 포함되지 않으면 IAM API 작업을 호출할 수 없습니다. `AssumeRole` 또는 `GetCallerIdentity` 를 제외한 AWS STS API 작업 호출 불가 콘솔로의 SSO는 허용되지 않습니다.[6]

[1] MFA 지원. AssumeRole 및 GetSessionToken API 작업을 호출할 때 멀티 팩터 인증(MFA)에 대한 정보를 포함시킬 수 있습니다. 이는 API 호출의 결과물인 임시 보안 자격 증명을 MFA 디바이스로 인증된 사용자들만 사용할 수 있게 해줍니다. 자세한 내용은 MFA를 통한 보안 API 액세스 단원을 참조하십시오.

[2] 세션 정책 지원. 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 정책입니다. 이 정책은 세션에 할당된 역할/사용자 자격 증명 기반 정책의 권한을 제한합니다. 결과적으로 얻는 세션의 권한은 엔터티의 자격 증명 기반 정책과 세션 정책의 교집합입니다 세션 정책을 사용하여 수임된 역할의 자격 증명 기반 정책에서 허용되는 것보다 더 많은 권한을 부여할 수는 없습니다. 역할 세션 권한에 대한 자세한 정보는 세션 정책 섹션을 참조하세요.

[3] 최대 세션 기간 설정. `DurationSeconds` 파라미터를 사용하여 역할 세션 기간을 900초(15초)에서 해당 역할에 대한 최대 세션 기간 설정까지 지정합니다. 역할에 대한 최댓값을 확인하는 방법을 알아보려면 역할의 최대 세션 기간 업데이트 섹션을 참조하세요.

[4] GetCallerIdentity. 이 작업을 실행하는 데 따로 권한이 필요하지 않습니다. 관리자가 `sts:GetCallerIdentity` 작업에 대한 액세스를 명시적으로 거부하는 정책을 IAM 사용자 또는 역할에게 추가하더라도 이 작업을 계속해서 실행할 수 있습니다. 권한이 필요하지 않은 이유는 IAM 사용자 또는 역할의 액세스가 거부되어도 반환되는 정보는 동일하기 때문입니다. 응답 예제를 보려면 iam:DeleteVirtualMFADevice를 수행할 권한이 없음 섹션을 참조하세요.

[5] 콘솔로 SSO(Single Sign-On)하기 SSO를 지원하기 위해 AWS는 페더레이션 엔드포인트(`https://signin.aws.amazon.com/federation`)를 호출해 임시 보안 자격 증명을 전달할 수 있게 해줍니다. 엔드포인트는 암호 없이도 사용자를 콘솔에 바로 로그인시켜주는 URL을 구성하는 데 사용 가능한 토큰을 반환합니다. 자세한 내용은 AWS 보안 블로그에서 SAML 2.0 페더레이션 사용자가 AWS Management Console에 액세스할 수 있게 하기 및 AWS 관리 콘솔에 대한 크로스 계정 액세스를 활성화하는 방법을 참조하세요.

[6] 임시 자격 증명을 검색한 이후에 연동 SSO 엔드포인트로 자격 증명을 전달하여 AWS Management Console에 액세스할 수 없습니다. 자세한 내용은 사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기 단원을 참조하십시오.

AWS 리소스에서 임시 자격 증명 사용

임시 보안 자격 증명을 사용해 AWS CLI 또는 AWS API에서 AWS 리소스를 프로그래밍 방식으로 요청할 수 있습니다(AWS SDK 사용). 임시 보안 자격 증명은 IAM 사용자 자격 증명과 같은 장기 보안 자격 증명을 사용하는 것과 동일한 권한을 제공합니다. 그러나 몇 가지 차이점이 있습니다.

- 임시 보안 자격 증명을 사용해 호출할 경우 그 호출에 반드시 세션 토큰이 포함되어야 하는데, 이 세션 토큰은 임시 자격 증명과 함께 반환됩니다. AWS는 세션 토큰을 사용해 임시 보안 자격 증명의 유효성을 검증합니다.
- 임시 자격 증명은 지정된 간격 후에 만료됩니다. 임시 자격 증명이 만료된 후에는 그 자격 증명을 사용한 어떤 요청도 실패할 것이므로 일련의 새로운 임시 자격 증명을 생성해야 합니다. 임시 자격 증명을 원래 지정된 간격 이상으로 확장하거나 새로 고칠 수 없습니다.
- 임시 자격 증명을 사용하여 요청하면 보안 주체에 태그 세트가 포함될 수 있습니다. 이러한 태그는 세션 태그와 사용자가 맡은 역할에 연결된 태그에서 가져옵니다. 세션 태그에 대한 자세한 내용은 AWS STS에서 세션 태그 전달 섹션을 참조하세요.

AWS SDK, AWS Command Line Interface (AWS CLI) 또는 Tools for Windows PowerShell을 사용하는 경우 임시 보안 자격 증명을 가져오고 사용하는 방법은 컨텍스트에 따라 다릅니다. EC2 인스턴스 내부에서 코드, AWS CLI 또는 Tools for Windows PowerShell 명령을 실행 중이라면 Amazon EC2에 대한 역할을 이용할 수 있습니다. 그렇지 않은 경우 AWS STS API를 호출해 임시 자격 증명을 얻은 다음, 그 자격 증명을 사용해 AWS 서비스를 명시적으로 호출할 수 있습니다.

> **Note**
>
> AWS Security Token Service(AWS STS)를 사용하면 AWS 리소스에 대한 액세스를 제어할 수 있는 임시 보안 자격 증명을 생성하여 신뢰받는 사용자에게 제공할 수 있습니다. AWS STS에 대한 자세한 정보는 IAM의 임시 보안 자격 증명 섹션을 참조하세요. AWS STS는 `https://sts.amazonaws.com`에 기본 엔드포인트가 있는 전역적 서비스입니다. 이 엔드포인트는 미국 동부(버지니아 북부) 리전에 있지만 이 엔드포인트 및 다른 엔드포인트에서 얻은 자격 증명은 전역적으로 유효합니다. 이러한 자격 증명은 모든 리전의 서비스 및 리소스에서 작동합니다. 지원되는 리전에서 엔드포인트에 대한 AWS STS API 호출을 할 수도 있습니다. 이렇게 지리적으로 더 가까운 리전에 있는 서버에서 요청함으로써 지연 시간을 단축할 수 있습니다. 자격 증명은 어떤 리전에서 오는지 상관없이 전역적으로 유효합니다. 자세한 내용은 AWS STS에서 AWS 리전 관리 단원을 참조하십시오.

목차

Amazon EC2 인스턴스에서 임시 자격 증명 사용

EC2 인스턴스 내에서 AWS CLI 명령 또는 코드를 실행하고자 하는 경우 자격 증명을 얻는 바람직한 방법은 Amazon EC2에 대한 역할을 사용하는 것입니다. EC2 인스턴스 상에서 실행되는 애플리케이션에 부여하고 싶은 권한을 지정하는 IAM 역할을 생성합니다. 인스턴스를 시작할 때 그 역할을 인스턴스에 연결합니다.

인스턴스 상에서 실행되는 애플리케이션, AWS CLI 및 Tools for Windows PowerShell 명령은 인스턴스 메타데이터로부터 자동 임시 보안 자격 증명을 얻을 수 있습니다. 임시 보안 자격 증명을 명시적으로 가져올 필요는 없습니다. AWS SDK, AWS CLI 및 Tools for Windows PowerShell은 EC2 인스턴스 메타데이터 서비스(IMDS)에서 보안 인증 정보를 자동으로 가져와서 사용합니다. 임시 자격 증명은 그 인스턴스에 연결된 역할에 대해 정의한 권한이 있습니다.

자세한 내용 및 예시는 다음을 참조하세요.

- Amazon Elastic Compute Cloud에서 IAM 역할을 사용하여 AWS 리소스에 대한 액세스 권한 부여 - AWS SDK for Java
- IAM 역할을 사용하여 액세스 권한 부여 - AWS SDK for .NET
- 역할 생성 - AWS SDK for Ruby

AWS SDK에서 임시 보안 자격 증명 사용

코드에서 임시 보안 자격 증명을 사용하려면 AssumeRole과 같이 AWS STS API를 프로그래밍 방식으로 호출하고 결과 자격 증명 및 세션 토큰을 추출합니다. 그런 다음 해당 값을 AWS에 대한 후속 호출을 위한 자격 증명으로 사용하면 됩니다. 다음 예는 AWS SDK를 사용할 경우 임시 보안 자격 증명을 사용하는 방법에 대한 유사 코드를 보여줍니다.

```
assumeRoleResult = AssumeRole(role-arn);
tempCredentials = new SessionAWSCredentials(
   assumeRoleResult.AccessKeyId,
   assumeRoleResult.SecretAccessKey,
   assumeRoleResult.SessionToken);
s3Request = CreateAmazonS3Client(tempCredentials);
```

Python으로 작성된 예제(AWS SDK for Python (Boto) 사용)는 IAM 역할로 전환(AWS API) 섹션을 참조하세요. 이 예제에서는 AssumeRole을 호출하여 임시 보안 자격 증명을 가져온 다음 해당 자격 증명을 사용하여 Amazon S3를 호출하는 방법을 보여줍니다.

AssumeRole, GetFederationToken 및 기타 API 작업을 호출하는 방법에 대한 자세한 내용은 AWS Security Token Service API 참조를 참조하세요. 이러한 호출의 결과에서 임시 보안 자격 증명 및 세션 토큰을 얻는 방법에 대한 자세한 내용은 사용하고 있는 SDK의 설명서를 참조하세요. 기본 AWS 설명서 페이지의 SDK 및 도구 키트 섹션에서 모든 AWS SDK에 대한 설명서를 확인할 수 있습니다.

이전 자격 증명이 만료되기 전에 반드시 새로운 일련의 자격 증명을 얻도록 해야 합니다. 일부 SDK에서는 자격 증명 갱신 프로세스를 관리해주는 공급자를 사용할 수 있습니다. 사용하고 있는 SDK의 설명서를 확인하세요.

AWS CLI에서 임시 보안 자격 증명 사용

AWS CLI에서 임시 보안 자격 증명을 사용할 수 있습니다. 이 임시 보안 자격 증명은 정책을 테스트하는 데 유용합니다.

AWS CLI를 사용하여 AssumeRole 또는 GetFederationToken과 같은 AWS STS API를 호출한 다음 결과 출력을 캡처할 수 있습니다. 다음 예는 파일에 출력을 전송하는 AssumeRole에 대한 호출을

보여줍니다. 이 예제에서 `profile` 파라미터는 AWS CLI 구성 파일의 프로파일로 간주됩니다. 또한 역할을 수임할 권한이 있는 IAM 사용자의 자격 증명을 참조하는 것으로 간주됩니다.

```
aws sts assume-role --role-arn arn:aws:iam::123456789012:role/role-name --role-session-name "RoleSession1" --profile IAM-user-name > assume-role-output.txt
```

명령이 끝나면 라우팅한 위치에서 액세스 키 ID, 보안 액세스 키 및 세션 토큰을 추출할 수 있습니다. 수동으로 또는 스크립트를 사용하여 이 작업을 수행할 수 있습니다. 그런 다음 이 값을 환경 변수에 할당할 수 있습니다.

AWS CLI 명령을 실행할 때 AWS CLI는 환경 변수로 시작하여 구성 파일로 넘어가는 특정 순서로 자격 증명을 찾습니다. 따라서 임시 자격 증명을 환경 변수에 넣은 후에 AWS CLI는 그 자격 증명을 기본 값으로 사용합니다. (명령에 `profile` 파라미터를 지정한다면 AWS CLI는 환경 변수를 건너뜁니다. 대신 AWS CLI는 구성 파일을 검색합니다. 이로써 필요한 경우 환경 변수에서 자격 증명을 무시할 수 있게 됩니다.)

다음 예는 임시 보안 자격 증명에 대한 환경 변수를 설정한 다음, AWS CLI 명령을 호출하는 방법을 보여줍니다. `profile` 파라미터는 AWS CLI 명령에 포함되어 있지 않기 때문에 AWS CLI는 먼저 환경 변수에서 자격 증명을 검색하고, 따라서 임시 자격 증명을 사용합니다.

Linux

```
$ export AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
$ export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of session token>
$ aws ec2 describe-instances --region us-west-1
```

Windows

```
C:\> SET AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE
C:\> SET AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
C:\> SET AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of token>
C:\> aws ec2 describe-instances --region us-west-1
```

API 작업을 통해 임시 보안 자격 증명 사용

AWS로 직접 HTTPS API 요청을 하는 경우, AWS Security Token Service(AWS STS)에서 가져오는 임시 보안 자격 증명으로 그러한 요청에 서명할 수 있습니다. 이렇게 하려면 AWS STS에서 받은 보안 액세스 키와 액세스 키 ID를 사용합니다. 장기 자격 증명을 사용하는 것과 동일한 방식으로 액세스 키 ID 및 보안 액세스 키를 사용하여 요청에 서명합니다. 또한 AWS STS로부터 받는 세션 토큰을 API 요청에

추가합니다. 그 세션 토큰을 HTTP 헤더 또는 `X-Amz-Security-Token`이라는 쿼리 문자열 파라미터에 추가합니다. 그 세션 토큰을 HTTP 헤더 또는 쿼리 문자열 파라미터에 추가하되, 하나에만 추가해야 합니다. HTTPS API 요청 서명에 대한 자세한 내용은 AWS 일반 참조의 AWS API 요청에 서명을 참조하세요.

추가 정보

다른 AWS 서비스와 함께 AWS STS를 사용하는 방법에 대한 자세한 내용은 다음 링크를 참조하세요.

- Amazon S3. Amazon Simple Storage Service 사용 설명서에서 IAM 사용자 임시 보안 인증을 사용하여 요청 또는 페더레이션 사용자 임시 보안 인증을 사용하여 요청을 참조하세요.
- Amazon SNS Amazon Simple Notification Service 개발자 안내서의 Amazon SNS로 자격 증명 기반 정책 사용을 참조하세요.
- Amazon SQS. Amazon Simple Queue Service 개발자 안내서의 Amazon SQS의 Identity and Access Management를 참조하세요.
- Amazon SimpleDB Amazon SimpleDB 개발자 안내서의 임시 보안 자격 증명 사용을 참조하세요.

사용자 임시 보안 자격 증명에 대한 권한 제어

AWS Security Token Service(AWS STS)를 사용하면 AWS 리소스에 대한 액세스를 제어할 수 있는 임시 보안 자격 증명을 생성하여 신뢰받는 사용자에게 제공할 수 있습니다. 에 대한 자세한 내용은 AWS STS 섹션을 참조하세요.IAM의 임시 보안 자격 증명. 임시 보안 자격 증명은 AWS STS가 발급한 후 만료 기간 동안 유효하며 취소될 수 없습니다. 그러나 임시 보안 자격 증명에 할당된 권한은 자격 증명을 사용해 요청이 이루어질 때마다 평가되기 때문에 자격 증명이 발급된 이후에라도 액세스 권한을 변경함으로써 자격 증명 취소 효과를 얻을 수 있습니다.

다음 주제는 독자가 AWS 권한 및 정책에 대한 유효한 지식이 있다고 가정합니다. 이 주제에 대한 자세한 내용은 AWS리소스에 대한 액세스 관리 섹션을 참조하세요.

주제

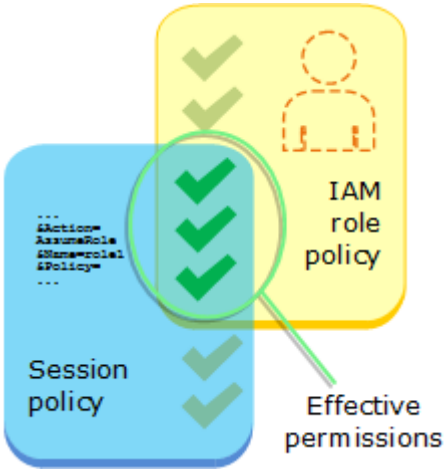
- AssumeRole, AssumeRoleWithSAML 및 AssumeRoleWithWebIdentity에 대한 권한
- 위임된 역할로 수행한 작업 모니터링 및 제어
- GetFederationToken에 대한 권한
- GetSessionToken에 대한 권한
- 임시 보안 자격 증명에 대한 권한 비활성화
- 임시 보안 자격 증명을 생성할 수 있는 권한 부여

- ID 인식 콘솔 세션을 사용하는 권한 부여

AssumeRole, AssumeRoleWithSAML 및 AssumeRoleWithWebIdentity에 대한 권한

수임된 역할에 대한 권한 정책은 `AssumeRole`, `AssumeRoleWithSAML` 및 `AssumeRoleWithWebIdentity`에 의해 반환되는 임시 보안 자격 증명에 대한 권한을 결정합니다. 역할을 생성 또는 업데이트할 때 이러한 권한을 정의합니다.

인라인 또는 관리형 세션 정책을 `AssumeRole`, `AssumeRoleWithSAML` 또는 `AssumeRoleWithWebIdentity` API 작업의 파라미터로 전달할 수 있습니다. 세션 정책은 역할의 임시 자격 증명 세션에 대한 권한을 제한합니다. 결과적으로 얻는 세션의 권한은 역할의 자격 증명 기반 정책의 교차와 세션 정책입니다. 후속 AWS API 호출 시에도 역할의 임시 자격 증명을 사용하여 역할이 속한 계정의 리소스에 액세스할 수 있습니다. 세션 정책을 사용하여 수임된 역할의 자격 증명 기반 정책에서 허용되는 것보다 더 많은 권한을 부여할 수는 없습니다. 이 역할의 효과적인 권한을 AWS가 어떻게 결정하는지 자세히 알아보려면 정책 평가 로직 섹션을 참조하세요.



'허용' 또는 '거부' 권한 부여 결정을 내릴 때 `AssumeRole`에 대한 원래 호출을 생성한 자격 증명에 연결된 정책은 AWS에서 평가하지 않습니다. 해당 사용자는 맡은 역할에 의해 할당된 권한을 위해 자신의 원래 권한을 일시적으로 포기합니다. `AssumeRoleWithSAML` 및 `AssumeRoleWithWebIdentity` API 작업의 경우 API 호출자가 AWS 자격 증명이 아니기 때문에 평가할 정책이 없습니다.

예: AssumeRole을 사용한 권한 할당

서로 다른 종류의 정책으로 `AssumeRole` API 작업을 사용할 수 있습니다. 여기 몇 가지 예가 있습니다.

역할 권한 정책

이 예에서는 선택 사항인 Policy 파라미터에 세션 정책을 지정하지 않고 AssumeRole API 작업을 호출합니다. 임시 자격 증명에 할당된 권한은 위임된 역할의 권한 정책에 따라 결정됩니다. 다음 예제 권한 정책은 S3 버킷 productionapp에 포함된 객체를 모두 나열하도록 역할 권한을 부여합니다. 또한 해당 역할이 이 버킷 내에서 객체를 가져오고, 배치하고, 삭제하도록 허용합니다.

Example 역할 권한 정책 예

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::productionapp"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::productionapp/*"
    }
  ]
}
```

파라미터로 전달되는 세션 정책

사용자에게 이전 예제와 동일한 역할을 수임하도록 허용하려 한다고 가정해 보겠습니다. 하지만 이 경우 역할 세션에 대해 productionapp S3 버킷에/에서 객체를 넣거나 가져오는 작업만을 허용하는 권한을 부여하고자 합니다. 객체를 삭제할 수 없도록 하고자 합니다. 이렇게 하기 위한 한 가지 방법은 새 역할을 만들어 그 역할의 권한 정책에 원하는 권한을 지정하는 것입니다. 또 다른 방법은 AssumeRole API를 호출하여 선택 사항인 Policy 파라미터의 세션 정책을 API 작업의 일부로 포함하는 것입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 자격 증명 기반 정책의 교집합과 세션 정책입니다. 세션 정책을 사용하여 수임된 역할의 자격 증명 기반 정책에서 허용되는 것보다 더 많은 권한을 부여할 수는 없습니다. 역할 세션 권한에 대한 자세한 정보는 세션 정책 섹션을 참조하세요.

새 세션의 임시 자격 증명을 검색한 후 이를 권한을 부여하고자 하는 사용자에게 전달할 수 있습니다.

예를 들어 다음 정책이 API 호출의 파라미터로 전달된다고 가정합시다. 세션을 사용하는 사람에게는 다음 작업에 대한 수행 권한만 부여됩니다.

- productionapp 버킷에 있는 모든 객체의 목록을 조회합니다.
- 객체를 가져와 productionapp 버킷에 넣습니다.

다음 세션 정책에서는 s3:DeleteObject 권한이 필터링되어 위임된 세션에 s3:DeleteObject 권한이 부여되지 않습니다. 이 정책은 역할 세션에 대한 최대 권한을 설정하여 역할에 대한 기존 권한 정책을 재정의합니다.

Example **AssumeRole** API 호출로 전달된 세션 정책 예

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::productionapp"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::productionapp/*"
    }
  ]
}
```

리소스 기반 정책

일부 AWS 리소스는 리소스 기반 정책을 지원하고 이 정책은 임시 보안 자격 증명에 영향을 미치는 권한을 정의하는 또 다른 메커니즘을 제공합니다. Amazon S3 버킷, Amazon SNS 주제, Amazon SQS 대기열 같은 일부 리소스만이 리소스 기반 정책을 지원합니다. 다음 예는 앞의 예들을 확장한 것으로서 productionapp이라는 S3 버킷을 사용합니다. 다음 정책은 버킷에 연결되어 있습니다.

다음 리소스 기반 정책을 productionapp 버킷에 연결할 때 모든 사용자들은 버킷에서 객체를 삭제할 권한을 거부당합니다. (정책의 Principal 요소에 유의하세요.) 역할 권한 정책이 DeleteObject

권한을 부여한다 해도 여기에는 모든 수임된 역할 사용자들이 포함됩니다. 명시적인 Deny 문은 항상 Allow 문보다 우선 적용됩니다.

Example 버킷 정책 예제

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": {"AWS": "*"},
    "Effect": "Deny",
    "Action": "s3:DeleteObject",
    "Resource": "arn:aws:s3:::productionapp/*"
  }
}
```

다수의 정책 유형이 AWS에 의해 어떻게 결합되고 평가되는지에 대한 자세한 정보는 정책 평가 로직 섹션을 참조하세요.

위임된 역할로 수행한 작업 모니터링 및 제어

IAM 역할은 권한이 할당된 IAM의 객체입니다. IAM 자격 증명 또는 AWS 외부의 자격 증명을 사용하여 해당 역할을 수임하면 역할에 할당된 권한이 있는 세션을 받습니다.

AWS에서 작업을 수행하는 경우 세션에 대한 정보를 AWS CloudTrail에 로깅하여 계정 관리자가 모니터링하도록 할 수 있습니다. 관리자는 AWS에서 작업을 수행하는 사용자 또는 애플리케이션을 식별하는 사용자 지정 문자열을 전달하기 위해 자격 증명을 요구하도록 역할을 구성할 수 있습니다. 이 자격 증명 정보는 AWS CloudTrail에 소스 자격 증명으로 저장됩니다. 관리자가 CloudTrail에서 활동을 검토할 때 소스 자격 증명 정보를 보고 수임된 역할 세션에서 누가 또는 어떤 작업을 수행했는지 확인할 수 있습니다.

소스 자격 증명은 설정된 후에는 역할 세션 중 수행되는 모든 AWS 작업에 대해 요청에 표시됩니다. 설정된 값은 역할이 AWS CLI 또는 AWS API를 통해 다른 역할을 수임(역할 체인이라고 함)할 때 유지됩니다. 역할 세션 중에는 설정된 값을 변경할 수 없습니다. 관리자는 소스 자격 증명의 존재 여부나 값에 따라 세분화된 권한을 구성하여 공유 역할로 수행되는 AWS 작업을 보다 효과적으로 제어할 수 있습니다. 소스 자격 증명 속성을 사용할 수 있는지 여부, 필요한지 여부 및 사용할 수 있는 값을 결정할 수 있습니다.

소스 자격 증명을 사용하는 방법은 역할 세션 이름 및 세션 태그와 크게 다릅니다. 소스 자격 증명 값은 설정한 후에는 변경할 수 없고 역할 세션에서 수행되는 추가 작업에 대해 유지됩니다. 세션 태그와 역할 세션 이름을 사용하는 방법은 다음과 같습니다.

- 세션 태그 - 역할을 수임하거나 사용자를 페더레이션할 때 세션 태그를 전달할 수 있습니다. 세션 태그는 역할을 수임할 때 표시됩니다. 태그 조건 키를 사용하여 해당 태그를 기반으로 보안 주체에 권한을 부여하는 정책을 정의할 수 있습니다. 그런 다음 CloudTrail을 사용하여 역할을 수임하거나 사용자를 페더레이션하기 위한 요청을 볼 수 있습니다. 세션 태그에 대한 자세한 내용은 AWS STS에서 세션 태그 전달 섹션을 참조하세요.
- 역할 세션 이름 - 역할 신뢰 정책에서 `sts:RoleSessionName` 조건 키를 사용하여 사용자가 역할을 수임할 때 특정 세션 이름을 제공하도록 요구할 수 있습니다. 여러 보안 주체가 한 역할을 사용할 때 역할 세션 간을 구분하기 위해 역할 세션 이름을 사용할 수 있습니다. 역할 세션 이름에 대한 자세한 내용은 sts:RoleSessionName을 참조하세요.

역할을 수임하는 자격 증명을 제어하려는 경우 소스 자격 증명을 사용하는 것이 좋습니다. 또한 소스 자격 증명은 CloudTrail 로그를 마이닝하여 누가 역할을 사용하여 작업을 수행했는지 확인하는 데에도 유용합니다.

주제

- 소스 자격 증명을 사용하도록 설정
- 소스 자격 증명에 대해 알아야 할 사항
- 소스 자격 증명 설정에 필요한 권한
- 역할을 수임할 때 소스 자격 증명 지정
- AssumeRole에 소스 자격 증명 사용
- AssumeRoleWithSAML에 소스 자격 증명 사용
- AssumeRoleWithWebIdentity에 소스 자격 증명 사용
- 소스 자격 증명 정보를 사용하여 액세스 제어
- CloudTrail에서 소스 자격 증명 보기

소스 자격 증명을 사용하도록 설정

소스 자격 증명을 사용하도록 설정하는 방법은 역할을 수임할 때 사용하는 방법에 따라 다릅니다. 예를 들어 IAM 사용자는 `AssumeRole` 작업을 사용하여 역할을 수임할 수 있습니다. 엔터프라이즈 자격 증명(인력 자격 증명이라고도 함)이 있으면 `AssumeRoleWithSAML`을 사용하여 AWS 리소스에 액세스할 수 있습니다. 최종 사용자가 모바일 또는 웹 애플리케이션에 액세스하는 경우 `AssumeRoleWithWebIdentity`를 사용하여 설정할 수 있습니다. 다음은 기존 환경에서 소스 자격 증명 정보를 활용하도록 설정하는 방법을 이해하는 데 도움이 되는 간략한 작업 흐름의 개요입니다.

1. 테스트 사용자 및 역할 구성 - 프로덕션 이전 환경을 사용하여 테스트 사용자 및 역할을 구성하고 소스 자격 증명을 설정할 수 있도록 정책을 구성합니다.

   페더레이션 자격 증명을 위해 자격 증명 공급자(IdP)를 사용하는 경우 어설션 또는 토큰으로 원하는 소스 자격 증명의 사용자 속성을 전달하도록 IdP를 구성합니다.
2. 역할 수임 - 테스트를 위해 설정한 사용자와 역할을 사용하여 역할 수임과 소스 자격 증명 전달을 테스트합니다.
3. CloudTrail 검토 - CloudTrail 로그에서 테스트 역할의 소스 자격 증명 정보를 검토합니다.
4. 사용자 교육 - 프로덕션 이전 환경에서 테스트한 후에는 필요한 경우 소스 자격 증명 정보를 전달하는 방법을 사용자에게 알려야 합니다. 사용자가 프로덕션 환경에서 소스 자격 증명을 제공해야 하는 시기의 기한을 설정합니다.
5. 프로덕션 정책 구성 - 프로덕션 환경의 정책을 구성한 다음 프로덕션 사용자 및 역할에 추가합니다.
6. 활동 모니터링 - CloudTrail 로그를 사용하여 프로덕션 역할 활동을 모니터링합니다.

소스 자격 증명에 대해 알아야 할 사항

소스 자격 증명 사용 시 다음 사항에 유의하세요.

- 자격 증명 공급자(IdP)에 연결된 모든 역할에 대한 역할 신뢰 정책에 `sts:SetSourceIdentity` 권한이 있어야 합니다. 역할 신뢰 정책에 이 권한이 없는 역할의 경우 `AssumeRole*` 작업이 실패합니다. 각 역할에 대한 역할 신뢰 정책을 업데이트하지 않으려는 경우 소스 자격 증명을 전달하는 데 개별 IdP 인스턴스를 사용할 수 있습니다. 그런 다음 개별 IdP에 연결된 역할에만 `sts:SetSourceIdentity` 권한을 추가합니다.
- 자격 증명이 소스 자격 증명을 설정하는 경우 `sts:SourceIdentity` 키가 요청에 있습니다. 역할 세션 중에 수행된 후속 작업의 경우 `aws:SourceIdentity` 키가 요청에 있습니다. AWS에서는 `sts:SourceIdentity` 또는 `aws:SourceIdentity` 키에 있는 소스 자격 증명 값을 제어하지 않습니다. 소스 자격 증명을 요구하도록 선택한 경우 사용자 또는 IIdP에서 제공할 속성을 선택해야 합니다. 보안을 위해 이러한 값이 제공되는 방식을 제어할 수 있어야 합니다.
- 소스 자격 증명의 값은 2~64자여야 하며 영숫자, 밑줄 및 다음 문자만 포함할 수 있습니다. . , + = @ -(하이픈). **`aws:`** 텍스트로 시작하는 값은 사용할 수 없습니다. 이 접두사는 AWS 내부 전용으로 예약되어 있습니다.
- AWS 서비스 또는 서비스 연결 역할이 페더레이션 또는 인력 자격 증명 대신 작업을 수행하는 경우에는 소스 자격 증명 정보가 CloudTrail에 의해 캡처되지 않습니다.

> **Important**
>
> 역할을 수임할 때 소스 자격 증명을 설정해야 하는 역할로는 AWS Management Console에서 전환할 수 없습니다. 이러한 역할을 수임하려면 AWS CLI 또는 AWS API를 사용하여 `AssumeRole` 작업을 호출하고 소스 자격 증명 파라미터를 지정할 수 있습니다.

소스 자격 증명 설정에 필요한 권한

API 작업과 일치하는 작업 외에도 정책에는 다음과 같은 권한 전용 작업이 있어야 합니다.

```
sts:SetSourceIdentity
```

- 소스 자격 증명을 지정하려면 보안 주체(IAM 사용자 및 역할)에 `sts:SetSourceIdentity`에 대한 권한이 있어야 합니다. 관리자는 역할 신뢰 정책과 보안 주체의 사용 권한 정책에서 이를 구성할 수 있습니다.
- 다른 역할이 있는 역할을 수임할 때(역할 체인이라고 함) 역할을 맡는 보안 주체의 권한 정책 및 대상 역할을 역할 신뢰 정책 모두에서 `sts:SetSourceIdentity`에 대한 권한이 필요합니다. 그렇지 않으면 역할 수임 작업이 실패합니다.
- 소스 자격 증명을 사용하는 경우 IdP에 연결된 모든 역할에 대한 역할 신뢰 정책에 `sts:SetSourceIdentity` 권한이 있어야 합니다. 이 권한 없이 IdP에 연결된 모든 역할의 경우 `AssumeRole*` 작업이 실패합니다. 각 역할에 대한 역할 신뢰 정책을 업데이트하지 않으려는 경우 소스 자격 증명을 전달하는 데 개별 IdP 인스턴스를 사용하고 별도의 IdP에 연결된 역할에만 `sts:SetSourceIdentity` 권한을 추가할 수 있습니다.
- 계정 경계에 걸쳐 소스 자격 증명을 설정하려면 두 위치에 `sts:SetSourceIdentity` 권한을 포함해야 합니다. 원본 계정에 있는 보안 주체의 권한 정책과 대상 계정에 있는 역할의 역할 신뢰 정책에 있어야 합니다. 예를 들어 역할 체인을 통해 역할이 다른 계정의 역할을 수임하는 데 사용되는 경우 이렇게 해야 할 수 있습니다.

계정 관리자로서, 자기 계정의 IAM 사용자 `DevUser`가 동일한 계정의 `Developer_Role`을 수임하도록 허용하려 한다고 가정합니다. 하지만 사용자가 소스 자격 증명을 IAM 사용자 이름으로 설정한 경우에만 이 작업을 허용하려고 합니다. IAM 사용자에 다음 정책을 연결할 수 있습니다.

Example DevUser에 연결된 자격 증명 기반 정책의 예

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRole",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::123456789012:role/Developer_Role"
    },
    {
      "Sid": "SetAwsUserNameAsSourceIdentity",
      "Effect": "Allow",
      "Action": "sts:SetSourceIdentity",
      "Resource": "arn:aws:iam::123456789012:role/Developer_Role",
      "Condition": {
        "StringLike": {
          "sts:SourceIdentity": "${aws:username}"
        }
      }
    }
  ]
}
```

허용 가능한 소스 자격 증명 값을 적용하려면 다음 역할 신뢰 정책을 구성할 수 있습니다. 이 정책은 IAM 사용자에게 역할을 수임하고 소스 자격 증명을 설정할 수 있는 DevUser 권한을 부여합니다. `sts:SourceIdentity` 조건 키는 허용 가능한 소스 자격 증명 값을 정의합니다.

Example 소스 자격 증명에 대한 역할 신뢰 정책의 예

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDevUserAssumeRole",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/DevUser"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
```

```
          "sts:SourceIdentity": "DevUser"
        }
      }
    }
  ]
}
```

IAM 사용자 DevUser의 자격 증명을 사용하여 사용자가 다음 AWS CLI 요청으로 DeveloperRole을 수임하려고 합니다.

Example AssumeRole CLI 요청의 예

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Developer_Role \
--role-session-name Dev-project \
--source-identity DevUser \
```

AWS에서 요청을 평가할 때 요청 컨텍스트에 DevUser의 sts:SourceIdentity가 포함되어 있습니다.

역할을 수임할 때 소스 자격 증명 지정

AWS STS AssumeRole* API 작업 중 하나를 사용하여 역할의 임시 보안 자격 증명을 가져올 때 소스 자격 증명을 지정할 수 있습니다. 사용하는 API 작업은 사용 사례에 따라 다릅니다. 예를 들어 IAM 역할을 사용하여 IAM 사용자에게 보통은 액세스할 수 없는 AWS 리소스에 대한 액세스 권한을 부여하는 경우 AssumeRole 작업을 사용할 수 있습니다. 엔터프라이즈 ID 페더레이션을 사용하여 인력 사용자를 관리하는 경우 AssumeRoleWithSAML 작업을 사용할 수 있습니다. OIDC 페더레이션을 사용하여 최종 사용자가 모바일 또는 웹 애플리케이션에 액세스할 수 있도록 허용하는 경우 AssumeRoleWithWebIdentity 작업을 사용할 수 있습니다. 다음 섹션에서는 각 작업에 소스 자격 증명을 사용하는 방법을 설명합니다. 임시 자격 증명의 일반적인 시나리오에 대한 자세한 내용은 임시 자격 증명과 관련된 일반적인 시나리오 섹션을 참조하세요.

AssumeRole에 소스 자격 증명 사용

AssumeRole 작업은 AWS 리소스에 액세스하는 데 사용할 수 있는 임시 자격 증명 세트를 반환합니다. IAM 사용자 또는 역할 자격 증명을 사용하여 AssumeRole을 호출할 수 있습니다. 역할을 수임하는 동안 소스 자격 증명을 전달하려면 --source-identity AWS CLI 옵션 또는 SourceIdentity AWS API 파라미터를 사용합니다. 다음 예제에서는 AWS CLI를 사용하여 소스 자격 증명을 지정하는 방법을 보여줍니다.

Example AssumeRole CLI 요청의 예

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/developer \
--role-session-name Audit \
--source-identity Admin \
```

AssumeRoleWithSAML에 소스 자격 증명 사용

`AssumeRoleWithSAML` 작업을 호출하는 보안 주체는 SAML 기반 연동을 사용하여 인증됩니다. 이 작업은 AWS 리소스에 액세스하는 데 사용할 수 있는 임시 자격 증명 세트를 반환합니다. AWS Management Console 액세스를 위해 SAML 기반 연동을 사용하는 방법에 대한 자세한 내용은 SAML 2.0 페더레이션 사용자가 AWS Management Console에 액세스할 수 있게 하기 섹션을 참조하세요. AWS CLI 또는 AWS API 액세스에 대한 자세한 내용은 SAML 2.0 연동 섹션을 참조하세요. Active Directory 사용자를 위한 SAML 연동을 설정하는 방법에 대한 튜토리얼은 AWS 보안 블로그의 Active Directory Federation Services(ADFS)를 사용한 AWS 페더레이션 인증을 참조하세요.

관리자는 회사 디렉터리의 멤버가 AWS STS `AssumeRoleWithSAML` 작업을 사용하여 AWS로 연동하도록 허용할 수 있습니다. 이렇게 하려면 다음 작업을 완료해야 합니다.

1. 조직에서 SAML 공급자를 구성합니다.
2. IAM에서 SAML 공급자를 생성합니다
3. 페더레이션 사용자를 위해 AWS에서 역할 및 해당 권한 구성
4. SAML IdP 구성을 완료하고 SAML 인증 응답에 대한 어설션 생성하기

소스 자격 증명에 대해 SAML 속성을 설정하려면 `Name` 속성과 함께 `Attribute` 요소를 `https://aws.amazon.com/SAML/Attributes/SourceIdentity`로 설정합니다. `AttributeValue` 요소를 사용하여 소스 자격 증명 값을 지정합니다. 예를 들어, 다음 자격 증명 속성을 소스 자격 증명으로 전달한다고 가정합니다.

```
SourceIdentity:DiegoRamirez
```

이러한 속성을 전달하려면 SAML 어설션에 다음 요소를 포함합니다.

Example SAML 어설션의 코드 조각 예

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SourceIdentity">
<AttributeValue>DiegoRamirez</AttributeValue>
```

```
</Attribute>
```

AssumeRoleWithWebIdentity에 소스 자격 증명 사용

`AssumeRoleWithWebIdentity` 작업을 호출하는 보안 주체는 OIDC(OpenID Connect) 호환 페더레이션을 사용하여 인증됩니다. 이 작업은 AWS 리소스에 액세스하는 데 사용할 수 있는 임시 자격 증명 세트를 반환합니다. AWS Management Console 액세스를 위해 OIDC 페더레이션을 사용하는 방법에 대한 자세한 내용은 OIDC 페더레이션 섹션을 참조하세요.

OpenID Connect(OIDC)에서 소스 자격 증명을 전달하려면 JSON 웹 토큰(JWT)에 소스 자격 증명을 포함시켜야 합니다. `AssumeRoleWithWebIdentity` 요청을 제출할 때 토큰의 `https://aws.amazon.com/ source_identity` 네임스페이스에 소스 자격 증명을 포함합니다. OIDC 토큰 및 클레임에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 사용자 풀과 함께 토큰 사용을 참조하세요.

예를 들어, 다음의 디코딩된 JWT는 `Admin` 소스 자격 증명과 함께 `AssumeRoleWithWebIdentity`를 호출하는 데 사용되는 토큰입니다.

Example 디코딩된 JSON 웹 토큰의 예

```
{
    "sub": "johndoe",
    "aud": "ac_oic_client",
    "jti": "ZYUCeRMQVtqHypVPWAN3VB",
    "iss": "https://xyz.com",
    "iat": 1566583294,
    "exp": 1566583354,
    "auth_time": 1566583292,
    "https://aws.amazon.com/source_identity":"Admin"
}
```

소스 자격 증명 정보를 사용하여 액세스 제어

소스 자격 증명이 처음 설정되면 sts:SourceIdentity 키가 요청에 표시됩니다. 소스 자격 증명이 설정된 후 aws:SourceIdentity 키는 역할 세션 중에 수행된 모든 후속 요청에 표시됩니다. 관리자는 소스 자격 증명의 존재 또는 속성의 값에 따라 AWS 작업을 수행할 조건부 권한 부여를 부여하는 정책을 작성할 수 있습니다.

개발자가 프로덕션 크리티컬 AWS 리소스에 쓰기 권한이 있는 중요한 역할을 수임하는 소스 자격 증명을 설정하도록 요구한다고 가정해 봅시다. 또한 `AssumeRoleWithSAML`을 사용하는 인력 자격 증명에

AWS 액세스 권한을 부여한다고 가정합니다. 수석 개발자 Saanvi 및 Diego만 역할에 액세스하게 만들고 싶기 때문에 역할에 대해 다음과 같은 신뢰 정책을 생성합니다.

Example 소스 자격 증명에 대한 역할 신뢰 정책의 예(SAML)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SAMLProviderAssumeRoleWithSAML",
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:saml-provider/name-of-identity-
provider"
      },
      "Action": [
        "sts:AssumeRoleWithSAML"
      ],
      "Condition": {
        "StringEquals": {
          "SAML:aud": "https://signin.aws.amazon.com/saml"
        }
      }
    },
    {
      "Sid": "SetSourceIdentitySrEngs",
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:saml-provider/name-of-identity-
provider"
      },
      "Action": [
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringLike": {
          "sts:SourceIdentity": [
            "Saanvi",
            "Diego"
          ]
        }
      }
    }
  ]
```

```
}
```

신뢰 정책에는 Saanvi 또는 Diego가 중요한 역할을 수임하도록 설정된 소스 자격 증명을 필요로 하는 `sts:SourceIdentity` 조건이 포함되어 있습니다.

또는 페더레이션을 위해 OIDC 공급자를 사용하고 사용자가 `AssumeRoleWithWebIdentity`로 인증되는 경우의 역할 신뢰 정책은 다음과 같습니다.

Example 소스 자격 증명에 대한 역할 신뢰 정책의 예(OIDC 공급자)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/server.example.com"
      },
      "Action": [
        "sts:AssumeRoleWithWebIdentity",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
          "server.example.com:aud": "oidc-audience-id"
        },
        "StringLike": {
          "sts:SourceIdentity": [
            "Saanvi",
            "Diego"
          ]
        }
      }
    }
  ]
}
```

역할 체인 및 교차 계정 요구 사항

`CriticalRole`을 수임한 사용자에게 다른 계정의 `CriticalRole_2`도 수임하도록 허용하고 싶다고 가정해 봅시다. `CriticalRole`을 수임하기 위해 얻은 역할 세션 자격 증명이 다른 계정의 두 번째 역할 `CriticalRole_2`로의 역할 체인에 사용됩니다. 역할이 계정 경계를 건너서 수임

되고 있습니다. 따라서 `sts:SetSourceIdentity` 권한은 `CriticalRole`에 대한 권한 정책 및 `CriticalRole_2`에 대한 역할 신뢰 정책 모두에 대해 부여되어야 합니다.

Example CriticalRole에 대한 권한 정책 예

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRoleAndSetSourceIdentity",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Resource": "arn:aws:iam::222222222222:role/CriticalRole_2"
    }
  ]
}
```

계정 경계에 걸친 소스 자격 증명 설정을 보호하기 위해 다음 역할 신뢰 정책은 `CriticalRole`에 대한 역할 보안 주체만을 신뢰합니다.

Example CriticalRole_2에 대한 역할 신뢰 정책의 예제

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:role/CriticalRole"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringLike": {
          "aws:SourceIdentity": ["Saanvi","Diego"]
        }
      }
    }
```

```
  ]
}
```

사용자는 CriticalRole을 수임하여 얻은 역할 세션 자격 증명을 사용하여 다음 호출을 수행합니다. 소스 자격 증명은 CriticalRole을 수임하는 동안 설정되었으므로 명시적으로 다시 설정할 필요가 없습니다. `CriticalRole`이 수임될 때 사용자가 다음과 같이 설정된 값과 다른 소스 자격 증명을 설정하려고 시도하면 역할 수임 요청이 거부됩니다.

Example AssumeRole CLI 요청의 예

```
aws sts assume-role \
--role-arn arn:aws:iam::222222222222:role/CriticalRole_2 \
--role-session-name Audit \
```

호출하는 보안 주체가 역할을 수임하면 요청의 소스 자격 증명은 첫 번째 수임된 역할 세션에서 유지됩니다. 따라서 `aws:SourceIdentity` 및 `sts:SourceIdentity` 키 둘 모두 요청 컨텍스트에 표시됩니다.

CloudTrail에서 소스 자격 증명 보기

CloudTrail을 사용하여 역할을 수임하거나 사용자를 연동하기 위한 요청을 볼 수 있습니다. AWS에서 작업을 수행하는 역할 또는 사용자 요청을 볼 수도 있습니다. CloudTrail 로그 파일에는 수임하는 역할 또는 페더레이션 사용자 세션의 소스 자격 증명 설정에 대한 정보가 포함됩니다. 자세한 내용은 AWS CloudTrail을 사용하여 IAM 및 AWS STS API 호출 로깅 단원을 참조하세요.

예를 들어 사용자가 AWS STS `AssumeRole` 요청을 만들고 소스 자격 증명을 설정한다고 가정합니다. CloudTrail 로그에서 `requestParameters` 키에서 `sourceIdentity` 정보를 찾을 수 있습니다.

Example AWS CloudTrail 로그의 requestParameters 섹션 예제

```
"eventVersion": "1.05",
    "userIdentity": {
        "type": "AWSAccount",
        "principalId": "AIDAJ45Q7YFFAREXAMPLE",
        "accountId": "111122223333"
    },
    "eventTime": "2020-04-02T18:20:53Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRole",
    "awsRegion": "us-east-1",
```

```
    "sourceIPAddress": "203.0.113.64",
    "userAgent": "aws-cli/1.16.96 Python/3.6.0 Windows/10 botocore/1.12.86",
    "requestParameters": {
        "roleArn": "arn:aws:iam::123456789012:role/DevRole",
        "roleSessionName": "Dev1",
        "sourceIdentity": "source-identity-value-set"
    }
```

사용자가 수임된 역할 세션을 사용하여 작업을 수행하는 경우 소스 자격 증명 정보는 CloudTrail 로그의 `userIdentity` 키에 표시됩니다.

Example AWS CloudTrail 로그의 userIdentity 키 예제

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAJ45Q7YFFAREXAMPLE:Dev1",
    "arn": "arn:aws:sts::123456789012:assumed-role/DevRole/Dev1",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAJ45Q7YFFAREXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/DevRole",
        "accountId": "123456789012",
        "userName": "DevRole"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-02-21T23:46:28Z"
      },
      "sourceIdentity": "source-identity-value-present"
    }
  }
}
```

CloudTrail 로그의 AWS STS API 이벤트 예제를 보려면 CloudTrail 로그의 IAM API 이벤트 예제 섹션을 참조하세요. CloudTrail 로그 파일에 저장된 정보에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 이벤트 참조를 참조하세요.

GetFederationToken에 대한 권한

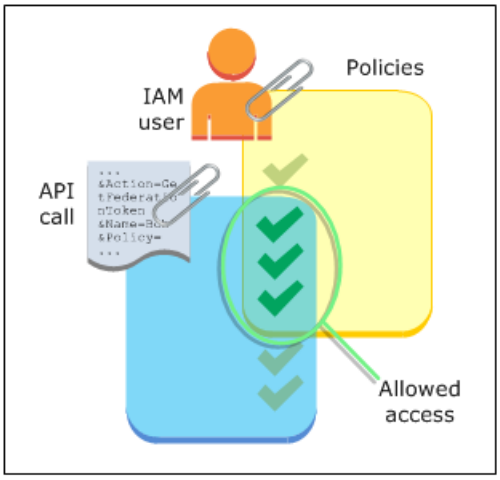
IAM 사용자가 `GetFederationToken` 작업을 호출하고 해당 사용자에 대한 임시 자격 증명을 반환합니다. 이 작업은 사용자를 연동합니다. 페더레이션 사용자에게 할당된 권한은 둘 중 한 곳에 정의되어 있습니다.

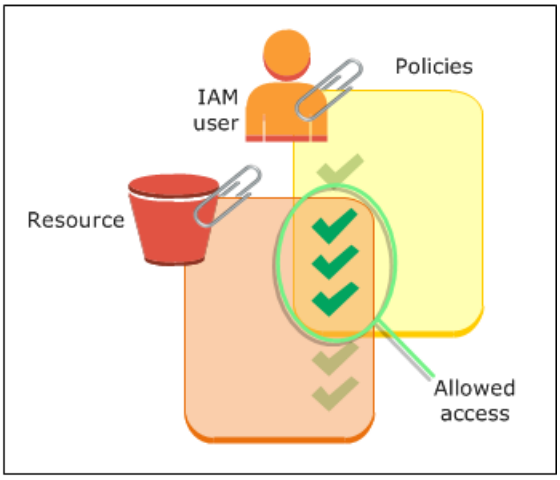
- `GetFederationToken` API 호출의 파라미터로 전달되는 세션 정책. (가장 일반적)
- 정책의 `Principal` 요소에서 페더레이션 사용자를 명시적으로 호명하는 리소스 기반 정책. (일반적이지 않음)

세션 정책은 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 연동 사용자 세션을 생성하고 세션 정책을 전달할 때 결과적으로 얻는 세션의 권한은 사용자의 자격 증명 기반 정책의 교차와 세션 정책입니다. 세션 정책을 사용하여 연동된 사용자의 자격 증명 기반 정책에서 허용되는 권한을 부여할 수는 없습니다.

대부분의 경우 `GetFederationToken` API 호출로 정책을 전달하지 않으면 그 결과 얻게 되는 임시 보안 자격 증명은 아무 권한이 없습니다. 하지만 리소스 기반 정책은 세션에 대한 추가 권한을 제공할 수 있습니다. 세션을 허용된 보안 주체로 지정하는 리소스 기반 정책을 사용하여 리소스에 액세스할 수 있습니다.

다음 그림은 `GetFederationToken` 호출에 의해 반환되는 임시 보안 자격 증명에 대한 권한을 정책들이 어떻게 상호 작용하여 결정하는지를 시각적으로 재현한 것입니다.





예: GetFederationToken을 사용한 권한 할당

서로 다른 종류의 정책으로 `GetFederationToken` API 작업을 사용할 수 있습니다. 여기 몇 가지 예가 있습니다.

IAM 사용자에게 연결된 정책은 다음과 같습니다.

이 예시에는 2개의 백엔드 웹 서비스에 의존하는 브라우저 기반 클라이언트 애플리케이션이 있습니다. 백엔드 서비스 중 하나는 자신만의 인증 서버로서 고유한 자격 증명 시스템을 사용해 클라이언트 애플리케이션을 인증합니다. 다른 백엔드 서비스는 AWS 서비스로, 클라이언트 애플리케이션의 기능 중 일부를 제공합니다. 이 클라이언트 애플리케이션은 서버에 의해 인증되고, 서버는 적절한 권한 정책을 생성하거나 가져옵니다. 서버는 이제 `GetFederationToken` API를 호출해 임시 보안 자격 증명을 얻은 다음, 그 자격 증명을 클라이언트 애플리케이션에 반환합니다. 이제 클라이언트 애플리케이션은 임시 보안 자격 증명을 사용해 AWS 서비스에 직접 요청할 수 있게 됩니다. 이 아키텍처는 클라이언트 애플리케이션이 장기 AWS 자격 증명을 포함하지 않고도 AWS 요청을 할 수 있도록 허용합니다.

인증 서버에서 이름이 `token-app`인 IAM 사용자의 장기 보안 자격 증명을 사용하여 `GetFederationToken` API를 호출합니다. 하지만 장기 IAM 사용자 자격 증명은 서버에 유지되고 클라이언트에 배포되지 않습니다. 다음 예시 정책은 `token-app` IAM 사용자에게 연결되어 페더레이션 사용자(클라이언트)에게 필요한 가장 폭넓은 권한 집합을 정의합니다. `sts:GetFederationToken` 권한은 인증 서비스가 페더레이션 사용자에 대한 임시 보안 자격 증명을 얻는 데 필요하다는 점에 유의하십시오.

> **Note**
>
> AWS는 샘플 Java 애플리케이션을 제공함으로써 이 목적에 기여하는데, Java 애플리케이션은 자격 증명 등록을 위한 토큰 벤딩 머신 - 샘플 Java 웹 애플리케이션에서 다운로드할 수 있습니다.

Example **`GetFederationToken`**을 호출하는 IAM 사용자 **`token-app`**에 연결된 정책의 예

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:GetFederationToken",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "dynamodb:ListTables",
      "Resource": "*"
    },
```

```
    {
      "Effect": "Allow",
      "Action": "sqs:ReceiveMessage",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "sns:ListSubscriptions",
      "Resource": "*"
    }
  ]
}
```

이전 정책은 IAM 사용자에게 여러 가지 권한을 부여합니다. 하지만 이 정책만으로는 페더레이션 사용자에게 권한을 부여하지 않습니다. IAM 사용자가 `GetFederationToken`을 호출하고 정책을 API 호출의 파라미터로 전달하지 않는다면, 그 결과로 얻은 페더레이션 사용자에게는 유효한 권한이 없습니다.

파라미터로 전달되는 세션 정책

페더레이션 사용자에게 적절한 권한이 할당되도록 하는 가장 일반적인 방법은 `GetFederationToken` API 호출의 세션 정책을 전달하는 것입니다. 앞의 예시를 확장해 IAM 사용자 `token-app`의 자격 증명을 사용하여 `GetFederationToken` 호출이 이루어진다고 가정합니다. 그런 다음 세션 정책이 API 호출의 파라미터로 전달된다고 가정합니다. 결과적으로 페더레이션 사용자는 이름이 `productionapp`인 Amazon S3 버킷의 콘텐츠를 나열할 권한을 갖습니다. 사용자는 `productionapp` 버킷의 항목들에 대한 `GetObject`, `PutObject`, Amazon S3 및 `DeleteObject` 작업을 수행할 수 없습니다.

페더레이션 사용자에게 이 권한이 할당되는 것은 권한이 IAM 사용자 정책과 전달한 세션 정책의 교차 지점이기 때문입니다.

페더레이션 사용자는 Amazon SNS, Amazon SQS, Amazon DynamoDB 또는 S3 버킷 (`productionapp` 제외)에서 작업을 수행할 수 없습니다. 이러한 작업은 관련 권한이 `GetFederationToken` 호출과 연결된 IAM 사용자에게 부여되었더라도 거부됩니다.

Example **GetFederationToken** API 호출의 파라미터로 전달된 세션 정책의 예

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::productionapp"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": ["arn:aws:s3:::productionapp/*"]
    }
  ]
}
```

리소스 기반 정책

일부 AWS 리소스는 리소스 기반 정책을 지원하고, 이 정책은 페더레이션 사용자에게 직접 권한을 부여하는 또 다른 메커니즘을 제공합니다. 일부 AWS 서비스만이 리소스 기반 정책을 지원합니다. 예를 들어, Amazon S3에는 버킷이 있고, Amazon SNS에는 주제가 있고, Amazon SQS에는 정책을 연결할 수 있는 대기열이 있습니다. 리소스 기반 정책을 지원하는 모든 서비스 목록은 AWS IAM으로 작업하는 서비스 및 표의 "리소스 기반 정책" 열을 참조하세요. 리소스 기반 정책을 사용하여 페더레이션 사용자에게 직접 권할을 할당할 수 있습니다. 리소스 기반 정책의 `Principal` 요소에서 페더레이션 사용자의 Amazon 리소스 이름(ARN)을 지정하면 됩니다. 다음 예에서는 이를 설명하고 이름이 `productionapp`인 S3 버킷을 사용하여 앞의 예시를 확장합니다.

다음 리소스 기반 정책은 버킷에 연결되어 있습니다. 이 버킷 정책은 Carol이라는 페더레이션 사용자가 버킷에 액세스할 수 있도록 허용합니다. 다음 리소스 기반 정책이 적용되고 앞서 기술된 예시 정책이 `token-app` IAM 사용자에 연결되어 있으면, Carol이라는 페더레이션 사용자는 `productionapp`라는 버킷에 대해 `s3:GetObject`, `s3:PutObject` 및 `s3:DeleteObject` 작업을 수행할 수 있는 권한이 있습니다. 이는 `GetFederationToken` API 호출의 파라미터로 전달되는 세션 정책이 없을 때에도 해당됩니다. 왜냐하면 이 경우에 Carol이라는 페더레이션 사용자는 다음 리소스 기반 정책에 의해 명시적으로 권한을 부여받았기 때문입니다.

그 권한이 IAM 사용자 및 페더레이션 사용자 둘 다에게 명시적으로 부여될 때만 페더레이션 사용자는 권한을 부여받는다는 것을 명심하세요. 다음 예제처럼 정책의 `Principal` 요소에서 페더레이션 사용자의 이름을 명시적으로 지정하는 리소스 기반 정책을 통해서도 계정 내에서 권한을 부여할 수 있습니다.

Example 페더레이션 사용자에 대한 액세스를 허용하는 버킷 정책의 예

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": {"AWS": "arn:aws:sts::account-id:federated-user/Carol"},
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:PutObject",
      "s3:DeleteObject"
    ],
    "Resource": ["arn:aws:s3:::productionapp/*"]
  }
}
```

정책이 평가되는 방식에 대한 자세한 내용을 알아보려면 정책 평가 로직을 참조하세요.

GetSessionToken에 대한 권한

`GetSessionToken` API 작업 또는 `get-session-token` CLI 명령을 호출해야 하는 기본적인 경우는 사용자가 멀티 팩터 인증(MFA)으로 인증되어야 할 때입니다. MFA로 인증된 사용자가 요청하는 경우에 한해 특정 작업들을 허용하는 정책을 작성하는 것도 가능합니다. MFA 권한 부여 확인을 성공적으로 통과하려면 사용자는 먼저 `GetSessionToken`을 호출하여 선택 사항인 `SerialNumber` 및 `TokenCode` 파라미터를 포함해야 합니다. 사용자가 MFA 디바이스를 통해 인증을 받으면 `GetSessionToken` API 작업에서 반환하는 자격 증명에는 MFA 컨텍스트가 포함됩니다. 이 컨텍스트에서는 사용자가 MFA 디바이스를 통해 인증을 받았고 MFA 인증이 필요한 API 작업에 대한 권한이 있음을 표시합니다.

GetSessionToken에 필요한 권한

사용자는 권한이 없어도 세션 토큰을 얻을 수 있습니다. `GetSessionToken` 작업의 목적은 MFA를 사용하는 사용자를 인증하는 것입니다. 정책을 사용하여 인증 작업을 제어할 수는 없습니다.

대부분의 AWS 작업을 수행할 수 있는 권한을 부여하려면 이름이 같은 작업을 정책에 추가합니다. 예를 들어 사용자를 생성하려면 `CreateUser` API 작업, `create-user` CLI 명령 또는 AWS

Management Console을 사용해야 합니다. 이러한 작업을 수행하려면 `CreateUser` 작업에 액세스할 수 있게 허용하는 정책이 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateUser",
            "Resource": "*"
        }
    ]
}
```

정책에 `GetSessionToken` 작업을 포함할 수 있지만, 사용자가 `GetSessionToken` 작업을 수행할 수 있는 권한에는 영향을 미치지 않습니다.

GetSessionToken에서 부여하는 권한

`GetSessionToken`이 IAM 사용자의 자격 증명으로 호출되면, 임시 보안 자격 증명은 IAM 사용자와 동일한 권한을 갖습니다. 마찬가지로 `GetSessionToken`이 AWS 계정 루트 사용자 보안 인증 정보로 호출되면, 임시 보안 자격 증명은 루트 사용자 권한을 갖습니다.

> **Note**
>
> 루트 사용자 자격 증명으로 `GetSessionToken`을 호출하지 않는 것이 좋습니다. 대신에 모범 사례에 따라 필요한 권한을 지닌 IAM 사용자를 생성하세요. 그런 다음 이러한 IAM 사용자를 AWS와의 일상적인 상호 작용에 사용하세요.

`GetSessionToken`을 호출할 때 얻는 임시 자격 증명은 다음과 같은 기능과 한계를 지닙니다.

- `https://signin.aws.amazon.com/federation`에서 페더레이션 Single Sign-On 엔드포인트로 자격 증명을 전달하여 AWS Management Console에 액세스할 수 있습니다. 자세한 내용은 사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기 단원을 참조하십시오.
- 자격 증명을 사용해 IAM 또는 AWS STS API 작업을 호출할 수 없습니다. 자격 증명을 사용해 다른 서비스에 대한 API 작업을 호출할 수는 있습니다.AWS

AWS STS API 작업 비교에서 이 API 작업과 이 작업의 한계 및 기능을 임시 보안 자격 증명을 생성하는 다른 API와 비교해 보세요.

GetSessionToken을 사용한 MFA 보호 API 액세스에 대한 자세한 내용은 MFA를 통한 보안 API 액세스 섹션을 참조하세요.

임시 보안 자격 증명에 대한 권한 비활성화

임시 보안 인증 정보는 만료될 때까지 유효합니다. 이러한 보안 인증 정보는 900초(15분)부터 최대 129,600초(36시간)까지 지정된 기간 동안 유효합니다. 기본 세션 기간은 43,200초(12시간)입니다. 이러한 보안 인증 정보는 취소할 수 있지만, 보안 인증 정보가 도용되어 악의적인 계정 활동에 사용되지 않도록 하려면 역할의 권한도 변경해야 합니다. 임시 보안 인증 정보에 할당된 권한은 AWS 요청을 위해 사용될 때마다 평가됩니다. 보안 인증 정보에서 모든 권한을 제거하면 이를 사용하는 AWS 요청은 실패합니다.

정책 업데이트가 적용되는 데 몇 분 정도 걸릴 수 있습니다. 역할의 임시 보안 인증 정보를 취소하여 역할을 수임하는 모든 사용자에게 재인증과 새 보안 인증 정보 요청을 강제합니다.

AWS 계정 루트 사용자에 대한 권한은 변경할 수 없습니다. 따라서 루트 사용자로 로그인할 때 GetFederationToken 또는 GetSessionToken을 호출하여 생성된 임시 보안 자격 증명에 대한 권한도 변경할 수 없습니다. 이런 이유 때문에 루트 사용자로 GetFederationToken 또는 GetSessionToken을 호출하지 않는 것이 좋습니다.

> **Important**
>
> IAM Identity Center 권한 세트에서 생성된 역할은 IAM에서 편집할 수 없습니다. IAM Identity Center에서 사용자의 활성 권한 세트 세션을 취소해야 합니다. 자세한 내용은 IAM Identity Center 사용 설명서의 권한 세트에 의해 생성된 활성 IAM 역할 세션 취소를 참조하세요.

주제

- 역할과 관련된 모든 세션에 대한 액세스 거부
- 특정 세션에 대한 액세스 거부
- 조건 컨텍스트 키로 사용자 세션 거부
- 리소스 기반 정책으로 세션 사용자 거부

역할과 관련된 모든 세션에 대한 액세스 거부

다음에 의한 의심스러운 액세스가 우려되는 경우 이 방법을 사용합니다.

- 크로스 계정 액세스를 사용하는 다른 계정의 보안 주체

- 계정 내 AWS 리소스에 액세스할 권한이 있는 외부 사용자 자격 증명
- OIDC 공급자를 통해 모바일 또는 웹 애플리케이션에서 인증된 사용자

이 절차는 역할을 수임할 권한이 있는 모든 사용자에 대한 권한을 거부합니다.

`AssumeRole`, `AssumeRoleWithSAML`, `AssumeRoleWithWebIdentity`, `GetFederationToken` 또는 `GetSessionToken`을 직접적으로 호출하여 획득한 임시 보안 인증 정보에 할당된 권한을 변경하거나 제거하려면 역할에 대한 권한을 정의하는 권한 정책을 편집 또는 삭제하면 됩니다.

> **Important**
>
> 보안 주체 액세스를 허용하는 리소스 기반 정책이 있는 경우 해당 리소스에 대한 명시적 거부도 추가해야 합니다. 세부 정보는 리소스 기반 정책으로 세션 사용자 거부를 참조하세요.

1. AWS Management Console에 로그인하고 IAM 콘솔을 엽니다.
2. 탐색 창에서 편집하려는 역할의 이름을 선택합니다. 검색 상자를 사용하여 목록을 필터링할 수 있습니다.
3. 관련 정책을 선택합니다.
4. 권한 탭을 선택합니다.
5. JSON 탭을 선택하고 정책을 업데이트하여 모든 리소스 및 작업을 거부합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

6. 검토 페이지에서 정책 요약을 검토하고 나서 변경 사항 저장을 선택하여 작업을 저장합니다.

정책을 업데이트하면 이러한 변경은 역할의 권한 정책을 변경하기 전에 발급된 보안 인증 정보를 비롯해 해당 역할에 연결된 모든 임시 보안 인증 정보의 권한에 영향을 미칩니다. 정책을 업데이트한 후 역

할의 임시 보안 인증 정보를 취소하여 역할의 발급된 보안 인증 정보에 대한 모든 권한을 즉시 취소할 수 있습니다.

특정 세션에 대한 액세스 거부

모두 거부 정책을 사용하여 IdP에서 수임할 수 있는 역할을 업데이트하거나 역할을 완전히 삭제하면 해당 역할에 액세스할 수 있는 모든 사용자의 액세스가 중단됩니다. 역할과 연결된 다른 모든 세션의 권한에 영향을 미치지 않고 `Principal` 요소를 기반으로 액세스를 거부할 수 있습니다.

`Principal`에 대한 권한은 조건 컨텍스트 키 또는 리소스 기반 정책을 사용하여 거부될 수 있습니다.

> **Tip**
>
> AWS CloudTrail 로그를 사용하여 페더레이션 사용자의 ARN을 찾을 수 있습니다. 자세한 내용은 How to Easily Identify Your Federated Users by Using AWS CloudTrail을 참조하세요.

조건 컨텍스트 키로 사용자 세션 거부

보안 인증 정보를 생성한 IAM 사용자 또는 역할의 권한에 영향을 미치지 않고 임시 보안 인증 정보에 대한 액세스를 거부하고자 하는 상황에서 조건 컨텍스트 키를 사용할 수 있습니다.

조건 컨텍스트 키에 대한 자세한 내용은 AWS 글로벌 조건 컨텍스트 키 섹션을 참조하세요.

> **Note**
>
> 보안 주체 액세스를 허용하는 리소스 기반 정책이 있는 경우 이 단계를 완료한 후 리소스 기반 정책에 명시적 거부 문도 추가해야 합니다.

정책을 업데이트한 후 역할의 임시 보안 인증 정보를 취소하여 모든 발급된 보안 인증 정보를 즉시 취소할 수 있습니다.

aws:PrincipalArn

조건 컨텍스트 키 aws:PrincipalArn을 사용하여 특정 보안 주체 ARN에 대한 액세스를 거부할 수 있습니다. IAM 사용자의 고유 식별자(ID), 역할 또는 페더레이션 사용자를 지정하면 정책의 Condition 요소에 임시 보안 인증 정보가 연결됩니다.

1. IAM 콘솔 탐색 창에서 편집하려는 역할의 이름을 선택합니다. 검색 상자를 사용하여 목록을 필터링할 수 있습니다.

2. 관련 정책을 선택합니다.
3. 권한 탭을 선택합니다.
4. JSON 탭을 선택하고 다음 예제와 같이 보안 주체 ARN에 대한 거부 문을 추가합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "aws:PrincipalArn": [
            "arn:aws:iam::222222222222:role/ROLENAME",
            "arn:aws:iam::222222222222:user/USERNAME",
            "arn:aws:sts::222222222222:federated-user/USERNAME"
          ]
        }
      }
    }
  ]
}
```

5. 검토 페이지에서 정책 요약을 검토하고 나서 변경 사항 저장을 선택하여 작업을 저장합니다.

aws:SourceIdentity

조건 컨텍스트 키 aws:SourceIdentity 사용을 통해 역할 세션과 연결된 특정 소스 ID에 대한 액세스를 거부할 수 있습니다. 이는 보안 주체가 AWS STS `assume-role*` CLI 명령 또는 AWS STS `AssumeRole*` API 작업을 사용하여 역할을 수임할 때 `SourceIdentity` 요청 파라미터를 설정하여 역할 세션이 실행된 경우에만 적용됩니다. 정책의 `Condition` 요소에 임시 보안 자격 증명이 연결된 소스 ID를 지정하여 이를 수행할 수 있습니다.

컨텍스트 키 sts:RoleSessionName과 달리 소스 ID가 설정된 후에는 값을 변경할 수 없습니다. `aws:SourceIdentity` 키는 역할에서 수행되는 모든 작업의 요청 컨텍스트에 있습니다. 세션 자격 증명을 사용하여 다른 역할을 수임하는 경우 소스 ID가 후속 역할 세션에 유지됩니다. 한 역할에서 다른 역할을 맡는 것을 역할 체인이라고 합니다.

다음 정책은 조건 컨텍스트 키 `aws:SourceIdentity`를 사용하여 임시 보안 인증 정보 세션에 대한 액세스를 거부하는 방법의 예를 보여줍니다. 역할 세션과 연결된 소스 ID를 지정하면 자격 증명을

생성한 역할의 권한에 영향을 미치지 않으면서 명명된 소스 ID가 포함된 역할 세션을 거부합니다. 이 예제에서 역할 세션이 실행되었을 때 보안 주체가 설정한 소스 ID는 `nikki_wolf@example.com`입니다. 소스 ID가 정책 조건에 포함되고 정책 효과가 `Deny`로 설정되어 있기 때문에 소스 ID `nikki_wolf@example.com`을 포함하는 역할 세션의 요청은 거부됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:SourceIdentity": [
            "nikki_wolf@example.com",
            "<source identity value>"
          ]
        }
      }
    }
  ]
}
```

aws:userid

조건 컨텍스트 키 aws:userid를 사용하여 IAM 사용자 또는 역할에 연결된 모든 또는 특정 임시 보안 인증 정보에 대한 액세스를 거부할 수 있습니다. IAM 사용자의 고유 식별자(ID), 역할 또는 페더레이션 사용자를 지정하면 정책의 `Condition` 요소에 임시 보안 인증 정보가 연결됩니다.

다음 정책은 조건 컨텍스트 키 `aws:userid`를 사용하여 임시 보안 인증 정보 세션에 대한 액세스를 거부하는 방법의 예를 보여줍니다.

- `AIDAXUSER1`은 IAM 사용자의 고유 식별자를 나타냅니다. IAM 사용자의 고유 식별자를 컨텍스트 키 `aws:userid`의 값으로 지정하면 IAM 사용자와 연결된 모든 세션을 거부합니다.
- `AROAXROLE1`은 IAM 역할의 고유 식별자를 나타냅니다. IAM 역할의 고유 식별자를 컨텍스트 키 `aws:userid`의 값으로 지정하면 역할과 연결된 모든 세션을 거부합니다.
- `AROAXROLE2:<caller-specified-role-session-name>`는 수임된 역할 세션의 고유 식별자를 나타냅니다. 수임된 역할 고유 식별자의 호출자 지정 역할 세션 이름 부분에서 StringLike 조건 연산자가 사용되는 경우 역할 세션 이름 또는 와일드카드 문자를 지정할 수 있습니다. 역할 세션 이름

을 지정하면 보안 인증 정보를 생성한 역할의 권한에 영향을 미치지 않으면서 명명된 역할 세션을 거부합니다. 역할 세션 이름에 와일드카드를 지정하면 해당 역할과 연결된 모든 세션을 거부합니다.

> **Note**
>
> 수임한 역할 세션의 고유 식별자에 속하는 호출자 지정 역할 세션 이름은 역할 체인 중에 변경될 수 있습니다. 역할 체인은 한 역할이 다른 역할을 수임할 때 발생합니다. 보안 주체가 AWS STS `AssumeRole` API 작업을 사용하여 역할을 수임할 때 `RoleSessionName` 요청 파라미터를 사용하여 역할 세션 이름이 설정됩니다.

- `account-id:<federated-user-caller-specified-name>`은 페더레이션 사용자 세션의 고유 식별자를 나타냅니다. 페더레이션 사용자는 GetFederationToken API를 직접적으로 호출하는 IAM 사용자에 의해 생성됩니다. 페더레이션 사용자의 고유 식별자를 지정한 경우 보안 인증 정보를 생성한 역할의 권한에 영향을 미치지 않으면서 명명된 페더레이션 사용자 세션을 거부합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:userId": [
            "AIDAXUSER1",
            "AROAXROLE1",
            "AROAXROLE2:<caller-specified-role-session-name>",
            "account-id:<federated-user-caller-specified-name>"
          ]
        }
      }
    }
  ]
}
```

key values 키 값의 구체적인 예제는 보안 주체 키 값 섹션을 참조하세요. IAM 고유 식별자에 대한 자세한 내용은 고유 식별자 섹션을 참조하세요.

리소스 기반 정책으로 세션 사용자 거부

보안 주체 ARN이 리소스 기반 정책에도 포함된 경우 리소스 기반 정책의 `Principal` 요소에 있는 특정 사용자의 `principalId` 또는 `sourceIdentity` 값을 기반으로 액세스를 취소해야 합니다. 역할에 대한 권한 정책만 업데이트하는 경우 사용자는 리소스 기반 정책에서 허용된 작업을 계속 수행할 수 있습니다.

1. 서비스가 리소스 기반 정책을 지원하는지 여부는 AWS IAM으로 작업하는 서비스에서 참조하세요.
2. AWS Management Console에 로그인하고 서비스에 대한 콘솔을 엽니다. 각 서비스는 콘솔에서 정책을 연결하는 위치가 다릅니다.
3. 정책 문을 편집하여 보안 인증 정보의 식별 정보를 지정합니다.
   a. `Principal`에서 거부할 보안 인증 정보의 ARN을 입력합니다.
   b. `Effect`에서 'Deny'를 입력합니다.
   c. `Action`에서 거부할 서비스 네임스페이스와 작업 이름을 입력합니다. 모든 작업을 거부하려면 와일드카드 (*) 문자를 사용합니다. 예: 's3:*.'
   d. `Resource`에서 대상 리소스의 ARN을 입력합니다. 예: "arn:aws:s3:::EXAMPLE-BUCKET."

```
{
"Version": "2012-10-17",
  "Statement": {
    "Principal": [
            "arn:aws:iam::222222222222:role/ROLENAME",
            "arn:aws:iam::222222222222:user/USERNAME",
            "arn:aws:sts::222222222222:federated-user/USERNAME"
    ],
    "Effect": "Deny",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::EXAMPLE-BUCKET"
  }
}
```

4. 작업을 저장합니다.

임시 보안 자격 증명을 생성할 수 있는 권한 부여

기본적으로 IAM 사용자는 페더레이션 사용자 및 역할을 위한 임시 보안 자격 증명을 생성할 수 있는 권한이 없습니다. 사용자에게 이러한 권한을 제공하려면 정책을 사용해야 합니다. 사용자에게 직접 권한을 부여할 수 있지만, 그룹에게 권한을 부여할 것을 강력히 권고합니다. 그렇게 하면 권한 관리가 훨씬 쉬워집니다. 어떤 사용자가 권한에 연결된 작업을 수행할 필요가 더 이상 없는 경우에는 그룹에서 그 사용자를 삭제하기만 하면 됩니다. 다른 어떤 사용자가 그 작업을 수행해야 한다면 해당 그룹에 추가해 권한을 부여하면 됩니다.

페더레이션 사용자 또는 역할을 위해 임시 보안 자격 증명을 생성할 수 있는 권한을 IAM 그룹에 부여하려면 다음 권한 중 하나 또는 둘 다를 부여하는 정책을 연결하면 됩니다.

- 페더레이션 사용자들이 IAM 역할에 액세스하도록 하려면 AWS STS `AssumeRole`에 대한 액세스 권한을 부여하세요.
- 역할이 필요 없는 페더레이션 사용자에 대해서는 AWS STS `GetFederationToken`에 대한 액세스 권한을 부여하십시오.

`AssumeRole` 및 `GetFederationToken` API 작업 간의 차이점을 보려면 임시 보안 자격 증명 요청 섹션을 참조하세요.

IAM 사용자는 `GetSessionToken`을 호출하여 임시 보안 자격 증명을 생성할 수도 있습니다. 사용자는 권한이 없어도 `GetSessionToken`을 호출할 수 있습니다. 이 작업의 목적은 MFA를 사용하는 사용자를 인증하는 것입니다. 정책을 사용하여 인증을 제어할 수는 없습니다. 즉 IAM 사용자가 임시 자격 증명을 생성할 목적으로 `GetSessionToken`을 호출하는 작업을 하지 못하게 할 수 없습니다.

Example 역할 수임 권한을 부여하는 정책 예

다음 정책 예는 AWS 계정 `123123123123`의 `UpdateApp` 역할을 위해 `AssumeRole`을 호출할 수 있는 권한을 부여합니다. `AssumeRole`을 사용하는 경우, 페더레이션 사용자를 대신해 보안 자격 증명을 생성하는 사용자(또는 애플리케이션)는 역할 권한 정책에 아직 지정되지 않은 어떤 권한도 위임할 수 없습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::123123123123:role/UpdateAPP"
  }]
```

```
}
```

Example 페더레이션 사용자를 위한 임시 보안 자격 증명을 생성할 수 있는 권한을 부여하는 정책 예

다음과 같은 정책 예시는 GetFederationToken에 액세스할 수 있는 권한을 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:GetFederationToken",
    "Resource": "*"
  }]
}
```

> **Important**
>
> IAM 사용자에게 GetFederationToken을 사용해 페더레이션 사용자를 위한 임시 보안 자격 증명을 생성할 수 있는 권한을 부여하면 이로써 해당 사용자가 자신의 권한을 위임할 수 있게 허용하는 것이므로 주의하시기 바랍니다. 여러 IAM 사용자 및 AWS 계정에 걸쳐 권한을 위임하는 것에 대한 자세한 내용은 액세스 권한 위임을 위한 정책의 예 섹션을 참조하세요. 임시 보안 자격 증명에서 권한을 제어하는 것에 대한 자세한 정보는 사용자 임시 보안 자격 증명에 대한 권한 제어 섹션을 참조하세요.

Example 페더레이션 사용자에 대해 임시 보안 자격 증명을 생성할 수 있는 사용자 제한 권한을 부여하는 정책 예

IAM 사용자가 GetFederationToken을 호출하도록 허용할 때는 IAM 사용자가 위임할 수 있는 권한을 제한하는 것이 좋습니다. 예를 들어 다음 정책은 IAM 사용자가 이름이 Manager로 시작하는 페더레이션 사용자에 대해서만 임시 보안 자격 증명을 생성하도록 허용하는 방법을 보여줍니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:GetFederationToken",
    "Resource": ["arn:aws:sts::123456789012:federated-user/Manager*"]
  }]
}
```

ID 인식 콘솔 세션을 사용하는 권한 부여

ID 인식 콘솔 세션을 사용하면 AWS IAM Identity Center 사용자가 로그인할 때 사용자 및 세션 ID를 사용자의 AWS 콘솔 세션에 포함할 수 있습니다. 예를 들어 Amazon Q Developer Pro는 ID 인식 콘솔 세션을 사용하여 서비스 경험을 개인화합니다. ID 인식 콘솔 세션에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 ID 인식 콘솔 세션 활성화를 참조하세요. Amazon Q 개발자 설정에 대한 자세한 내용은 Amazon Q 개발자 사용 설명서의 Amazon Q 개발자 설정을 참조하세요.

사용자가 ID 인식 콘솔 세션을 사용하려면 ID 기반 정책을 사용하여 IAM 보안 주체에게 자신의 콘솔 세션을 나타내는 리소스에 대한 `sts:SetContext` 권한을 부여해야 합니다.

> **Important**
>
> 기본적으로 사용자는 ID 인식 콘솔 세션의 컨텍스트를 설정할 권한이 없습니다. 이를 허용하려면 아래 정책 예와 같이 ID 기반 정책에서 IAM 보안 주체에게 `sts:SetContext` 권한을 부여해야 합니다.

다음 예제의 ID 기반 정책은 IAM 보안 주체에 `sts:SetContext` 권한을 부여하여 보안 주체가 자신의 AWS 콘솔 세션에 대해 ID 인식 콘솔 세션 컨텍스트를 설정할 수 있도록 합니다. 정책 리소스 `arn:aws:sts::account-id:self`는 호출자의 AWS 세션을 나타냅니다. IAM Identity Center 권한 세트를 사용하여 정책을 배포하는 경우와 같이 여러 계정에 동일한 권한 정책을 배포하는 경우 `account-id` ARN 세그먼트를 와일드카드 문자(*)로 바꿀 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:SetContext",
      "Resource": "arn:aws:sts::account-id:self"
    }
  ]
}
```

AWS STS에서 AWS 리전 관리

기본적으로 AWS Security Token Service(AWS STS)는 글로벌 서비스로 사용 가능하고 모든 AWS STS 요청은 `https://sts.amazonaws.com`의 단일 엔드포인트로 전송됩니다. AWS는 지연 시간을

줄이고, 중복으로 구축하고, 세션 토큰 유효성을 높이려면 전역 엔드포인트 대신 리전별 AWS STS 엔드포인트를 사용할 것을 권장합니다.

- 지연 시간 감소 - 서비스 및 애플리케이션에서 지리적으로 더 가까운 엔드포인트에 AWS STS 호출을 함으로써 지연 시간 및 응답 시간을 단축하며 AWS STS 서비스에 액세스할 수 있습니다.
- 중복 구축 - 워크로드 내에서 발생하는 장애의 영향을 예측 가능한 영향 억제 범위를 가진 한정된 수의 구성 요소로 제한할 수 있습니다. 리전별 AWS STS 엔드포인트를 사용하면 구성 요소의 범위를 세션 토큰의 범위에 맞출 수 있습니다. 이 신뢰성 요소에 대한 자세한 내용은 AWS Well-Architected Framework의 장애 격리를 사용하여 워크로드 보호를 참조하세요.
- 세션 토큰 유효성 증가 - 리전별 AWS STS 엔드포인트의 세션 토큰은 모든 AWS 리전에서 유효합니다. 전역 STS 엔드포인트의 세션 토큰은 기본적으로 STS가 활성화된 AWS 리전에서만 유효합니다. 계정에 대해 새 리전을 활성화하려는 경우 리전별 AWS STS 엔드포인트의 세션 토큰을 사용할 수 있습니다. 전역 엔드포인트를 사용하는 경우 전역 엔드포인트에 대한 AWS STS 세션 토큰의 리전 호환성을 변경해야 합니다. 이를 통해 토큰이 모든 AWS 리전에서 유효합니다.

전역 엔드포인트 세션 토큰 관리

대부분의 AWS 리전은 기본적으로 모든 AWS 서비스의 작업이 활성화되어 있습니다. 이러한 리전은 AWS STS 사용이 자동으로 활성화됩니다. 아시아 태평양(홍콩)과 같은 일부 리전은 수동으로 활성화해야 합니다. AWS 리전 활성화 및 비활성화에 대한 자세한 내용은 AWS Account Management 참조 안내서의 계정에서 사용할 수 있는 AWS 리전 지정을 참조하세요. 이러한 AWS 리전을 활성화할 때 자동으로 AWS STS 사용이 활성화됩니다. 비활성화된 리전에 대한 AWS STS 엔드포인트를 활성화할 수는 없습니다. 모든 AWS 리전에서 유효한 세션 토큰에는 기본적으로 활성화된 리전에서 유효한 토큰보다 많은 문자가 포함되어 있습니다. 이 설정을 변경하면 토큰을 임시로 저장한 기존 시스템에 영향을 미칠 수 있습니다.

AWS Management Console, AWS CLI 또는 AWS API를 사용하여 이 설정을 변경할 수 있습니다.

전역 엔드포인트에 대한 세션 토큰의 리전 호환성 변경(콘솔)

1. 루트 사용자 또는 IAM 관리 작업을 수행할 권한이 있는 사용자로 로그인합니다. 세션 토큰의 호환성을 변경하려면 `iam:SetSecurityTokenServicePreferences` 작업을 허용하는 정책이 있어야 합니다.
2. IAM 콘솔(IAM console)을 엽니다. 탐색 창에서 계정 설정(Account settings)를 선택합니다.
3. Security Token Service (STS)(Security Token Service (STS)) 섹션의 Session Tokens from the STS endpoints(STS 엔드포인트의 세션 토큰). Global endpoint(글로벌 엔드포인트)는 `Valid only in AWS ## enabled by default`를 나타냅니다. 변경을 선택합니다.

4. Change region compatibility(리전 호환성 변경) 대화 상자에서 All AWS 리전를 선택합니다. 변경 사항 저장(Save changes)을 선택합니다.

> **Note**
>
> 모든 AWS 리전에서 유효한 세션 토큰에는 기본적으로 활성화된 리전에서 유효한 토큰보다 많은 문자가 포함되어 있습니다. 이 설정을 변경하면 토큰을 임시로 저장한 기존 시스템에 영향을 미칠 수 있습니다.

전역 엔드포인트에 대한 세션 토큰의 리전 호환성 변경(AWS CLI)

세션 토큰 버전을 설정합니다. 버전 1 토큰은 기본적으로 이용 가능한 AWS 리전에서만 유효합니다. 이러한 토큰은 아시아 태평양(홍콩)과 같이 수동으로 활성화된 리전에서 작동하지 않습니다. 버전 2는 모든 리전에서 유효합니다. 하지만 버전 2 토큰에 포함된 문자가 더 많고 일시적으로 토큰을 저장하는 시스템에 영향을 미칠 수 있습니다.

- `aws iam set-security-token-service-preferences`

전역 엔드포인트에 대한 세션 토큰의 리전 호환성 변경(AWS API)

세션 토큰 버전을 설정합니다. 버전 1 토큰은 기본적으로 이용 가능한 AWS 리전에서만 유효합니다. 이러한 토큰은 아시아 태평양(홍콩)과 같이 수동으로 활성화된 리전에서 작동하지 않습니다. 버전 2는 모든 리전에서 유효합니다. 하지만 버전 2 토큰에 포함된 문자가 더 많고 일시적으로 토큰을 저장하는 시스템에 영향을 미칠 수 있습니다.

- `SetSecurityTokenServicePreferences`

AWS 리전에서 AWS STS 활성화 및 비활성화

리전에 대한 STS 엔드포인트를 활성화할 때 AWS STS에서 AWS STS 요청을 수행하는 계정의 사용자 및 역할에 임시 자격 증명을 발급할 수 있습니다. 이러한 자격 증명은 기본적으로 활성화되었거나 수동으로 활성화된 모든 리전에서 사용할 수 있습니다. 기본적으로 활성화된 리전의 경우, 임시 보안 인증 정보가 생성되는 계정에서 리전별 STS 엔드포인트를 활성화해야 합니다. 요청 시 사용자가 동일한 계정으로 로그인하거나 각기 다른 계정으로 로그인하는지 여부는 중요하지 않습니다. 수동으로 활성화된 리전의 경우, 요청을 수행하는 계정과 임시 보안 인증 정보가 생성되는 계정 모두에서 리전을 활성화해야 합니다.

예를 들어, 계정 A의 사용자가 AWS STS 리전 엔드포인트 `https://sts.ap-east-1.amazonaws.com`으로 `sts:AssumeRole` API 요청을 보내려 할 수 있습니다. 이는 계정 B의 `Developer` 역할에 대한 임시 자격 증명을 요청하기 위한 것입니다. 이 요청은 계정 B의 엔터티에 대한 자격 증명을 만들기 위한 것이기 때문에 계정 B는 `ap-east-1` 리전을 활성화해야 합니다. 계정 A(또는 다른 계정)의 사용자는 자신의 계정에서 리전이 활성화되었는지 여부와 상관없이 `ap-east-1` AWS STS 엔드포인트를 호출하여 계정 B의 자격 증명을 요청할 수 있습니다.

> **Note**
>
> 활성 리전은 해당 계정의 임시 자격 증명을 이용하는 모두가 이용할 수 있습니다. 어떤 IAM 사용자 또는 역할이 리전에 액세스할 수 있는지 여부를 제어하려면 권한 정책에서 `aws:RequestedRegion` 조건 키를 사용합니다.

기본적으로 활성화된 리전에서 AWS STS 활성화 또는 비활성화(콘솔)

1. 루트 사용자 또는 IAM 관리 작업을 수행할 권한이 있는 사용자로 로그인합니다.
2. IAM 콘솔을 열고 탐색 창에서 계정 설정(Account Settings)을 선택합니다.
3. Security Token Service (STS)(Security Token Service(STS)) 섹션의 Endpoints(엔드포인트)에서 구성하려는 리전을 찾은 다음 STS status(STS 상태) 열에서 Active(활성) 또는 Inactive(비활성)를 선택합니다.
4. 열리는 대화 상자에서 Activate(활성화) 또는 Deactivate(비활성화)를 선택합니다.

활성화해야 하는 리전의 경우 리전을 활성화하면 AWS STS가 자동으로 활성화됩니다. 리전을 활성화한 후에는 AWS STS가 해당 리전에 대해 항상 활성화되어 있으며 비활성화할 수 없습니다. 기본적으로 비활성화된 리전을 활성화하는 방법에 대해 알아보려면 AWS Account Management 참조 안내서의 계정에서 사용할 수 있는 AWS 리전 지정을 참조하세요.

AWS STS 리전 사용을 위한 코드 작성

리전을 활성화한 후에 AWS STS API 호출을 그 리전으로 보낼 수 있습니다. 다음 Java 코드 조각은 `AWSSecurityTokenService` 객체를 구성해 유럽(밀라노)(eu-south-1) 리전으로 요청하는 방법을 보여줍니다.

```
EndpointConfiguration regionEndpointConfig = new EndpointConfiguration("https://sts.eu-south-1.amazonaws.com", "eu-south-1");
AWSSecurityTokenService stsRegionalClient =
 AWSSecurityTokenServiceClientBuilder.standard()
```

```
.withCredentials(credentials)
.withEndpointConfiguration(regionEndpointConfig)
.build();
```

AWS STS에서는 리전 엔드포인트에 호출할 것을 권장합니다. 리전을 수동으로 활성화하는 방법에 대해 알아보려면 AWS Account Management 참조 안내서의 계정에서 사용할 수 있는 AWS 리전 지정을 참조하세요.

이 예에서 첫 번째 줄은 `regionEndpointConfig`라는 `EndpointConfiguration` 객체를 인스턴스화하여 엔드포인트의 URL과 AWS 리전을 파라미터로 전달합니다.

AWS SDK의 환경 변수를 사용해 AWS STS 리전 엔드포인트를 설정하는 방법은 AWS SDK 및 도구 참조 가이드의 AWS STS 리전화된 엔드포인트를 참조하세요.

다른 모든 언어 및 프로그래밍 환경의 조합에 대해서는 해당 SDK 문서를 참조하세요.

리전 및 엔드포인트

다음 표에서는 해당 리전과 그 엔드포인트를 나열합니다. 기본적으로 어떤 것들이 활성화되며, 어떤 것을 활성화 또는 비활성화할 수 있는지를 보여줍니다.

지역명	엔드포인트	기본 활성화	수동으로 활성화/비활성화
--전역--	sts.amazonaws.com	 예	 아니요
미국 동부(오하이오)	sts.us-east-2.amazonaws.com	 예	 예
미국 동부(버지니아 북부)	sts.us-east-1.amazonaws.com	 예	 아니요

지역명	엔드포인트	기본 활성화	수동으로 활성화/비활성화
미국 서부(캘리포니아 북부)	sts.us-west-1.amazonaws.com	 예	 예
미국 서부(오레곤)	sts.us-west-2.amazonaws.com	 예	 예
아프리카(케이프타운)	sts.af-south-1.amazonaws.com	 아니요[1]	 아니요
아시아 태평양(홍콩)	sts.ap-east-1.amazonaws.com	 아니요[1]	 아니요
아시아 태평양(하이데라바드)	sts.ap-south-2.amazonaws.com	 아니요[1]	 아니요
아시아 태평양(자카르타)	sts.ap-southeast-3.amazonaws.com	 아니요[1]	 아니요

지역명	엔드포인트	기본 활성화	수동으로 활성화/비활성화
아시아 태평양(멜버른)	sts.ap-southeast-4.amazonaws.com	 아니요[1]	 아니요
아시아 태평양(뭄바이)	sts.ap-south-1.amazonaws.com	 예	 예
아시아 태평양(오사카)	sts.ap-northeast-3.amazonaws.com	 예	 예
아시아 태평양(서울)	sts.ap-northeast-2.amazonaws.com	 예	 예
아시아 태평양(싱가포르)	sts.ap-southeast-1.amazonaws.com	 예	 예
아시아 태평양(시드니)	sts.ap-southeast-2.amazonaws.com	 예	 예

지역명	엔드포인트	기본 활성화	수동으로 활성화/비활성화
아시아 태평양(도쿄)	sts.ap-northeast-1.amazonaws.com	 예	 예
캐나다(중부)	sts.ca-central-1.amazonaws.com	 예	 예
캐나다 서부(캘거리)	sts.ca-west-1.amazonaws.com	 예	 예
중국(베이징)	sts.cn-north-1.amazonaws.com.cn	 예[2]	 아니요
중국(닝샤)	sts.cn-northwest-1.amazonaws.com.cn	 예[2]	 예
유럽(프랑크푸르트)	sts.eu-central-1.amazonaws.com	 예	 예

지역명	엔드포인트	기본 활성화	수동으로 활성화/비활성화
유럽(아일랜드)	sts.eu-west-1.amazonaws.com	 예	 예
유럽(런던)	sts.eu-west-2.amazonaws.com	 예	 예
유럽(밀라노)	sts.eu-south-1.amazonaws.com	 아니요[1]	 아니요
유럽(파리)	sts.eu-west-3.amazonaws.com	 예	 예
유럽(스페인)	sts.eu-south-2.amazonaws.com	 아니요[1]	 아니요
유럽(스톡홀름)	sts.eu-north-1.amazonaws.com	 예	 예

지역명	엔드포인트	기본 활성화	수동으로 활성화/비활성화
유럽(취리히)	sts.eu-central-2.amazonaws.com	 아니요[1]	 아니요
이스라엘(텔아비브)	sts.il-central-1.amazonaws.com	 아니요[1]	 아니요
중동(바레인)	sts.me-south-1.amazonaws.com	 아니요[1]	 아니요
중동(UAE)	sts.me-central-1.amazonaws.com	 아니요[1]	 아니요
남아메리카(상파울루)	sts.sa-east-1.amazonaws.com	 예	 예

[1]리전을 사용하려면 리전을 활성화해야 합니다. 그러면 AWS STS가 자동으로 활성화됩니다. 이러한 리전에서는 수동으로 AWS STS를 활성화하거나 비활성화할 수 없습니다.

[2]중국에서 AWS를 사용하려면 중국 AWS와 관련된 계정 및 보안 인증이 있어야 합니다.

AWS CloudTrail 및 리전 엔드포인트

리전 및 글로벌 엔드포인트에 대한 호출은 AWS CloudTrail의 `tlsDetails` 필드에 로그인됩니다. `us-east-2.amazonaws.com`와(과) 같은 리전 및 엔드포인트에 대한 호출은 CloudTrail에서 해당 지역으로 기록됩니다. 전역적 엔드포인트 `sts.amazonaws.com`에 대한 호출은 글로벌 서비스에 대한 호출로 로깅됩니다. 글로벌 AWS STS 엔드포인트의 이벤트는 us-east-1에 기록됩니다.

> **Note**
>
> 이 필드를 지원하는 서비스에 대해서만 `tlsDetails`을(를) 볼 수 있습니다. AWS CloudTrail 사용 설명서의 CloudTrail에서 TLS 세부 정보를 지원하는 서비스를 참조하십시오 자세한 내용은 AWS CloudTrail을 사용하여 IAM 및 AWS STS API 호출 로깅 단원을 참조하십시오.

보유자 토큰 사용

일부 AWS 서비스의 경우 프로그래밍 방식으로 리소스에 액세스하기 전에 AWS STS 서비스 보유자 토큰을 가져올 수 있는 권한이 있어야 합니다. 이러한 서비스는 기존의 서명 버전 4의 서명된 요청을 사용하는 대신 보유자 토큰을 사용해야 하는 프로토콜을 지원합니다. 보유자 토큰이 필요한 AWS CLI 또는 AWS API 작업을 수행할 때 AWS 서비스는 사용자를 대신하여 보유자 토큰을 요청합니다. 이 서비스는 토큰을 제공하므로 해당 서비스에서 후속 작업을 수행하는 데 사용할 수 있습니다.

AWS STS 서비스 보유자 토큰에는 권한에 영향을 줄 수 있는 원래 보안 주체 인증 정보가 포함됩니다. 이 정보에는 보안 주체 태그, 세션 태그 및 세션 정책이 포함될 수 있습니다. 토큰의 액세스 키 ID는 `ABIA` 접두사로 시작합니다. 이렇게 하면 CloudTrail 로그에서 서비스 보유자 토큰을 사용하여 수행된 작업을 식별할 수 있습니다.

> **Important**
>
> 보유자 토큰은 이 토큰을 생성하는 서비스에 대한 호출과 이 토큰이 생성된 리전에서만 사용할 수 있습니다. 보유자 토큰을 사용하여 다른 서비스 또는 리전에서 작업을 수행할 수 없습니다.

보유자 토큰을 지원하는 서비스의 예는 AWS CodeArtifact입니다. NPM, Maven 또는 PIP와 같은 패키지 관리자를 사용하여 AWS CodeArtifact와 상호 작용하기 전에 `aws codeartifact get-authorization-token` 작업을 호출해야 합니다. 이 작업은 AWS CodeArtifact 작업을 수행하는 데 사용할 수 있는 보유자 토큰을 반환합니다. 또는 동일한 작업을 완료한 다음 클라이언트를 자동으로 구성하는 `aws codeartifact login` 명령을 사용할 수 있습니다.

AWS 서비스에서 보유자 토큰을 생성하는 작업을 수행하는 경우 IAM 정책에 다음 권한이 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowServiceBearerToken",
            "Effect": "Allow",
            "Action": "sts:GetServiceBearerToken",
            "Resource": "*"
        }
    ]
}
```

서비스 보유자 토큰의 예는 AWS CodeArtifact 사용 설명서의 AWS CodeArtifact에 대한 자격 증명 기반 정책 사용 섹션을 참조하세요.

임시 자격 증명을 사용하는 샘플 애플리케이션

AWS Security Token Service(AWS STS)를 사용하면 AWS 리소스에 대한 액세스를 제어할 수 있는 임시 보안 자격 증명을 생성하여 신뢰받는 사용자에게 제공할 수 있습니다. 에 대한 자세한 내용은 AWS STS 섹션을 참조하세요.IAM의 임시 보안 자격 증명. AWS STS를 사용해 임시 보안 자격 증명을 관리하는 방법에 대해 알아보려면, 완전한 샘플 시나리오를 구현하는 다음과 같은 샘플 애플리케이션을 다운로드하세요.

- Windows Active Directory, ADFS 및 SAML 2.0을 사용하여 AWS에 대한 페더레이션 활성화. Windows Active Directory(AD), Active Directory Federation Services(ADFS) 2.0 및 SAML(Security Assertion Markup Language) 2.0을 사용하여 엔터프라이즈 페더레이션을 사용하여 AWS에 액세스를 위임하는 방법을 보여 줍니다.
- 사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기. Single-Sign-On(SSO)을 가능케 하는 사용자 지정 페더레이션 프록시를 생성해 기존 Active Directory 사용자가 AWS Management Console에 로그인할 수 있게 하는 방법을 보여줍니다.
- AWS Management Console에 대한 Single-Sign-On(SSO)에 대해 Shibboleth를 사용하는 방법.. Shibboleth 및 SAML을 사용해 사용자에게 AWS Management Console에 대한 SSO(Single-Sign-On) 액세스 권한을 제공하는 방법을 보여줍니다.

OIDC 페더레이션의 샘플

다음 샘플 애플리케이션은 Login with Amazon, Amazon Cognito, Facebook 또는 Google 같은 공급자를 통해 OIDC 페더레이션을 사용하는 방법을 보여 줍니다. 이러한 임시 AWS 보안 자격 증명에 대해 이러한 공급자의 인증을 얻고 AWS 서비스에 액세스할 수 있습니다.

- Amazon Cognito 튜토리얼 - 모바일 개발용 AWS SDK와 함께 Amazon Cognito를 사용하는 것이 좋습니다. Amazon Cognito는 모바일 앱을 위한 자격 증명을 관리하는 가장 간단한 방법으로서, 동기화 및 교차 디바이스 자격 증명과 같은 부가 기능을 제공합니다. Amazon Cognito에 대한 자세한 내용은 Amplify 설명서의 Amplify를 사용한 인증을 참조하세요.

사용자 지정 자격 증명 브로커가 AWS 콘솔에 액세스할 수 있도록 하기

코드를 작성하고 실행해 조직 네트워크에 로그인하는 사용자가 AWS Management Console에 안전하게 액세스할 수 있게 하는 URL을 생성할 수 있습니다. URL에는 AWS에서 얻고 AWS에 사용자를 인증하는 로그인 토큰이 포함되어 있습니다. 결과 콘솔 세션에는 페더레이션으로 인해 별도의 `AccessKeyId`이(가) 포함될 수 있습니다. 관련 CloudTrail 이벤트를 통해 페더레이션 로그인을 위한 액세스 키 사용을 추적하려면 AWS CloudTrail을 사용하여 IAM 및 AWS STS API 호출 로깅 및 AWS Management Console 로그인 이벤트를 참조하세요.

> **Note**
>
> 조직에서 SAML과 호환이 되는 자격 증명 공급자(IdP)를 사용한다면, 코드를 작성하지 않고도 콘솔에 대한 액세스를 설정할 수 있습니다. 이는 Microsoft의 Active Directory Federation Services 또는 오픈 소스 Shibboleth와 같은 공급자와 함께 작동합니다. 자세한 내용은 섹션을 참조하세요SAML 2.0 페더레이션 사용자가 AWS Management Console에 액세스할 수 있게 하기

조직의 사용자가 AWS Management Console에 액세스할 수 있도록 하려는 경우 다음 단계를 수행하여 사용자 지정 자격 증명 브로커를 생성할 수 있습니다.

1. 사용자가 로컬 자격 증명 시스템에 의해 인증되는지 확인합니다.
2. AWS Security Token Service(AWS STS) AssumeRole(권장) 또는 GetFederationToken API 작업을 호출하여 사용자를 위한 임시 보안 자격 증명을 얻을 수 있습니다. 역할을 수임하는 데 사용할 수 있는 여러 방법을 알아보려면 역할 수임 방법 섹션을 참조하세요. 보안 자격 증명을 획득할 때 선택적 세션 태그를 전달하는 방법은 AWS STS에서 세션 태그 전달 섹션을 참조하세요.

- 역할에 대한 임시 보안 자격 증명을 얻기 위해 AssumeRole* API 작업 중 하나를 사용한 경우 이 호출에는 DurationSeconds 파라미터를 포함할 수 있습니다. 이 파라미터는 역할 세션 기간을 900초(15초)에서 해당 역할에 대한 최대 세션 기간 설정까지 지정합니다. AssumeRole* 작업에서 DurationSeconds를 사용하는 경우에는 장기 자격 증명이 있는 IAM 사용자로 호출해야 합니다. 그렇지 않으면 3단계의 연동 엔드포인트 호출에 실패합니다. 역할에 대한 최댓값을 확인 또는 변경하는 방법을 알아보려면 역할의 최대 세션 기간 업데이트 섹션을 참조하세요.
- 보안 자격 증명을 얻기 위해 GetFederationToken API 작업을 사용한 경우 이 호출에는 DurationSeconds 파라미터를 포함할 수 있습니다. 이 파라미터는 역할 세션에 대한 기간을 지정합니다. 이 값은 900초(15분)~129,600초(36시간)일 수 있습니다. IAM 사용자의 장기 AWS 보안 자격 증명을 사용하는 경우에만 이 API를 호출할 수 있습니다. AWS 계정 루트 사용자 자격 증명을 사용하여 호출할 수도 있지만 권장되는 방법은 아닙니다. 루트 사용자로 호출한 경우 기본 세션은 한 시간 동안 지속됩니다. 또는 900초(15분)에서 최대 3,600초(1시간)로 세션을 지정할 수 있습니다.

3. AWS 연동 엔드포인트를 호출하고 임시 보안 자격 증명을 제공하여 로그인 토큰을 요청하세요.
4. 토큰을 포함하는 콘솔에 대한 URL을 생성합니다:
   - URL에 AssumeRole* API 작업 중 하나를 사용하는 경우 SessionDuration HTTP 파라미터를 포함할 수 있습니다. 이 파라미터는 콘솔 세션 시간을 900초(15분)~43200초(12시간)로 지정합니다.
   - URL에 GetFederationToken API 작업을 사용하는 경우 DurationSeconds 파라미터를 포함할 수 있습니다. 이 파라미터는 연동된 콘솔 세션에 대한 기간을 지정합니다. 이 값은 900초(15분)~129,600초(36시간)일 수 있습니다.

   > **Note**
   >
   > - SessionDuration을 사용하여 임시 자격 증명을 얻을 경우에는 GetFederationToken HTTP 파라미터를 사용하지 마세요. 이 파라미터를 사용하면 작업이 실패합니다.
   > - 하나의 역할이 자격 증명을 사용하여 다른 역할을 수임하는 것을 역할 체인이라고 합니다. 역할 함께 묶기를 사용하는 경우 새 자격 증명의 유효 기간은 최대 1시간으로 제한됩니다. 역할을 사용하여 EC2 인스턴스에서 실행되는 애플리케이션에 권한을 부여하는 경우, 이러한 애플리케이션에는 이 제한이 적용되지 않습니다.

5. 사용자에게 URL을 부여하거나 사용자 대신 URL을 호출합니다.

연동 엔드포인트가 제공하는 URL은 생성된 후 15분 동안 유효합니다. 이 시간은 URL과 연결된 임시 보안 자격 증명 세션의 기간(초)과 다릅니다. 이러한 자격 증명은 생성된 시각을 시작으로 생성 시 지정한 기간 동안 유효합니다.

> **Important**
>
> URL은 연결된 임시 보안 자격 증명에서 권한을 허용한 경우 AWS을 통해 AWS Management Console 리소스에 대한 액세스 권한을 부여한다는 것에 유의하세요. 이러한 이유 때문에 URL은 비밀로 취급해야 합니다. 예를 들어 SSL 연결을 통해 302 HTTP 응답 상태 코드를 사용함으로써 안전한 리디렉션을 통해 URL을 반환하는 것이 좋습니다. 302 HTTP 응답 상태 코드에 대한 자세한 정보는 RFC 2616, 단원 10.3.3을 참조하세요.

이 작업을 완료하려면 AWS Identity and Access Management(IAM)를 위한 HTTPS 쿼리 API 및 AWS Security Token Service(AWS STS)를 참조하세요. 아니면 적절한 AWS SDK와 함께 Java, Ruby 또는 C#과 같은 프로그래밍 언어를 사용할 수도 있습니다. 다음 주제에서는 이러한 각 메서드를 설명합니다.

주제

- IAM 쿼리 API 작업을 사용한 예제 코드
- Python을 사용한 예제 코드
- Java를 사용한 예제 코드
- URL을 생성하는 방법을 보여주는 예(Ruby)

IAM 쿼리 API 작업을 사용한 예제 코드

페더레이션 사용자에게 AWS Management Console에 대한 직접 액세스를 부여하는 URL을 생성할 수 있습니다. 이 작업은 IAM 및 AWS STS HTTPS Query API를 사용합니다. 쿼리 요청에 대한 자세한 정보는 쿼리 요청 실행 섹션을 참조하세요.

> **Note**
>
> 다음 절차에는 텍스트 문자열에 대한 예시가 있습니다. 가독성을 증진하기 위해 일부 긴 예시에는 줄 바꿈이 추가되었습니다. 자신만이 쓸 용도로 이러한 문자열을 생성할 때는 줄 바꿈을 모두 빼야 합니다.

AWS Management Console에서 페더레이션 사용자에게 리소스 액세스 권한을 부여하려면

1. 자격 증명 및 인증 시스템에서 사용자를 인증합니다.
2. 사용자에 대한 임시 보안 자격 증명을 얻습니다. 임시 자격 증명은 액세스 키 ID, 비밀 액세스 키 및 세션 토큰으로 구성됩니다. 임시 자격 증명 생성에 대한 자세한 정보는 다음(IAM의 임시 보안 자격 증명)을 참조하세요.

   임시 자격 증명을 얻으려면 AWS STS AssumeRole API(권장) 또는 GetFederationToken API를 호출하면 됩니다. 이러한 API 작업 간의 차이에 대한 자세한 정보는 AWS 보안 블로그에서 AWS 계정에 대한 액세스 권한을 안전하게 위임하기 위한 API 옵션 이해하기를 참조하세요.

   > **Important**
   >
   > GetFederationToken API를 사용하여 임시 보안 자격 증명을 생성한 경우 해당 역할을 수임한 사용자에게 자격 증명을 부여하는 권한을 지정해야 합니다. `AssumeRole*`로 시작하는 API 작업 중 어느 것에 대해서도 IAM 역할을 사용해 권한을 할당할 수 있습니다. 다른 API 작업의 경우 그 메커니즘이 API에 따라 달라집니다. 자세한 내용은 사용자 임시 보안 자격 증명에 대한 권한 제어를 참조하세요. 또한 `AssumeRole*` API 작업을 사용하는 경우 장기 자격 증명을 사용하여 IAM 사용자로 호출해야 합니다. 그렇지 않으면 3단계의 연동 엔드포인트 호출에 실패합니다.

3. 임시 보안 자격 증명을 획득한 후에는 자격 증명을 JSON 세션 문자열로 구성해 로그인 토큰과 교환합니다. 다음 예에서는 자격 증명을 인코딩하는 방법을 보여줍니다. 자리 표시자 텍스트를 이전 단계에서 받는 자격 증명의 적절한 값들로 교체합니다.

   ```
   {"sessionId":"*** temporary access key ID ***",
   "sessionKey":"*** temporary secret access key ***",
   "sessionToken":"*** session token ***"}
   ```

4. URL encode 이전 단계의 세션 문자열. 인코딩하고 있는 정보가 지닌 중요성으로 인해 이러한 인코딩에는 웹 서비스를 사용하지 않는 것이 좋습니다. 대신 개발 도구 키트에 로컬로 설치된 함수 또는 기능을 사용하여 이 정보를 안전하게 인코딩합니다. Python의 `urllib.quote_plus` 함수, Java의 `URLEncoder.encode` 함수 또는 Ruby의 `CGI.escape` 함수를 사용할 수 있습니다. 이 주제 후반의 예제를 참조하세요.
5.

   > **Note**
   >
   > AWS에서는 여기에서 POST 요청을 지원합니다.

AWS 페더레이션 엔드포인트로 요청을 전송하세요.

`https://region-code.signin.aws.amazon.com/federation`

가능한 *region-code* 값 목록은 AWS 로그인 엔드포인트의 리전(Region) 열을 참조하세요. 선택적으로 기본 AWS 로그인 페더레이션 엔드포인트를 사용할 수 있습니다.

`https://signin.aws.amazon.com/federation`

요청에는 `Action` 및 `Session` 파라미터가 포함되어야 하며, `AssumeRole*` API 사용했다면 다음 예제의 `SessionDuration` HTTP 파라미터를 선택적으로 포함시킬 수 있습니다.

```
Action = getSigninToken
SessionDuration = time in seconds
Session = *** the URL encoded JSON string created in steps 3 & 4 ***
```

> **Note**
>
> 이 단계의 다음 지침은 GET 요청을 사용하는 경우에만 작동합니다.

`SessionDuration` HTTP 파라미터는 연동된 콘솔 세션에 대한 기간을 지정합니다. 이 기간은 `DurationSeconds` 파라미터를 사용하여 지정하는 임시 자격 증명의 기간과는 다릅니다. `SessionDuration`의 최댓값은 43200(12시간)까지 지정할 수 있습니다. `SessionDuration` 파라미터가 없을 때는 2단계에서 AWS STS에서 검색한 자격 증명의 지속 시간을 세션의 기본값으로 사용합니다(기본 1시간). `AssumeRole` 파라미터를 사용하여 기간을 지정하는 방법에 대한 자세한 정보는 `DurationSeconds` API 관련 문서를 참조하세요. 연동 엔드포인트의 `getSigninToken` 작업을 이용하면 한 시간보다 긴 콘솔 세션을 만들 수 있습니다.

> **Note**
>
> - `SessionDuration`을 사용하여 임시 자격 증명을 얻을 경우에는 `GetFederationToken` HTTP 파라미터를 사용하지 마세요. 이 파라미터를 사용하면 작업이 실패합니다.
> - 하나의 역할이 자격 증명을 사용하여 다른 역할을 수임하는 것을 역할 체인이라고 합니다. 역할 함께 묶기를 사용하는 경우 새 자격 증명의 유효 기간은 최대 1시간으로 제한됩

니다. 역할을 사용하여 EC2 인스턴스에서 실행되는 애플리케이션에 권한을 부여하는 경우, 이러한 애플리케이션에는 이 제한이 적용되지 않습니다.

기간이 연장된 콘솔 세션을 활성화하면 자격 증명이 노출될 위험이 높아집니다. 이러한 위험을 줄이려면 IAM 콘솔의 역할 요약 페이지에서 세션 취소(Revoke Sessions)를 선택하여 원하는 역할의 활성 콘솔 세션을 즉시 비활성화하면 됩니다. 자세한 내용은 IAM 역할 임시 보안 자격 증명 취소 단원을 참조하십시오.

다음은 요청에 대한 예시입니다. 가독성을 위해 줄바꿈이 되어 있지만 한 줄로 된 문자열로 제출해야 합니다.

```
https://signin.aws.amazon.com/federation
?Action=getSigninToken
&SessionDuration=1800
&Session=%7B%22sessionId%22%3A+%22ASIAJUMHIZPTOKTBMK5A%22%2C+%22sessionKey%22
%3A+%22LSD7LWI%2FL%2FN%2BgYpan5QFz0XUpc8s7HYjRsgcsrsm%22%2C+%22sessionToken%2
2%3A+%22FQoDYXdzEBQaDLbj3VWv2u50NN%2F3yyLSASwYtWhPnGPMNmzZFfZsL0Qd3vtYHw5A5dW
AjOsrkdPkghomIe3mJip5%2F0djDBbo7SmO%2FENDEiCdpsQKodTpleKA8xQq0CwFg6a69xdEBQT8
FipATnLbKoyS4b%2FebhnsTUjZZQWp0wXXqFF7gSm%2FMe2tXe0jzsdP0O12obez9lijPSdF1k2b5
PfGhiuyAR9aD5%2BubM0pY86fKex1qsytjvyTbZ9nXe6DvxVDcnCOhOGETJ7XFkSFdH0v%2FYR25C
UAhJ3nXIkIbG7Ucv9cOEpCf%2Fg23ijRgILIBQ%3D%3D%22%7D
```

연동 엔드포인트의 응답은 SigninToken 값이 있는 JSON 문서입니다. 다음의 예와 유사합니다.

```
{"SigninToken":"*** the SigninToken string ***"}
```

6.

> **Note**
>
> AWS에서는 여기에서 POST 요청을 지원합니다.

마지막으로 페더레이션 사용자가 AWS Management Console에 액세스하는 데 사용할 수 있는 URL을 생성하십시오. 그 URL은 다음 파라미터와 함께 Step 5에서 사용한 것과 동일한 연동 URL 엔드포인트입니다.

```
?Action = login
&Issuer = *** the form-urlencoded URL for your internal sign-in page ***
&Destination = *** the form-urlencoded URL to the desired AWS console page ***
```

```
&SigninToken = *** the value of SigninToken received in the previous step ***
```

> **Note**
>
> 이 단계의 다음 지침은 GET API를 사용하는 경우에만 작동합니다.

다음 예는 최종 URL이 결국 어떤 모양을 갖추게 되는지 보여줍니다. 이 URL은 생성된 시각으로부터 15분 동안 유효합니다. URL에 내장된 콘솔 세션과 임시 보안 자격 증명은 이를 처음 요청할 때 `SessionDuration` HTTP 파라미터에 지정한 지속 기간만큼 유효합니다.

```
https://signin.aws.amazon.com/federation
?Action=login
&Issuer=https%3A%2F%2Fexample.com
&Destination=https%3A%2F%2Fconsole.aws.amazon.com%2F
&SigninToken=VCQgs5qZZt3Q6fn8Tr5EXAMPLEmLnwB7JjUc-SHwnUUWabcRdnWsi4DBn-dvC
CZ85wrD0nmldUcZEXAMPLE-vXYH4Q__mleuF_W2BE5HYexbe9y4Of-kje53SsjNNecATfjIzpW1
WibbnH6YcYRiBoffZBGExbEXAMPLE5aiKX4THWjQKC6gg6alHu6JFrnOJoK3dtP6I9a6hi6yPgm
iOkPZMmNGmhsvVxetKzr8mx3pxhHbMEXAMPLETv1pij0rok3IyCR2YVcIjqwfWv32HU2Xlj471u
3fU6uOfUComeKiqTGX974xzJOZbdmX_t_lLrhEXAMPLEDDIisSnyHGw2xaZZqudm4mo2uTDk9Pv
9l5K0ZCqIgEXAMPLEcA6tgLPykEWGUyH6BdSC6166n4M4JkXIQgac7_7821YqixsNxZ6rsrpzwf
nQoS14O7R0eJCCJ684EXAMPLEZRdBNnuLbUYpz2Iw3vIN0tQgOujwnwydPscM9F7foaEK3jwMkg
Apeb1-6L_OB12MZhuFxx55555EXAMPLEhyETEd4ZulKPdXHkgl6T9ZkIlHz2Uy1RUTUhhUxNtSQ
nWc5xkbBoEcXqpoSIeK7yhje9Vzhd61AEXAMPLElbWeouACEMG6-Vd3dAgFYd6i5FYoyFrZLWvm
0LSG7RyYKeYN5VIzUk3YWQpyjP0RiT5KUrsUi-NEXAMPLExMOMdoODBEgKQsk-iu2ozh6r8bxwC
RNhujg
```

Python을 사용한 예제 코드

다음 예제는 Python을 사용해 페더레이션 사용자에게 AWS Management Console에 대한 직접 액세스 권한을 부여하는 URL을 프로그래밍 방식으로 생성하는 방법을 보여줍니다. 다음과 같은 두 가지 예제가 있습니다.

- GET 요청을 통해 AWS에 페더레이션
- POST 요청을 통해 AWS에 페더레이션

두 예제 모두 AWS SDK for Python (Boto3)과 AssumeRole API를 사용해 임시 보안 자격 증명을 가져옵니다.

GET 요청 사용

```
import urllib, json, sys
import requests # 'pip install requests'
import boto3 # AWS SDK for Python (Boto3) 'pip install boto3'

# Step 1: Authenticate user in your own identity system.

# Step 2: Using the access keys for an IAM user in your AWS ##,
# call "AssumeRole" to get temporary access keys for the federated user

# Note: Calls to AWS STS AssumeRole must be signed using the access key ID
# and secret access key of an IAM user or using existing temporary credentials.
# The credentials can be in Amazon EC2 instance metadata, in environment variables,
# or in a configuration file, and will be discovered automatically by the
# client('sts') function. For more information, see the Python SDK docs:
# http://boto3.readthedocs.io/en/latest/reference/services/sts.html
# http://boto3.readthedocs.io/en/latest/reference/services/
sts.html#STS.Client.assume_role
sts_connection = boto3.client('sts')

assumed_role_object = sts_connection.assume_role(
    RoleArn="arn:aws:iam::account-id:role/ROLE-NAME",
    RoleSessionName="AssumeRoleSession",
)

# Step 3: Format resulting temporary credentials into JSON
url_credentials = {}
url_credentials['sessionId'] =
 assumed_role_object.get('Credentials').get('AccessKeyId')
url_credentials['sessionKey'] =
 assumed_role_object.get('Credentials').get('SecretAccessKey')
url_credentials['sessionToken'] =
 assumed_role_object.get('Credentials').get('SessionToken')
json_string_with_temp_credentials = json.dumps(url_credentials)

# Step 4. Make request to AWS federation endpoint to get sign-in token. Construct the
 parameter string with
# the sign-in action request, a 12-hour session duration, and the JSON document with
 temporary credentials
# as parameters.
request_parameters = "?Action=getSigninToken"
request_parameters += "&SessionDuration=43200"
if sys.version_info[0] < 3:
```

```
    def quote_plus_function(s):
        return urllib.quote_plus(s)
else:
    def quote_plus_function(s):
        return urllib.parse.quote_plus(s)
request_parameters += "&Session=" +
 quote_plus_function(json_string_with_temp_credentials)
request_url = "https://signin.aws.amazon.com/federation" + request_parameters
r = requests.get(request_url)
# Returns a JSON document with a single element named SigninToken.
signin_token = json.loads(r.text)

# Step 5: Create URL where users can use the sign-in token to sign in to
# the console. This URL must be used within 15 minutes after the
# sign-in token was issued.
request_parameters = "?Action=login"
request_parameters += "&Issuer=Example.org"
request_parameters += "&Destination=" + quote_plus_function("https://
console.aws.amazon.com/")
request_parameters += "&SigninToken=" + signin_token["SigninToken"]
request_url = "https://signin.aws.amazon.com/federation" + request_parameters

# Send final URL to stdout
print (request_url)
```

POST 요청 사용

```
import urllib, json, sys
import requests # 'pip install requests'
import boto3 # AWS SDK for Python (Boto3) 'pip install boto3'
import os
from selenium import webdriver # 'pip install selenium', 'brew install chromedriver'

# Step 1: Authenticate user in your own identity system.

# Step 2: Using the access keys for an IAM user in your AAWS ##,
# call "AssumeRole" to get temporary access keys for the federated user

# Note: Calls to AWS STS AssumeRole must be signed using the access key ID
# and secret access key of an IAM user or using existing temporary credentials.
# The credentials can be in Amazon EC2 instance metadata, in environment variables,

# or in a configuration file, and will be discovered automatically by the
```

```
# client('sts') function. For more information, see the Python SDK docs:
# http://boto3.readthedocs.io/en/latest/reference/services/sts.html
# http://boto3.readthedocs.io/en/latest/reference/services/
sts.html#STS.Client.assume_role
if sys.version_info[0] < 3:
    def quote_plus_function(s):
        return urllib.quote_plus(s)
else:
    def quote_plus_function(s):
        return urllib.parse.quote_plus(s)

sts_connection = boto3.client('sts')

assumed_role_object = sts_connection.assume_role(
    RoleArn="arn:aws:iam::account-id:role/ROLE-NAME",
    RoleSessionName="AssumeRoleDemoSession",
)

# Step 3: Format resulting temporary credentials into JSON
url_credentials = {}
url_credentials['sessionId'] =
 assumed_role_object.get('Credentials').get('AccessKeyId')
url_credentials['sessionKey'] =
 assumed_role_object.get('Credentials').get('SecretAccessKey')
url_credentials['sessionToken'] =
 assumed_role_object.get('Credentials').get('SessionToken')
json_string_with_temp_credentials = json.dumps(url_credentials)

# Step 4. Make request to AWS federation endpoint to get sign-in token. Construct the
 parameter string with
# the sign-in action request, a 12-hour session duration, and the JSON document with
 temporary credentials
# as parameters.
request_parameters = {}
request_parameters['Action'] = 'getSigninToken'
request_parameters['SessionDuration'] = '43200'
request_parameters['Session'] = json_string_with_temp_credentials

request_url = "https://signin.aws.amazon.com/federation"
r = requests.post( request_url, data=request_parameters)

# Returns a JSON document with a single element named SigninToken.
signin_token = json.loads(r.text)
```

```
# Step 5: Create a POST request where users can use the sign-in token to sign in to
# the console. The POST request must be made within 15 minutes after the
# sign-in token was issued.
request_parameters = {}
request_parameters['Action'] = 'login'
request_parameters['Issuer']='Example.org'
request_parameters['Destination'] = 'https://console.aws.amazon.com/'
request_parameters['SigninToken'] =signin_token['SigninToken']

jsrequest = '''
var form = document.createElement('form');
form.method = 'POST';
form.action = '{request_url}';
request_parameters = {request_parameters}
for (var param in request_parameters) {{
    if (request_parameters.hasOwnProperty(param)) {{
        const hiddenField = document.createElement('input');
        hiddenField.type = 'hidden';
        hiddenField.name = param;
        hiddenField.value = request_parameters[param];
        form.appendChild(hiddenField);
    }}
}}
document.body.appendChild(form);
form.submit();
'''.format(request_url=request_url, request_parameters=request_parameters)

driver = webdriver.Chrome()
driver.execute_script(jsrequest);
```

Java를 사용한 예제 코드

다음 예는 Java를 사용해 페더레이션 사용자에게 AWS Management Console에 대한 직접 액세스 권한을 부여하는 URL을 프로그래밍 방식으로 생성하는 방법을 보여줍니다. 다음 코드 조각은 AWS SDK for Java를 사용합니다.

```
import java.net.URLEncoder;
import java.net.URL;
import java.net.URLConnection;
import java.io.BufferedReader;
import java.io.InputStreamReader;
// Available at http://www.json.org/java/index.html
import org.json.JSONObject;
```

```
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;


/* Calls to AWS STS API operations must be signed using the access key ID
   and secret access key of an IAM user or using existing temporary
   credentials. The credentials should not be embedded in code. For
   this example, the code looks for the credentials in a
   standard configuration file.
*/
AWSCredentials credentials =
  new PropertiesCredentials(
         AwsConsoleApp.class.getResourceAsStream("AwsCredentials.properties"));

AWSSecurityTokenServiceClient stsClient =
  new AWSSecurityTokenServiceClient(credentials);

GetFederationTokenRequest getFederationTokenRequest =
  new GetFederationTokenRequest();
getFederationTokenRequest.setDurationSeconds(1800);
getFederationTokenRequest.setName("UserName");

// A sample policy for accessing Amazon Simple Notification Service (Amazon SNS) in the
 console.

String policy = "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Action\":\"sns:*\"," +
  "\"Effect\":\"Allow\",\"Resource\":\"*\"}]}";

getFederationTokenRequest.setPolicy(policy);

GetFederationTokenResult federationTokenResult =
  stsClient.getFederationToken(getFederationTokenRequest);

Credentials federatedCredentials = federationTokenResult.getCredentials();

// The issuer parameter specifies your internal sign-in
// page, for example https://mysignin.internal.mycompany.com/.
// The console parameter specifies the URL to the destination console of the
// AWS Management Console. This example goes to Amazon SNS.
// The signin parameter is the URL to send the request to.
```

```
String issuerURL = "https://mysignin.internal.mycompany.com/";
String consoleURL = "https://console.aws.amazon.com/sns";
String signInURL = "https://signin.aws.amazon.com/federation";

// Create the sign-in token using temporary credentials,
// including the access key ID,  secret access key, and session token.
String sessionJson = String.format(
  "{\"%1$s\":\"%2$s\",\"%3$s\":\"%4$s\",\"%5$s\":\"%6$s\"}",
  "sessionId", federatedCredentials.getAccessKeyId(),
  "sessionKey", federatedCredentials.getSecretAccessKey(),
  "sessionToken", federatedCredentials.getSessionToken());

// Construct the sign-in request with the request sign-in token action, a
// 12-hour console session duration, and the JSON document with temporary
// credentials as parameters.

String getSigninTokenURL = signInURL +
                           "?Action=getSigninToken" +
                           "&DurationSeconds=43200" +
                           "&SessionType=json&Session=" +
                           URLEncoder.encode(sessionJson,"UTF-8");

URL url = new URL(getSigninTokenURL);

// Send the request to the AWS federation endpoint to get the sign-in token
URLConnection conn = url.openConnection ();

BufferedReader bufferReader = new BufferedReader(new
  InputStreamReader(conn.getInputStream()));
String returnContent = bufferReader.readLine();

String signinToken = new JSONObject(returnContent).getString("SigninToken");

String signinTokenParameter = "&SigninToken=" + URLEncoder.encode(signinToken,"UTF-8");

// The issuer parameter is optional, but recommended. Use it to direct users
// to your sign-in page when their session expires.

String issuerParameter = "&Issuer=" + URLEncoder.encode(issuerURL, "UTF-8");

// Finally, present the completed URL for the AWS console session to the user

String destinationParameter = "&Destination=" + URLEncoder.encode(consoleURL,"UTF-8");
```

```
String loginURL = signInURL + "?Action=login" +
                     signinTokenParameter + issuerParameter + destinationParameter;
```

URL을 생성하는 방법을 보여주는 예(Ruby)

다음 예는 Ruby를 사용해 페더레이션 사용자에게 AWS Management Console에 대한 직접 액세스 권한을 부여하는 URL을 프로그래밍 방식으로 생성하는 방법을 보여줍니다. 이 코드 조각은 AWS SDK for Ruby를 사용합니다.

```
require 'rubygems'
require 'json'
require 'open-uri'
require 'cgi'
require 'aws-sdk'

# Create a new STS instance
#
# Note: Calls to AWS STS API operations must be signed using an access key ID
# and secret access key. The credentials can be in EC2 instance metadata
# or in environment variables and will be automatically discovered by
# the default credentials provider in the AWS Ruby SDK.
sts = Aws::STS::Client.new()

# The following call creates a temporary session that returns
# temporary security credentials and a session token.
# The policy grants permissions to work
# in the AWS SNS console.

session = sts.get_federation_token({
  duration_seconds: 1800,
  name: "UserName",
  policy: "{\"Version\":\"2012-10-17\",\"Statement\":{\"Effect\":\"Allow\",\"Action\":
\"sns:*\",\"Resource\":\"*\"}}",
})

# The issuer value is the URL where users are directed (such as
# to your internal sign-in page) when their session expires.
#
# The console value specifies the URL to the destination console.
# This example goes to the Amazon SNS console.
#
# The sign-in value is the URL of the AWS STS federation endpoint.
issuer_url = "https://mysignin.internal.mycompany.com/"
```

```
console_url = "https://console.aws.amazon.com/sns"
signin_url = "https://signin.aws.amazon.com/federation"

# Create a block of JSON that contains the temporary credentials
# (including the access key ID, secret access key, and session token).
session_json = {
  :sessionId => session.credentials[:access_key_id],
  :sessionKey => session.credentials[:secret_access_key],
  :sessionToken => session.credentials[:session_token]
}.to_json

# Call the federation endpoint, passing the parameters
# created earlier and the session information as a JSON block.
# The request returns a sign-in token that's valid for 15 minutes.
# Signing in to the console with the token creates a session
# that is valid for 12 hours.
get_signin_token_url = signin_url +
                       "?Action=getSigninToken" +
                       "&SessionType=json&Session=" +
                       CGI.escape(session_json)

returned_content = URI.parse(get_signin_token_url).read

# Extract the sign-in token from the information returned
# by the federation endpoint.
signin_token = JSON.parse(returned_content)['SigninToken']
signin_token_param = "&SigninToken=" + CGI.escape(signin_token)

# Create the URL to give to the user, which includes the
# sign-in token and the URL of the console to open.
# The "issuer" parameter is optional but recommended.
issuer_param = "&Issuer=" + CGI.escape(issuer_url)
destination_param = "&Destination=" + CGI.escape(console_url)
login_url = signin_url + "?Action=login" + signin_token_param +
  issuer_param + destination_param
```

임시 보안 자격 증명에 관한 추가 리소스

다음 시나리오 및 애플리케이션은 임시 보안 자격 증명 사용 방법을 안내합니다.

- AWS STSSourceIdentity를 ID 공급자와 통합하는 방법. 이 게시물에서는 Okta, Ping 또는 OneLogin을 IdP로 사용할 때 AWS STS `SourceIdentity` 속성을 설정하는 방법을 보여줍니다.

- OIDC 페더레이션. 이 섹션에서는 OIDC 페더레이션 및 `AssumeRoleWithWebIdentity` API를 사용할 때 IAM 역할을 구성하는 방법을 설명합니다.
- MFA를 통한 보안 API 액세스이 주제는 역할을 사용해 멀티 팩터 인증(MFA)이 계정에서 민감한 API 작업을 보호하도록 요구하는 방법을 설명합니다.

AWS의 정책 및 권한에 대한 자세한 내용은 다음 주제를 참조하세요.

- AWS리소스에 대한 액세스 관리
- 정책 평가 로직.
- Amazon Simple Storage Service 사용 설명서의 Amazon S3 리소스에 대한 액세스 권한 관리를 참조하세요.
- 해당 신뢰 영역(신뢰할 수 있는 조직 또는 계정) 외의 계정 내 보안 주체가 역할을 수임하는 권한이 있는지 자세히 알고 싶다면, IAM Access Analyzer란 무엇일까요?를 참조하세요.

AWS Identity and Access Management 리소스용 태그

태그는 사용자 또는 AWS 리소스에 할당할 수 있는 사용자 지정 속성 레이블입니다. 각 태그에는 다음 두 가지 부분이 있습니다.

- 태그 키(예: `CostCenter`, `Environment`, `Project` 또는 `Purpose`).
- 태그 값(예: `111122223333`, `Production` 또는 팀 이름)으로 알려진 선택적 필드. 태그 값을 생략하는 것은 빈 문자열을 사용하는 것과 같습니다.

태그 키와 태그 값을 합해서 키 값 페어라고 합니다. IAM 리소스에 사용할 수 있는 태그 수에 대한 제한은 IAM 및 AWS STS 할당량 섹션을 참조하세요.

> **Note**
>
> 태그 키 및 태그 키 값의 대소문자 구분에 대한 자세한 내용은 Case sensitivity을(를) 참조하세요.

태그를 사용하면 AWS 리소스를 식별하고 구성하는 데 도움이 됩니다. 많은 AWS 서비스가 태그 지정을 지원하므로 다른 서비스의 리소스에 동일한 태그를 할당하여 해당 리소스의 관련 여부를 나타낼 수 있습니다. 예를 들어 Amazon S3 버킷에 할당한 것과 동일한 태그를 IAM 역할에 할당할 수 있습니다. 태깅 전략에 대한 자세한 내용은 태깅 AWS 리소스 사용자 안내서를 참조하세요.

태그로 IAM 리소스를 식별, 구성 및 추적하는 것 외에도 IAM 정책의 태그를 사용하여 리소스를 보고 상호 작용할 수 있는 사용자를 제어할 수 있습니다. 태그를 사용하여 액세스를 제어하는 방법에 대한 자세한 내용은 태그를 사용하여 IAM 사용자 및 역할에 대한 액세스 제어 섹션을 참조하세요.

역할을 수임하거나 사용자를 연동할 때 AWS STS에서 태그를 사용하여 사용자 지정 속성을 추가할 수도 있습니다. 자세한 내용은 AWS STS에서 세션 태그 전달 단원을 참조하십시오.

주제

AWS 태그 이름 지정 규칙 선택

IAM 리소스에 태그 연결을 시작할 때 태그 이름 지정 규칙을 신중하게 선택합니다. 모든 AWS 태그에 동일한 규칙을 적용합니다. 정책에서 태그를 사용하여 AWS 리소스에 대한 액세스를 제어하는 경우 특히 중요합니다. AWS에서 태그를 이미 사용하고 있다면 이름 지정 규칙을 검토하고 적절하게 조정합니다.

> **Note**
>
> 계정이 AWS Organizations의 회원인 경우 조직에서의 태그 사용에 대한 자세한 내용은 조직 사용 설명서의 태그 정책을 참조하세요.

태그 이름 지정 모범 사례

다음은 태그에 대한 몇 가지 모범 사례 및 명명 규칙입니다.

태그 이름은 동일하게 사용해야 합니다. 예를 들어 태그 `CostCenter`와 `costcenter`는 서로 다르므로 하나는 재무 분석 및 보고를 위한 비용 할당 태그로 구성되고 다른 하나는 그렇지 않을 수 있습니다. 마찬가지로, `Name` 태그는 많은 리소스에서 AWS 콘솔에 나타나지만 `name` 태그는 그렇지 않습니다. 태그 키 및 태그 키 값의 대소문자 구분에 대한 자세한 내용은 Case sensitivity을(를) 참조하세요.

여러 태그가 AWS에서 미리 정의되거나 다양한 AWS 서비스에서 자동으로 생성됩니다. 다수의 AWS 정의 태그 이름은 단어를 구분하는 하이픈과 함께 모두 소문자를 사용하며, 접두사로 태그의 소스 서비스를 식별합니다. 예:

- `aws:ec2spot:fleet-request-id`는 인스턴스를 시작한 Amazon EC2 스팟 인스턴스 요청을 식별합니다.
- `aws:cloudformation:stack-name`은 리소스를 생성한 AWS CloudFormation 스택을 식별합니다.
- `elasticbeanstalk:environment-name`은 리소스를 생성한 애플리케이션을 식별합니다.

태그 이름은 모두 소문자를 사용하여 지정하는 것이 좋으며, 단어를 구분하는 하이픈과 조직 이름 또는 축약된 이름을 나타내는 접두사를 함께 사용합니다. 예를 들어, 이름이 AnyCompany인 가상의 회사에 대해 다음과 같은 태그를 정의할 수 있습니다.

- 내부 비용 센터 코드를 식별하는 `anycompany:cost-center`
- 환경이 개발, 테스트 또는 프로덕션인지 식별하는 `anycompany:environment-type`
- 리소스가 생성된 애플리케이션을 식별하는 `anycompany:application-id`

접두사를 사용하면 태그가 AWS 또는 사용 중인 타사 도구가 아닌 조직에서 정의한 것으로 명확하게 식별됩니다. 구분 기호에 하이픈과 함께 모두 소문자를 사용하면 태그 이름을 대문자로 표시하는 방법에 대한 혼동을 피할 수 있습니다. 예를 들어, `anycompany:project-id`는 `ANYCOMPANY:ProjectID`,`anycompany:projectID` 또는`Anycompany:ProjectId`보다 간단하게 기억할 수 있습니다.

IAM 및 AWS STS의 태그 지정 규칙

IAM 및 AWS STS에서 태그의 생성과 적용을 관리하는 규칙은 여러 가지가 있습니다.

태그 이름 지정

IAM 리소스, AWS STS assume-role 세션 및 AWS STS 페더레이션 사용자 세션에 대한 태그 이름 지정 규칙을 공식화하는 경우 다음 규칙을 준수합니다.

문자 요구 사항 - 태그 키 및 값은 문자, 숫자, 공백 및 _ . : / = + - @ 기호를 포함할 수 있습니다.

대소문자 구분 - 태그 키의 대소문자 구분은 태깅된 IAM 리소스의 유형에 따라 다릅니다. IAM 사용자 및 역할의 태그 키 값은 대소문자를 구분하지 않지만 대소문자는 유지됩니다. 즉, 별도의 **Department** 및 **department** 태그 키를 가질 수 없음을 의미합니다. **Department=finance** 태그로 사용자 태그를 지정하고 **department=hr** 태그를 추가하면 첫 번째 태그가 바뀝니다. 두 번째 태그는 추가되지 않습니다.

다른 IAM 리소스 유형의 경우 태그 키 값은 대소문자를 구분합니다. 즉, **Costcenter** 태그 키와 **costcenter** 태그 키를 별도로 지정할 수 있습니다. 예를 들어 고객 관리형 정책에 **Costcenter = 1234** 태그를 지정하고 **costcenter = 5678** 태그를 추가하면 정책에는 **Costcenter** 및 **costcenter** 태그 키가 모두 포함됩니다.

모범 사례로서, 대소문자가 일치하지 않는 유사한 태그를 사용하지 않는 것이 좋습니다. 태그를 대문자로 사용하는 전략을 세우고 이러한 전략을 모든 리소스 유형에 대해 일관되게 구현하는 것이 좋습니다. 태깅 모범 사례에 대한 자세한 내용은 AWS 일반 참조의 AWS 리소스 태깅을 참조하세요.

다음 목록은 IAM 리소스에 연결된 태그 키의 대소문자 구분 차이점을 보여줍니다.

태그 키 값이 대소문자를 구분하지 않습니다 .

- IAM 역할
- IAM 사용자

태그 키 값이 대소문자를 구분합니다.

- 고객 관리형 정책
- 인스턴스 프로파일
- OpenID Connect 자격 증명 공급자
- SAML 자격 증명 공급자
- 서버 인증서
- 가상 MFA 디바이스

또한 다음 규칙이 적용됩니다.

- **aws:**로 시작하는 태그 키 또는 값을 생성할 수 없습니다. 이 태그 접두사는 AWS 내부 전용으로 예약되어 있습니다.

- **phoneNumber =** 와 같이 값이 비어 있는 태그를 만들 수 있습니다. 빈 태그 키는 생성할 수 없습니다.
- 단일 태그에 여러 값을 지정할 수 없지만 단일 값으로 사용자 지정 다중 값 구조를 생성할 수 있습니다. 예를 들어 사용자 Zhang이 엔지니어링 팀과 QA 팀에서 근무한다고 가정합니다. **team = Engineering** 태그를 연결하고 **team = QA** 태그를 연결한 경우 태그 값을 **Engineering**에서 **QA**로 변경합니다 . 대신 사용자 지정 구분자를 사용하여 단일 태그에 여러 값을 포함할 수 있습니다. 이 예에서는 Zhang에게 **team = Engineering:QA** 태그를 연결할 수 있습니다.

> **Note**
>
> 이 예제에서 **team** 태그를 사용하여 엔지니어에 대한 액세스를 제어하려면 **Engineering:QA**를 포함하여 **Engineering**을 포함할 수 있는 모든 구성을 허용하는 정책을 만들어야 합니다. 정책에서의 태그 사용에 대한 자세한 내용은 태그를 사용하여 IAM 사용자 및 역할에 대한 액세스 제어 섹션을 참조하세요.

태그 적용 및 편집

태그를 IAM 리소스에 연결할 때는 다음 규칙을 준수합니다.

- 대부분의 IAM 리소스에 태그를 지정할 수 있지만 그룹, 수임한 역할, 액세스 보고서 또는 하드웨어 기반 MFA 디바이스에는 태그를 지정할 수 있습니다.
- Tag Editor를 사용하여 IAM 리소스에 태깅할 수 없습니다. Tag Editor는 IAM 태그를 지원하지 않습니다. Tag Editor를 다른 서비스와 함께 사용하는 방법에 대한 자세한 내용은 AWS Resource Groups 사용 설명서에서 Tag Editor 작업을 참조하세요.
- IAM 리소스를 태깅하려면 특정 권한이 있어야 합니다. 리소스를 태깅하거나 태깅 해제하려면 태그를 나열할 수 있는 권한이 있어야 합니다. 자세한 내용은 이 페이지 끝에 있는 각 IAM 리소스에 대한 주제 목록을 참조하세요.
- AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 섹션을 참조하세요.
- 여러 IAM 리소스에 동일한 태그를 적용할 수 있습니다. 예를 들어, 12명의 멤버가 있는 AWS_Development 부서가 있다고 가정합니다. 태그 키 **department** 및 값 **awsDevelopment**(**department = awsDevelopment**)를 가진 12명의 사용자와 역할이 있을 수 있습니다. 태그 지정을 지원하는 다른 서비스의 리소스에도 동일한 태그를 사용할 수 있습니다.

- IAM 엔터티(사용자 또는 역할)에는 태그 키가 같은 인스턴스가 여러 개 있을 수 없습니다. 예를 들어 태그 키 값 페어 **costCenter = 1234**가 지정된 사용자가 있는 경우 태그 키 값 페어 **costCenter = 5678**을 연결할 수 있습니다. IAM은 **costCenter** 태그의 값을 **5678**로 업데이트합니다.
- IAM 엔터티(사용자 또는 역할)에 연결된 태그를 편집하려면 새 값으로 태그를 연결하여 기존 태그를 덮어씁니다. 예를 들어, 태그 키 값 페어 **department = Engineering**을 가진 사용자가 있다고 가정합니다. 사용자를 QA 부서로 이동해야 하는 경우 **department = QA** 태그 키 값 페어를 사용자에게 연결할 수 있습니다. 결과적으로 **department** 태그 키의 **Engineering** 값이 **QA** 값으로 대체됩니다.

IAM 사용자 태깅

IAM 태그 키 값 페어를 사용하여 IAM 사용자에 사용자 지정 속성을 추가할 수 있습니다. 예를 들어 사용자에게 위치 정보를 추가하려면 태그 키 **location** 및 태그 값 **us_wa_seattle**을 추가할 수 있습니다. 또는 세 개의 개별 위치 태그 키 값 페어 **loc-country = us**, **loc-state = wa** 및 **loc-city = seattle**을 사용할 수도 있습니다. 태그를 사용하여 리소스에 대한 사용자의 액세스를 제어하거나 사용자에게 연결할 수 있는 태그를 제어할 수 있습니다. 태그를 사용하여 액세스를 제어하는 방법에 대한 자세한 내용은 태그를 사용하여 IAM 사용자 및 역할에 대한 액세스 제어 섹션을 참조하세요.

역할을 수임하거나 사용자를 연동할 때 AWS STS에서 태그를 사용하여 사용자 지정 속성을 추가할 수도 있습니다. 자세한 내용은 AWS STS에서 세션 태그 전달 단원을 참조하십시오.

IAM 사용자 태깅에 필요한 권한

IAM 사용자가 다른 사용자에 태깅할 수 있도록 권한을 구성해야 합니다. IAM 정책에서 다음 IAM 태그 작업 중 하나 또는 모두를 지정할 수 있습니다.

- `iam:ListUserTags`
- `iam:TagUser`
- `iam:UntagUser`

IAM 사용자가 특정 사용자의 태그를 추가, 나열 또는 제거하도록 허용하려면

태그를 관리해야 하는 IAM 사용자의 권한 정책에 다음 명령문을 추가합니다. 계정 번호를 사용하고 *<username>*을 태그를 관리해야 하는 사용자의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
```

```
    "Effect": "Allow",
    "Action": [
        "iam:ListUserTags",
        "iam:TagUser",
        "iam:UntagUser"
    ],
    "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

IAM 사용자가 태그를 자체 관리할 수 있게 하려면

사용자가 자신의 태그를 관리할 수 있도록 권한 정책에 다음 명령문을 추가합니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListUserTags",
        "iam:TagUser",
        "iam:UntagUser"
    ],
    "Resource": "arn:aws:iam::user/${aws:username}"
}
```

IAM 사용자가 특정 사용자에 태그를 추가하도록 허용하려면

특정 사용자에 태그를 추가하기만 하고 제거하지는 않으려면 IAM 사용자의 권한 정책에 다음 명령문을 추가합니다.

> **Note**
>
> `iam:TagUser` 작업을 수행하려면 `iam:ListUserTags` 작업도 포함해야 합니다.

이 정책을 사용하려면 *<username>*을 태그를 관리해야 하는 사용자의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
```

```
        "iam:ListUserTags",
        "iam:TagUser"
    ],
    "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

또는 IAMFullAccess 등의 AWS 관리형 정책을 사용하여 IAM에 모든 액세스 권한을 제공할 수 있습니다.

IAM 사용자의 태그 관리(콘솔)

AWS Management Console에서 IAM 사용자의 태그를 관리할 수 있습니다.

사용자의 태그를 관리하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 콘솔의 탐색 창에서 [사용자(Users)]를 선택한 다음 편집할 사용자의 이름을 선택합니다.
3. 태그 탭을 선택하고 다음 작업 중 하나를 완료하세요.
   - 사용자에게 아직 태그가 없는 경우 새 태그 추가를 선택합니다.
   - 기존 태그 세트를 관리하려면 태그 관리(Manage tags)를 선택합니다.
4. 태그를 추가하거나 제거하여 태그 세트를 완성합니다. 변경 사항 저장(Save changes)을 선택합니다.

IAM 사용자의 태그 관리(AWS CLI 또는 AWS API)

IAM 사용자의 태그를 나열, 연결 또는 제거할 수 있습니다. AWS CLI 또는 AWS API를 사용하여 IAM 사용자의 태그를 관리할 수 있습니다.

IAM 사용자에 현재 연결된 태그를 나열하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam list-user-tags
- AWS API: ListUserTags

IAM 사용자에 태그를 연결하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam tag-user
- AWS API: TagUser

IAM 사용자의 태그를 제거하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam untag-user
- AWS API: UntagUser

다른 AWS 서비스의 리소스에 대한 태그 연결 정보는 해당 서비스의 설명서를 참조하세요.

IAM 권한 정책에 태그를 사용하여 보다 세분화된 권한을 설정하는 방법에 대한 자세한 내용은 IAM 정책 요소: 변수 및 태그 섹션을 참조하세요.

IAM 역할 태깅

IAM 태그 키 값 페어를 사용하여 IAM 역할에 사용자 지정 속성을 추가할 수 있습니다. 예를 들어 역할에 위치 정보를 추가하려는 경우 태그 키 **location**과 태그 값 **us_wa_seattle**을 추가할 수 있습니다. 또는 세 개의 개별 위치 태그 키 값 페어 **loc-country = us**, **loc-state = wa** 및 **loc-city = seattle**을 사용할 수도 있습니다. 태그를 사용하여 리소스에 대한 역할의 액세스를 제어하거나 역할에 연결할 수 있는 태그를 제어할 수 있습니다. 태그를 사용하여 액세스를 제어하는 방법에 대한 자세한 내용은 태그를 사용하여 IAM 사용자 및 역할에 대한 액세스 제어 섹션을 참조하세요.

역할을 수임하거나 사용자를 연동할 때 AWS STS에서 태그를 사용하여 사용자 지정 속성을 추가할 수도 있습니다. 자세한 내용은 AWS STS에서 세션 태그 전달 단원을 참조하십시오.

IAM 역할 태깅에 필요한 권한

IAM 역할이 다른 엔터티(사용자 또는 역할)에 태깅할 수 있도록 권한을 구성해야 합니다. IAM 정책에서 다음 IAM 태그 작업 중 하나 또는 모두를 지정할 수 있습니다.

- `iam:ListRoleTags`
- `iam:TagRole`
- `iam:UntagRole`
- `iam:ListUserTags`
- `iam:TagUser`
- `iam:UntagUser`

IAM 역할이 특정 사용자의 태그를 추가, 나열 또는 제거하도록 허용하려면

태그를 관리해야 하는 IAM 역할의 권한 정책에 다음 명령문을 추가합니다. 계정 번호를 사용하고 *<username>*을 태그를 관리해야 하는 사용자의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용

하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListUserTags",
        "iam:TagUser",
        "iam:UntagUser"
    ],
    "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

IAM 역할이 특정 사용자에 태그를 추가하도록 허용하려면

특정 사용자에 태그를 추가하기만 하고 제거하지는 않으려면 IAM 역할의 권한 정책에 다음 명령문을 추가합니다.

이 정책을 사용하려면 *<username>*을 태그를 관리해야 하는 사용자의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListUserTags",
        "iam:TagUser"
    ],
    "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

IAM 역할이 특정 역할의 태그를 추가, 나열 또는 제거하도록 허용하려면

태그를 관리해야 하는 IAM 역할의 권한 정책에 다음 명령문을 추가합니다. *<rolename>*을 태그를 관리해야 하는 역할의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListRoleTags",
        "iam:TagRole",
```

```
        "iam:UntagRole"
    ],
    "Resource": "arn:aws:iam::<account-number>:role/<rolename>"
}
```

또는 IAMFullAccess 등의 AWS 관리형 정책을 사용하여 IAM에 모든 액세스 권한을 제공할 수 있습니다.

IAM 역할의 태그 관리(콘솔)

AWS Management Console에서 IAM 역할의 태그를 관리할 수 있습니다.

역할의 태그를 관리하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 콘솔의 탐색 창에서 [역할(Roles)]을 선택한 다음 편집할 역할의 이름을 선택합니다.
3. 태그 탭을 선택하고 다음 작업 중 하나를 완료하세요.
   - 역할에 아직 태그가 없는 경우 새 태그 추가(Add new tag)를 선택합니다.
   - 기존 태그 세트를 관리하려면 태그 관리(Manage tags)를 선택합니다.
4. 태그를 추가하거나 제거하여 태그 세트를 완성합니다. 그런 다음 Save changes(변경 사항 저장)을 선택합니다.

IAM 역할의 태그 관리(AWS CLI 또는 AWS API)

IAM 역할의 태그를 나열, 연결 또는 제거할 수 있습니다. AWS CLI 또는 AWS API를 사용하여 IAM 역할의 태그를 관리할 수 있습니다.

IAM 역할에 현재 연결된 태그를 나열하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam list-role-tags
- AWS API: ListRoleTags

IAM 역할에 태그를 연결하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam tag-role
- AWS API: TagRole

IAM 역할에서 태그를 제거하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam untag-role
- AWS API: UntagRole

다른 AWS 서비스의 리소스에 대한 태그 연결 정보는 해당 서비스의 설명서를 참조하세요.

IAM 권한 정책에 태그를 사용하여 보다 세분화된 권한을 설정하는 방법에 대한 자세한 내용은 IAM 정책 요소: 변수 및 태그 섹션을 참조하세요.

고객 관리형 정책 태깅

IAM 태그 키 값 페어를 사용하여 고객 관리형 정책에 사용자 지정 속성을 추가할 수 있습니다. 예를 들어 부서 정보를 정책에 태깅하려는 경우 태그 키 **Department**와 태그 값 **eng**를 추가할 수 있습니다. 또는 정책에 태깅하여 **Environment = lab**과 같은 특정 환경을 위한 정책을 지정할 수 있습니다. 태그를 사용하여 리소스에 대한 액세스를 제어하거나 리소스에 연결할 수 있는 태그를 제어할 수 있습니다. 태그를 사용하여 액세스를 제어하는 방법에 대한 자세한 내용은 태그를 사용하여 IAM 사용자 및 역할에 대한 액세스 제어 섹션을 참조하세요.

역할을 수임하거나 사용자를 연동할 때 AWS STS에서 태그를 사용하여 사용자 지정 속성을 추가할 수도 있습니다. 자세한 내용은 AWS STS에서 세션 태그 전달 단원을 참조하십시오.

고객 관리형 정책을 태깅하는 데 필요한 권한

IAM 엔터티(사용자 또는 역할)가 고객 관리형 정책을 태깅할 수 있도록 권한을 구성해야 합니다. IAM 정책에서 다음 IAM 태그 작업 중 하나 또는 모두를 지정할 수 있습니다.

- `iam:ListPolicyTags`
- `iam:TagPolicy`
- `iam:UntagPolicy`

IAM 엔터티(사용자 또는 역할)가 고객 관리형 정책의 태그를 추가, 나열 또는 제거하도록 허용하려면

태그를 관리해야 하는 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다. 계정 번호를 사용하고 *<policyname>*을 태그를 관리해야 하는 정책의 이름으로 바꿉니다. 이 예시 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListPolicyTags",
        "iam:TagPolicy",
        "iam:UntagPolicy"
    ],
    "Resource": "arn:aws:iam::<account-number>:policy/<policyname>"
}
```

IAM 엔터티(사용자 또는 역할)가 특정 고객 관리형 정책에 태그를 추가하도록 허용하려면

특정 정책의 태그를 추가하기만 하고 제거하지는 않으려면 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다.

> **Note**
>
> `iam:TagPolicy` 작업을 수행하려면 `iam:ListPolicyTags` 작업도 포함해야 합니다.

이 정책을 사용하려면 *<policyname>*을 태그를 관리해야 하는 정책의 이름으로 바꿉니다. 이 예시 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called "JSON 편집기를 사용하여 정책 생성" 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListPolicyTags",
        "iam:TagPolicy"
    ],
    "Resource": "arn:aws:iam::<account-number>:policy/<policyname>"
}
```

또는 IAMFullAccess 등의 AWS 관리형 정책을 사용하여 IAM에 모든 액세스 권한을 제공할 수 있습니다.

IAM 고객 관리형 정책의 태그 관리(콘솔)

AWS Management Console에서 IAM 고객 관리형 정책의 태그를 관리할 수 있습니다.

고객 관리형 정책의 태그를 관리하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 콘솔의 탐색 창에서 [정책(Policies)]을 선택한 다음 편집할 고객 관리형 정책의 이름을 선택합니다.
3. 태그 탭을 선택한 후 태그 관리를 선택합니다.
4. 태그를 추가하거나 제거하여 태그 세트를 완성합니다. 변경 사항 저장(Save changes)을 선택합니다.

IAM 고객 관리형 정책의 태그 관리(AWS CLI 또는 AWS API)

IAM 고객 관리형 정책의 태그를 나열, 연결 또는 제거할 수 있습니다. AWS CLI 또는 AWS API를 사용하여 IAM 고객 관리형 정책의 태그를 관리할 수 있습니다.

IAM 고객 관리형 정책에 현재 연결된 태그를 나열하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam list-policy-tags
- AWS API: ListPolicyTags

IAM 고객 관리형 정책에 태그를 연결하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam tag-policy
- AWS API: TagPolicy

IAM 고객 관리형 정책에서 태그를 제거하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam untag-policy
- AWS API: UntagPolicy

다른 AWS 서비스의 리소스에 대한 태그 연결 정보는 해당 서비스의 설명서를 참조하세요.

IAM 권한 정책에 태그를 사용하여 보다 세분화된 권한을 설정하는 방법에 대한 자세한 내용은 IAM 정책 요소: 변수 및 태그 섹션을 참조하세요.

OpenID Connect(OIDC) ID 제공업체 태깅

IAM 태그 키 값을 사용하여 IAM OpenID Connect(OIDC) 자격 증명 공급자에 사용자 지정 속성을 추가할 수 있습니다. 예를 들어 OIDC 자격 증명 공급자를 식별하기 위해 태그 키 **google**과 태그 값 **oidc**를 추가할 수 있습니다. 태그를 사용하여 리소스에 대한 액세스를 제어하거나 객체에 연결할 수 있는 태그를 제어할 수 있습니다. 태그를 사용하여 액세스를 제어하는 방법에 대한 자세한 내용은 태그를 사용하여 IAM 사용자 및 역할에 대한 액세스 제어 섹션을 참조하세요.

IAM OIDC 자격 증명 공급자를 태깅하는 데 필요한 권한

IAM 엔터티(사용자 또는 역할)가 IAM OIDC 자격 증명 공급자를 태깅할 수 있도록 권한을 구성해야 합니다. IAM 정책에서 다음 IAM 태그 작업 중 하나 또는 모두를 지정할 수 있습니다.

- `iam:ListOpenIDConnectProviderTags`
- `iam:TagOpenIDConnectProvider`
- `iam:UntagOpenIDConnectProvider`

IAM 엔터티가 IAM OIDC ID 제공업체의 태그를 추가, 나열 또는 제거하도록 허용하려면

태그를 관리해야 하는 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다. 계정 번호를 사용하고 *<OIDCProviderName>*를 태그를 관리해야 하는 OIDC 공급자의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListOpenIDConnectProviderTags",
        "iam:TagOpenIDConnectProvider",
        "iam:UntagOpenIDConnectProvider"
    ],
    "Resource": "arn:aws:iam::<account-number>:oidc-provider/<OIDCProviderName>"
}
```

IAM 엔터티(사용자 또는 역할)가 특정 IAM OIDC 자격 증명 공급자에 태그를 추가하도록 허용하려면

특정 자격 증명 공급자의 태그를 추가하기만 하고 제거하지는 않으려면 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다.

> **Note**
>
> `iam:TagOpenIDConnectProvider` 작업을 수행하려면 `iam:ListOpenIDConnectProviderTags` 작업도 포함해야 합니다.

이 정책을 사용하려면 *<OIDCProviderName>*을 태그를 관리해야 하는 OIDC 공급자의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListOpenIDConnectProviderTags",
        "iam:TagOpenIDConnectProvider"
    ],
    "Resource": "arn:aws:iam::<account-number>:oidc-provider/<OIDCProviderName>"
}
```

또는 IAMFullAccess 등의 AWS 관리형 정책을 사용하여 IAM에 모든 액세스 권한을 제공할 수 있습니다.

IAM OIDC 자격 증명 공급자의 태그 관리(콘솔)

AWS Management Console에서 IAM OIDC 자격 증명 공급자의 태그를 관리할 수 있습니다.

OIDC 자격 증명 공급자의 태그를 관리하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 콘솔의 탐색 창에서 [자격 증명 공급자(Identity providers)]를 선택한 다음 편집할 자격 증명 공급자의 이름을 선택합니다.
3. 태그 탭을 선택한 다음 태그 섹션에서 태그 관리를 선택하고 다음 작업 중 하나를 완료합니다.
   - OIDC 자격 증명 공급자에 아직 태그가 없거나 새 태그를 추가하려면 [태그 추가(Add tag)]를 선택합니다.
   - 기존 태그 키 및 값을 편집합니다.
   - 태그를 제거하려면 [태그 제거(Remove tag)]를 선택합니다.
4. 변경 사항 저장(Save changes)을 선택합니다.

IAM OIDC 자격 증명 공급자의 태그 관리(AWS CLI 또는 AWS API)

IAM OIDC 자격 증명 공급자의 태그를 나열, 연결 또는 제거할 수 있습니다. AWS CLI 또는 AWS API를 사용하여 IAM OIDC 자격 증명 공급자의 태그를 관리할 수 있습니다.

IAM OIDC 자격 증명 공급자에 현재 연결된 태그를 나열하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam list-open-id-connect-provider-tags
- AWS API: ListOpenIDConnectProviderTags

IAM OIDC 자격 증명 공급자에 태그를 연결하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam tag-open-id-connect-provider
- AWS API: TagOpenIDConnectProvider

IAM OIDC 자격 증명 공급자에서 태그를 제거하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam untag-open-id-connect-provider
- AWS API: UntagOpenIDConnectProvider

다른 AWS 서비스의 리소스에 대한 태그 연결 정보는 해당 서비스의 설명서를 참조하세요.

IAM 권한 정책에 태그를 사용하여 보다 세분화된 권한을 설정하는 방법에 대한 자세한 내용은 IAM 정책 요소: 변수 및 태그 섹션을 참조하세요.

IAM SAML ID 제공업체 태깅

IAM 태그 키 값 페어를 사용하여 SAML 자격 증명 공급자에 사용자 지정 속성을 추가할 수 있습니다. 예를 들어 공급자를 식별하기 위해 태그 키 **okta**와 태그 값 **saml**을 추가할 수 있습니다. 태그를 사용하여 리소스에 대한 액세스를 제어하거나 객체에 연결할 수 있는 태그를 제어할 수 있습니다. 태그를 사용하여 액세스를 제어하는 방법에 대한 자세한 내용은 태그를 사용하여 IAM 사용자 및 역할에 대한 액세스 제어 섹션을 참조하세요.

SAML 자격 증명 공급자를 태깅하는 데 필요한 권한

IAM 엔터티(사용자 또는 역할)가 SAML 2.0 기반 자격 증명 공급자(IdP)를 태깅할 수 있도록 권한을 구성해야 합니다. IAM 정책에서 다음 IAM 태그 작업 중 하나 또는 모두를 지정할 수 있습니다.

- `iam:ListSAMLProviderTags`
- `iam:TagSAMLProvider`
- `iam:UntagSAMLProvider`

IAM 엔터티(사용자 또는 역할)가 SAML 자격 증명 공급자의 태그를 추가, 나열 또는 제거하도록 허용하려면

태그를 관리해야 하는 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다. 계정 번호를 사용하고 *<SAMLProviderName>*를 태그를 관리해야 하는 SAML 공급자의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListSAMLProviderTags",
        "iam:TagSAMLProvider",
        "iam:UntagSAMLProvider"
    ],
    "Resource": "arn:aws:iam::<account-number>:saml-provider/<SAMLProviderName>"
}
```

IAM 엔터티(사용자 또는 역할)가 특정 SAML 자격 증명 공급자에 태그를 추가하도록 허용하려면

특정 SAML 공급자의 태그를 추가하기만 하고 제거하지는 않으려면 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다.

> **Note**
>
> `iam:TagSAMLProvider` 작업을 수행하려면 `iam:ListSAMLProviderTags` 작업도 포함해야 합니다.

이 정책을 사용하려면 *<SAMLProviderName>*을 태그를 관리해야 하는 SAML 공급자의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Effect": "Allow",
```

```
    "Action": [
        "iam:ListSAMLProviderTags",
        "iam:TagSAMLProvider"
    ],
    "Resource": "arn:aws:iam::<account-number>:saml-provider/<SAMLProviderName>"
}
```

또는 IAMFullAccess 등의 AWS 관리형 정책을 사용하여 IAM에 모든 액세스 권한을 제공할 수 있습니다.

IAM SAML 자격 증명 공급자의 태그 관리(콘솔)

AWS Management Console에서 IAM SAML 자격 증명 공급자의 태그를 관리할 수 있습니다.

SAML 자격 증명 공급자의 태그를 관리하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 콘솔의 탐색 창에서 [자격 증명 공급자(Identity providers)]를 선택한 다음 편집할 SAML 자격 증명 공급자의 이름을 선택합니다.
3. 태그 탭을 선택한 다음 태그 섹션에서 태그 관리를 선택하고 다음 작업 중 하나를 완료합니다.
   - SAML 자격 증명 공급자에 아직 태그가 없거나 새 태그를 추가하려면 [태그 추가(Add tag)]를 선택합니다.
   - 기존 태그 키 및 값을 편집합니다.
   - 태그를 제거하려면 [태그 제거(Remove tag)]를 선택합니다.
4. 태그를 추가하거나 제거하여 태그 세트를 완성합니다. 변경 사항 저장(Save changes)을 선택합니다.

IAM SAML 자격 증명 공급자의 태그 관리(AWS CLI 또는 AWS API)

IAM SAML 자격 증명 공급자의 태그를 나열, 연결 또는 제거할 수 있습니다. AWS CLI 또는 AWS API를 사용하여 IAM SAML 자격 증명 공급자의 태그를 관리할 수 있습니다.

SAML 자격 증명 공급자에 현재 연결된 태그를 나열하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam list-saml-provider-tags
- AWS API: ListSAMLProviderTags

SAML 자격 증명 공급자에 태그를 연결하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam tag-saml-provider
- AWS API: TagSAMLProvider

SAML 자격 증명 공급자에서 태그를 제거하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam untag-saml-provider
- AWS API: UntagSAMLProvider

다른 AWS 서비스의 리소스에 대한 태그 연결 정보는 해당 서비스의 설명서를 참조하세요.

IAM 권한 정책에 태그를 사용하여 보다 세분화된 권한을 설정하는 방법에 대한 자세한 내용은 IAM 정책 요소: 변수 및 태그 섹션을 참조하세요.

Amazon EC2 역할에 대한 인스턴스 프로파일 태깅

Amazon EC2 인스턴스를 시작할 때 인스턴스와 연결할 IAM 역할을 지정합니다. 인스턴스 프로파일이란 IAM 역할을 위한 컨테이너로서 인스턴스 시작 시 Amazon EC2 인스턴스에 역할 정보를 전달하는 데 사용됩니다. AWS CLI 또는 AWS API를 사용할 경우 인스턴스 프로파일을 태깅할 수 있습니다.

IAM 태그 키 값 페어를 사용하여 인스턴스 프로파일에 사용자 지정 속성을 추가할 수 있습니다. 예를 들어 인스턴스 프로파일에 부서 정보를 추가하려면 태그 키 **access-team**과 태그 값 **eng**를 추가할 수 있습니다. 이렇게 하면 일치하는 태그가 있는 보안 주체가 동일한 태그를 가진 인스턴스 프로파일에 액세스할 수 있습니다. **access-team = eng** 및 **project = peg**와 같이 여러 태그 키 값 페어를 사용하여 팀과 프로젝트를 지정할 수 있습니다. 태그를 사용하여 리소스에 대한 사용자의 액세스를 제어하거나 사용자에게 연결할 수 있는 태그를 제어할 수 있습니다. 태그를 사용하여 액세스를 제어하는 방법에 대한 자세한 내용은 태그를 사용하여 IAM 사용자 및 역할에 대한 액세스 제어 섹션을 참조하세요.

역할을 수임하거나 사용자를 연동할 때 AWS STS에서 태그를 사용하여 사용자 지정 속성을 추가할 수도 있습니다. 자세한 내용은 AWS STS에서 세션 태그 전달 단원을 참조하십시오.

인스턴스 프로파일을 태깅하는 데 필요한 권한

IAM 엔터티(사용자 또는 역할)가 인스턴스 프로파일을 태깅할 수 있도록 권한을 구성해야 합니다. IAM 정책에서 다음 IAM 태그 작업 중 하나 또는 모두를 지정할 수 있습니다.

- `iam:ListInstanceProfileTags`

- `iam:TagInstanceProfile`
- `iam:UntagInstanceProfile`

IAM 엔터티(사용자 또는 역할)가 인스턴스 프로파일의 태그를 추가, 나열 또는 제거하도록 허용하려면

태그를 관리해야 하는 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다. 계정 번호를 사용하고 *<InstanceProfileName>*을 태그를 관리해야 하는 인스턴스 프로파일의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called "JSON 편집기를 사용하여 정책 생성" 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListInstanceProfileTags",
        "iam:TagInstanceProfile",
        "iam:UntagInstanceProfile"
    ],
    "Resource": "arn:aws:iam::<account-number>:instance-profile/<InstanceProfileName>"
}
```

IAM 엔터티(사용자 또는 역할)가 특정 인스턴스 프로파일에 태그를 추가하도록 허용하려면

특정 인스턴스 프로파일의 태그를 추가하기만 하고 제거하지는 않으려면 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다.

> **Note**
>
> `iam:TagInstanceProfile` 작업을 수행하려면 `iam:ListInstanceProfileTags` 작업도 포함해야 합니다.

이 정책을 사용하려면 *<InstanceProfileName>*을 태그를 관리해야 하는 인스턴스 프로파일의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called "JSON 편집기를 사용하여 정책 생성" 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListInstanceProfileTags",
```

```
        "iam:TagInstanceProfile"
    ],
    "Resource": "arn:aws:iam::<account-number>:instance-profile/<InstanceProfileName>"
}
```

또는 IAMFullAccess 등의 AWS 관리형 정책을 사용하여 IAM에 모든 액세스 권한을 제공할 수 있습니다.

인스턴스 프로파일의 태그 관리(AWS CLI 또는 AWS API)

인스턴스 프로파일의 태그를 나열, 연결 또는 제거할 수 있습니다. AWS CLI 또는 AWS API를 사용하여 인스턴스 프로파일의 태그를 관리할 수 있습니다.

인스턴스 프로파일에 현재 연결된 태그를 나열하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam list-instance-profile-tags
- AWS API: ListInstanceProfileTags

인스턴스 프로파일에 태그를 연결하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam tag-instance-profile
- AWS API: TagInstanceProfile

인스턴스 프로파일에서 태그를 제거하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam untag-instance-profile
- AWS API: UntagInstanceProfile

다른 AWS 서비스의 리소스에 대한 태그 연결 정보는 해당 서비스의 설명서를 참조하세요.

IAM 권한 정책에 태그를 사용하여 보다 세분화된 권한을 설정하는 방법에 대한 자세한 내용은 IAM 정책 요소: 변수 및 태그 섹션을 참조하세요.

서버 인증서 태깅

IAM을 사용하여 SSL/TLS 인증서를 관리하는 경우 AWS CLI 또는 AWS API를 사용하여 IAM의 서버 인증서를 태깅할 수 있습니다. AWS Certificate Manager(ACM)에서 지원하는 리전의 인증서인 경우, IAM 대신 ACM을 사용하여 서버 인증서를 프로비저닝, 관리 및 배포하는 것이 좋습니다. 지원되지 않

는 리전에서는 IAM을 인증서 관리자로 사용해야 합니다. ACM이 지원하는 리전을 알아보려면 AWS 일반 참조의 AWS Certificate Manager 엔드포인트 및 할당량을 참조하세요.

IAM 태그 키 값 페어를 사용하여 서버 인증서에 사용자 지정 속성을 추가할 수 있습니다. 예를 들어 서버 인증서의 소유자 또는 관리자에 대한 정보를 추가하기 위해 태그 키 **owner**와 태그 값 **net-eng**를 추가합니다. 또는 태그 키 **CostCenter**와 태그 값 **1234**를 추가하여 비용 센터를 지정할 수도 있습니다. 태그를 사용하여 리소스에 대한 액세스를 제어하거나 리소스에 연결할 수 있는 태그를 제어할 수 있습니다. 태그를 사용하여 액세스를 제어하는 방법에 대한 자세한 내용은 태그를 사용하여 IAM 사용자 및 역할에 대한 액세스 제어 섹션을 참조하세요.

역할을 수임하거나 사용자를 연동할 때 AWS STS에서 태그를 사용하여 사용자 지정 속성을 추가할 수도 있습니다. 자세한 내용은 AWS STS에서 세션 태그 전달 단원을 참조하십시오.

서버 인증서를 태깅하는 데 필요한 권한

IAM 엔터티(사용자 또는 역할)가 서버 인증서를 태깅할 수 있도록 권한을 구성해야 합니다. IAM 정책에서 다음 IAM 태그 작업 중 하나 또는 모두를 지정할 수 있습니다.

- `iam:ListServerCertificateTags`
- `iam:TagServerCertificate`
- `iam:UntagServerCertificate`

IAM 엔터티(사용자 또는 역할)가 서버 인증서의 태그를 추가, 나열 또는 제거하도록 허용하려면

태그를 관리해야 하는 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다. 계정 번호를 사용하고 *<CertificateName>*을 태그를 관리해야 하는 서버 인증서의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListServerCertificateTags",
        "iam:TagServerCertificate",
        "iam:UntagServerCertificate"
    ],
    "Resource": "arn:aws:iam::<account-number>:server-certificate/<CertificateName>"
}
```

IAM 엔터티(사용자 또는 역할)가 특정 서버 인증서에 태그를 추가하도록 허용하려면

특정 서버 인증서의 태그를 추가하기만 하고 제거하지는 않으려면 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다.

> **Note**
>
> `iam:TagServerCertificate` 작업을 수행하려면 `iam:ListServerCertificateTags` 작업도 포함해야 합니다.

이 정책을 사용하려면 *<CertificateName>*을 태그를 관리해야 하는 서버 인증서의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListServerCertificateTags",
        "iam:TagServerCertificate"
    ],
    "Resource": "arn:aws:iam::<account-number>:server-certificate/<CertificateName>"
}
```

또는 IAMFullAccess 등의 AWS 관리형 정책을 사용하여 IAM에 모든 액세스 권한을 제공할 수 있습니다.

서버 인증서의 태그 관리(AWS CLI 또는 AWS API)

서버 인증서의 태그를 나열, 연결 또는 제거할 수 있습니다. AWS CLI 또는 AWS API를 사용하여 서버 인증서의 태그를 관리할 수 있습니다.

서버 인증서에 현재 연결된 태그를 나열하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam list-server-certificate-tags
- AWS API: ListServerCertificateTags

서버 인증서에 태그를 연결하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam tag-server-certificate
- AWS API: TagServerCertificate

서버 인증서에서 태그를 제거하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam untag-server-certificate
- AWS API: UntagServerCertificate

다른 AWS 서비스의 리소스에 대한 태그 연결 정보는 해당 서비스의 설명서를 참조하세요.

IAM 권한 정책에 태그를 사용하여 보다 세분화된 권한을 설정하는 방법에 대한 자세한 내용은 IAM 정책 요소: 변수 및 태그 섹션을 참조하세요.

가상 MFA 디바이스 태깅

IAM 태그 키 값 페어를 사용하여 가상 MFA 디바이스에 사용자 지정 속성을 추가할 수 있습니다. 예를 들어 사용자의 가상 MFA 디바이스에 대한 비용 센터 정보를 추가하려는 경우 태그 키 **CostCenter**와 태그 값 **1234**를 추가할 수 있습니다. 태그를 사용하여 리소스에 대한 액세스를 제어하거나 객체에 연결할 수 있는 태그를 제어할 수 있습니다. 태그를 사용하여 액세스를 제어하는 방법에 대한 자세한 내용은 태그를 사용하여 IAM 사용자 및 역할에 대한 액세스 제어 섹션을 참조하세요.

역할을 수임하거나 사용자를 연동할 때 AWS STS에서 태그를 사용하여 사용자 지정 속성을 추가할 수도 있습니다. 자세한 내용은 AWS STS에서 세션 태그 전달 단원을 참조하십시오.

가상 MFA 디바이스를 태깅하는 데 필요한 권한

IAM 엔터티(사용자 또는 역할)가 가상 MFA 디바이스를 태깅할 수 있도록 권한을 구성해야 합니다. IAM 정책에서 다음 IAM 태그 작업 중 하나 또는 모두를 지정할 수 있습니다.

- `iam:ListMFADeviceTags`
- `iam:TagMFADevice`
- `iam:UntagMFADevice`

IAM 엔터티(사용자 또는 역할)가 가상 MFA 디바이스의 태그를 추가, 나열 또는 제거하도록 허용하려면

태그를 관리해야 하는 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다. 계정 번호를 사용하고 *<MFATokenID>*를 태그를 관리해야 하는 가상 MFA 디바이스의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
```

```
    "Effect": "Allow",
    "Action": [
        "iam:ListMFADeviceTags",
        "iam:TagMFADevice",
        "iam:UntagMFADevice"
    ],
    "Resource": "arn:aws:iam::<account-number>:mfa/<MFATokenID>"
}
```

IAM 엔터티(사용자 또는 역할)가 특정 가상 MFA 디바이스에 태그를 추가하도록 허용하려면

특정 MFA 디바이스의 태그를 추가하기만 하고 제거하지는 않으려면 IAM 엔터티의 권한 정책에 다음 명령문을 추가합니다.

> **Note**
>
> `iam:TagMFADevice` 작업을 수행하려면 `iam:ListMFADeviceTags` 작업도 포함해야 합니다.

이 정책을 사용하려면 *<MFATokenID>*를 태그를 관리해야 하는 가상 MFA 디바이스의 이름으로 바꿉니다. 이 예제 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListMFADeviceTags",
        "iam:TagMFADevice"
    ],
    "Resource": "arn:aws:iam::<account-number>:mfa/<MFATokenID>"
}
```

또는 IAMFullAccess 등의 AWS 관리형 정책을 사용하여 IAM에 모든 액세스 권한을 제공할 수 있습니다.

가상 MFA 디바이스의 태그 관리(AWS CLI 또는 AWS API)

가상 MFA 디바이스의 태그를 나열, 연결 또는 제거할 수 있습니다. AWS CLI 또는 AWS API를 사용하여 가상 MFA 디바이스의 태그를 관리할 수 있습니다.

가상 MFA 디바이스에 현재 연결된 태그를 나열하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam list-mfa-device-tags
- AWS API: ListMFADeviceTags

가상 MFA 디바이스에 태그를 연결하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam tag-mfa-device
- AWS API: TagMFADevice

가상 MFA 디바이스에서 태그를 제거하려면(AWS CLI 또는 AWS API)

- AWS CLI: aws iam untag-mfa-device
- AWS API: UntagMFADevice

다른 AWS 서비스의 리소스에 대한 태그 연결 정보는 해당 서비스의 설명서를 참조하세요.

IAM 권한 정책에 태그를 사용하여 보다 세분화된 권한을 설정하는 방법에 대한 자세한 내용은 IAM 정책 요소: 변수 및 태그 섹션을 참조하세요.

AWS STS에서 세션 태그 전달

세션 태그는 AWS STS에서 IAM 역할을 수임하거나 사용자를 페더레이션할 때 전달하는 키 값 페어 속성입니다. AWS STS 또는 자격 증명 공급자(IdP)를 통해 AWS CLI 또는 AWS API를 요청하면 됩니다. AWS STS를 사용하여 임시 보안 자격 증명을 요청하면 세션이 생성됩니다. 세션은 만료되고 액세스 키 페어 및 세션 토큰과 같은 자격 증명을 갖습니다. 세션 자격 증명을 사용하여 후속 요청을 하면 요청 컨텍스트에 `aws:PrincipalTag` 컨텍스트 키가 포함됩니다. 정책의 `Condition` 요소에서 `aws:PrincipalTag` 키를 사용하여 해당 태그를 기반으로 액세스를 허용하거나 거부할 수 있습니다.

임시 자격 증명을 사용하여 요청하면 보안 주체에 태그 세트가 포함될 수 있습니다. 이러한 태그는 다음 소스에서 가져옵니다.

1. 세션 태그 - AWS CLI 또는 AWS API를 사용하여 역할을 수임하거나 사용자를 페더레이션할 때 전달되는 태그입니다. 이러한 작업에 대한 자세한 내용은 세션 태그 지정 작업 섹션을 참조하세요.
2. 수신 전이적 세션 태그 - 역할 체인의 이전 세션에서 상속되는 태그입니다. 자세한 내용은 이 주제의 후반부에서 세션 태그를 사용하는 역할 체인 섹션을 참조하세요.
3. IAM 태그 - IAM 수임된 역할에 연결된 태그입니다.

주제

세션 태그 지정 작업

AWS STS에서 다음 AWS CLI 또는 AWS API 작업을 사용하여 세션 태그를 전달할 수 있습니다. AWS Management Console 역할 전환 기능을 사용하여 세션 태그를 전달할 수 없습니다.

세션 태그를 전이적으로 설정할 수도 있습니다. 전이적 태그는 역할 체인 동안 지속됩니다. 자세한 내용은 세션 태그를 사용하는 역할 체인 단원을 참조하십시오.

다음 표에서는 세션 태그 전달 방법을 비교합니다.

Operation	역할을 위임할 수 있는 사용자	태그를 전달하는 방법	전이적 태그를 설정하는 방법
assume-role CLI 또는 AssumeRole API 작업	IAM 사용자 또는 세션	Tags API 파라미터 또는 --tags CLI 옵션	TransitiveTagKeys API 파라미터 또는 --transitive-tag-keys CLI 옵션
assume-role-with-saml CLI 또는 AssumeRol	SAML 자격 증명 공급자를 사용하여 인증된 모든 사용자	PrincipalTag SAML 속성	TransitiveTagKeys SAML 속성

Operation	역할을 위임할 수 있는 사용자	태그를 전달하는 방법	전이적 태그를 설정하는 방법
eWithSAML API 작업			
assume-role-with-web-identity CLI 또는 AssumeRoleWithWebIdentity API 작업	OIDC 공급자를 사용하여 인증된 모든 사용자	PrincipalTag OIDC 토큰	TransitiveTagKeys OIDC 토큰
get-federation-token CLI 또는 GetFederationToken API 작업	IAM 사용자 또는 루트 사용자	Tags API 파라미터 또는 --tags CLI 옵션	지원되지 않음

다음 조건에서는 세션 태그 지정을 지원하는 작업이 실패할 수 있습니다.

- 50개 이상의 세션 태그를 전달하는 경우
- 세션 태그 키의 일반 텍스트가 128자를 초과하는 경우
- 세션 태그 값의 일반 텍스트가 256자를 초과하는 경우
- 세션 정책의 일반 텍스트 총 크기가 2048자를 초과하는 경우
- 세션 정책과 태그를 합친 총 압축 크기가 너무 큰 경우. 작업이 실패할 경우 오류 메시지에 정책과 태그를 합친 크기가 크기 상한에 얼마나 가까운지가 백분율로 표시됩니다.

세션 태그에 대해 알아야 할 사항

세션 태그를 사용하기 전에 세션 및 태그에 대한 다음 세부 정보를 검토합니다.

- 세션 태그를 사용하는 경우 태그를 전달하는 자격 증명 공급자(IdP)에 연결된 모든 역할의 신뢰 정책에 `sts:TagSession` 권한이 있어야 합니다. 역할의 신뢰 정책에 이 권한이 없는 경우 `AssumeRole` 작업이 실패합니다.
- 세션을 요청할 때 보안 주체 태그를 세션 태그로 지정할 수 있습니다. 태그는 세션의 자격 증명을 사용하여 생성한 요청에 적용됩니다.
- 세션 태그는 키 값 페어를 사용합니다. 예를 들어, 세션에 연락처 정보를 추가하려면 세션 태그 키 `email` 및 태그 값 `johndoe@example.com`을 추가할 수 있습니다.
- 세션 태그는IAM 및 AWS STS의 태그 이름 지정 규칙을 따라야 합니다. 이 주제에는 세션 태그에 적용되는 대/소문자 구분 및 제한된 접두사에 대한 정보가 포함되어 있습니다.
- 새 세션 태그는 대소문자에 관계없이 동일한 태그 키의 기존에 맡은 역할 또는 페더레이션 사용자 태그를 재정의합니다.
- AWS Management Console을 사용하여 세션 태그를 전달할 수는 없습니다.
- 세션 태그는 현재 세션에서만 유효합니다.
- 세션 태그는 역할 체인을 지원합니다. 기본적으로 AWS STS에서는 후속 역할 세션에 태그를 전달하지 않습니다. 그러나 세션 태그를 전이적으로 설정할 수 있습니다. 전이적 태그는 역할 체인 중 유지되고 역할 신뢰 정책 평가 이후 일치하는 `ResourceTag` 값을 대체합니다. 자세한 내용은 세션 태그를 사용하는 역할 체인 단원을 참조하십시오.
- 세션 태그를 사용하여 리소스에 대한 액세스를 제어하거나 후속 세션에 전달할 수 있는 태그를 제어할 수 있습니다. 자세한 내용은 IAM 튜토리얼: ABAC에 SAML 세션 태그 사용 단원을 참조하십시오.
- AWS CloudTrail 로그에서 세션 태그를 비롯하여 세션의 보안 주체 태그를 볼 수 있습니다. 자세한 내용은 CloudTrail에서 세션 태그 보기 단원을 참조하십시오.
- 각 세션 태그에는 단일 값을 전달해야 합니다. AWS STS에서는 다중 값 세션 태그를 지원하지 않습니다.
- 최대 50개의 세션 태그를 전달할 수 있습니다. AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 단원을 참조하십시오.
- AWS 변환은 전달된 세션 정책과 세션 태그를 별도의 제한이 있는 압축된 이진 형식으로 압축합니다. 이 제한을 초과할 경우 AWS CLI 또는 AWS API 오류 메시지에 결합된 정책과 태그가 크기 상한에 얼마나 가까운지가 백분율로 표시됩니다.

세션 태그를 추가하는 데 필요한 권한

API 작업과 일치하는 작업 외에도 정책에는 다음과 같은 권한 전용 작업이 있어야 합니다.

```
sts:TagSession
```

> **Important**
>
> 세션 태그를 사용하는 경우 자격 증명 공급자(IdP)에 연결된 모든 역할에 대한 역할 신뢰 정책에 `sts:TagSession` 권한이 있어야 합니다. 이 권한 없이 세션 태그를 전달하는 IdP에 연결된 모든 역할의 경우 `AssumeRole` 작업이 실패합니다. 각 역할에 대한 역할 신뢰 정책을 업데이트하지 않으려는 경우 세션 태그를 전달하는 데 개별 IdP 인스턴스를 사용할 수 있습니다. 그런 다음 개별 IdP에 연결된 역할에만 `sts:TagSession` 권한을 추가합니다.

`sts:TagSession` 작업을 다음 조건 키와 함께 사용할 수 있습니다.

- `aws:PrincipalTag` - 이 키를 사용하여 요청한 보안 주체에 연결된 태그를 정책에서 지정한 태그와 비교합니다. 예를 들어, 요청을 하는 보안 주체에 지정된 태그가 있는 경우에만 보안 주체가 세션 태그를 전달하도록 허용할 수 있습니다.
- `aws:RequestTag` - 요청에서 전달된 태그 키 값 페어를 정책에서 지정한 태그 페어와 비교합니다. 예를 들어, 보안 주체가 지정된 세션 태그를 전달할 수는 있지만, 지정된 값만 사용하도록 허용할 수 있습니다.
- `aws:ResourceTag` - 정책에서 지정한 태그 키 값 페어를 리소스에 연결된 키 값 페어와 비교합니다. 예를 들어, 보안 주체가 수임하고 있는 역할에 지정된 태그가 포함된 경우에만 보안 주체가 세션 태그를 전달하도록 허용할 수 있습니다.
- `aws:TagKeys` - 요청의 태그 키를 정책에서 지정한 키와 비교합니다. 예를 들어, 보안 주체가 지정된 태그 키를 가진 세션 태그만 전달하도록 허용할 수 있습니다. 이 조건 키는 전달할 수 있는 최대 세션 태그 세트를 제한합니다.
- `sts:TransitiveTagKeys` - 요청의 전이적 세션 태그 키와 정책에 지정된 전이적 세션 태그 키를 비교합니다. 예를 들어, 보안 주체가 특정 태그만 전이적으로 설정하도록 허용하는 정책을 작성할 수 있습니다. 전이적 태그는 역할 체인 동안 지속됩니다. 자세한 내용은 세션 태그를 사용하는 역할 체인 단원을 참조하십시오.

예를 들어, 다음 역할 신뢰 정책은 `test-session-tags` 사용자가 정책이 연결된 역할을 수임할 수 있도록 허용합니다. 해당 사용자가 역할을 맡는 경우 AWS CLI 또는 AWS API를 사용하여 세 개의 필수 세션 태그와 필수 외부 ID를 전달해야 합니다. 또한 사용자는 `Project` 및 `Department` 태그를 전이적으로 설정하도록 선택할 수 있습니다.

Example 세션 태그에 대한 역할 신뢰 정책의 예

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AllowIamUserAssumeRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "arn:aws:iam::123456789012:user/test-session-tags"},
            "Condition": {
              "StringLike": {
                    "aws:RequestTag/Project": "*",
                    "aws:RequestTag/CostCenter": "*",
                    "aws:RequestTag/Department": "*"
                },
                "StringEquals": {"sts:ExternalId": "Example987"}
            }
        },
        {
            "Sid": "AllowPassSessionTagsAndTransitive",
            "Effect": "Allow",
            "Action": "sts:TagSession",
            "Principal": {"AWS": "arn:aws:iam::123456789012:user/test-session-tags"},
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/Project": "*",
                    "aws:RequestTag/CostCenter": "*"
                },
                "StringEquals": {
                    "aws:RequestTag/Department": [
                        "Engineering",
                        "Marketing"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "sts:TransitiveTagKeys": [
                        "Project",
                        "Department"
                    ]
                }
            }
        }
    ]
}
```

이 정책이 하는 일은 무엇입니까?

- AllowIamUserAssumeRole 문은 test-session-tags 사용자가 정책이 연결된 역할을 수임할 수 있도록 허용합니다. 해당 사용자가 역할을 맡을 때는 필수 세션 태그 및 외부 ID를 전달해야 합니다.
  - 이 문의 첫 번째 조건 블록의 경우 사용자가 Project, CostCenter 및 Department 세션 태그를 전달해야 합니다. 이 문에서 태그 값은 중요하지 않으므로 태그 값에 와일드카드(*)를 사용할 수 있습니다. 이 블록의 경우 사용자가 적어도 이 세 개의 세션 태그를 전달해야 하며 그렇지 않으면 작업이 실패합니다. 사용자는 추가 태그를 전달할 수 있습니다.
  - 두 번째 조건 블록의 경우 사용자가 Example987 값의 외부 ID를 전달해야 합니다.
- AllowPassSessionTagsAndTransitive 문은 sts:TagSession 권한 전용 작업을 허용합니다. 이 작업을 허용해야 사용자가 세션 태그를 전달할 수 있습니다. 정책에 두 번째 문은 없고 첫 번째 문만 포함되어 있는 경우 사용자는 역할을 맡을 수 없습니다.
  - 이 문의 첫 번째 조건 블록은 사용자가 CostCenter 및 Project 세션 태그에 대한 값을 전달하도록 허용합니다. 정책의 태그 값에 와일드카드(*)를 사용하여 이 작업을 수행하려면 StringLike 조건 연산자를 사용해야 합니다.
  - 두 번째 조건 블록은 사용자가 Department 세션 태그의 Engineering 또는 Marketing 값만 전달하도록 허용합니다.
  - 세 번째 조건 블록은 전이적으로 설정할 수 있는 최대 태그 세트를 나열합니다. 사용자는 하위 세트를 설정하거나 태그를 전이적으로 설정하지 않도록 선택할 수 있습니다. 그러나 추가 태그를 전이적으로 설정할 수는 없습니다. "Null":{"sts:TransitiveTagKeys":"false"}를 포함하는 다른 조건 블록을 추가하여 태그 중 하나 이상을 전이적으로 설정하도록 요구할 수 있습니다.

AssumeRole을 사용하여 세션 태그 전달

AssumeRole 작업은 AWS 리소스에 액세스하는 데 사용할 수 있는 임시 자격 증명 세트를 반환합니다. IAM 사용자 또는 역할 자격 증명을 사용하여 AssumeRole을 호출할 수 있습니다. 역할을 맡는 동안 세션 태그를 전달하려면 --tags AWS CLI 옵션 또는 Tags AWS API 파라미터를 사용합니다.

태그를 전이적으로 설정하려면 --transitive-tag-keys AWS CLI 옵션 또는 TransitiveTagKeys AWS API 파라미터를 사용합니다. 전이적 태그는 역할 체인 동안 지속됩니다. 자세한 내용은 세션 태그를 사용하는 역할 체인 단원을 참조하십시오.

다음 예제에서는 AssumeRole을 사용하는 샘플 요청을 보여 줍니다. 이 예제에서는 my-role-example 역할을 맡을 때 my-session이라는 세션을 생성합니다. 세션 태그 키 값 페어 Project = Automation, CostCenter = 12345 및 Department = Engineering을 추가합니다. 또한 해당 키를 지정하여 Project 및 Department 태그를 전이적으로 설정합니다. 각 세션 태그에는 단일 값을 전달해야 합니다. AWS STS에서는 다중 값 세션 태그를 지원하지 않습니다.

Example AssumeRole CLI 요청의 예

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/my-role-example \
--role-session-name my-session \
--tags Key=Project,Value=Automation Key=CostCenter,Value=12345
 Key=Department,Value=Engineering \
--transitive-tag-keys Project Department \
--external-id Example987
```

AssumeRoleWithSAML을 사용하여 세션 태그 전달

`AssumeRoleWithSAML` 작업은 SAML 기반 페더레이션을 사용하여 인증됩니다. 이 작업은 AWS 리소스에 액세스하는 데 사용할 수 있는 임시 자격 증명 세트를 반환합니다. AWS Management Console 액세스를 위해 SAML 기반 연동을 사용하는 방법에 대한 자세한 내용은 SAML 2.0 페더레이션 사용자가 AWS Management Console에 액세스할 수 있게 하기 섹션을 참조하세요. AWS CLI 또는 AWS API 액세스에 대한 자세한 내용은 SAML 2.0 연동 섹션을 참조하세요. Active Directory 사용자를 위한 SAML 페더레이션을 구성하는 방법에 관한 자습서는 AWS 보안 블로그에서 Active Directory Federation Services(ADFS)를 사용한 AWS 페더레이션 인증을 참조하세요.

관리자는 회사 디렉터리의 멤버가 AWS STS `AssumeRoleWithSAML` 작업을 사용하여 AWS로 연동하도록 허용할 수 있습니다. 이렇게 하려면 다음 작업을 완료해야 합니다.

1. AWS에 대한 SAML 공급자로 네트워크 구성
2. IAM에서 SAML 공급자 생성
3. 페더레이션 사용자를 위해 AWS에서 역할 및 권한 구성
4. SAML IdP 구성을 완료하고 SAML 인증 응답에 대한 어설션 생성하기

AWS에는 자격 증명 솔루션에 세션 태그에 대한 인증된 엔드 투 엔드 환경이 있는 파트너가 포함되어 있습니다. 이러한 자격 증명 공급자를 사용하여 세션 태그를 구성하는 방법은 서드 파티 SAML 솔루션 공급자를 AWS와 통합 섹션을 참조하세요.

SAML 속성을 세션 태그로 전달하려면 `Name` 속성이 `https://aws.amazon.com/SAML/Attributes/PrincipalTag:`*`{TagKey}`*로 설정된 `Attribute` 요소를 포함합니다. `AttributeValue` 요소를 사용하여 태그 값을 지정합니다. 각 세션 태그마다 별도의 `Attribute` 요소를 포함합니다.

예를 들어, 다음 자격 증명 속성을 세션 태그로 전달한다고 가정합니다.

- `Project:Automation`
- `CostCenter:12345`
- `Department:Engineering`

이러한 속성을 전달하려면 SAML 어설션에 다음 요소를 포함합니다.

Example SAML 어설션의 코드 조각 예

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Project">
  <AttributeValue>Automation</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:CostCenter">
  <AttributeValue>12345</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Department">
  <AttributeValue>Engineering</AttributeValue>
</Attribute>
```

앞의 태그를 전이적으로 설정하려면 `Name` 속성이 `https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys`로 설정된 다른 `Attribute` 요소를 포함합니다. 전이적 태그는 역할 체인 동안 지속됩니다. 자세한 내용은 세션 태그를 사용하는 역할 체인 섹션을 참조하세요.

`Project` 및 `Department` 태그를 전이적으로 설정하려면 다음과 같은 다중 값 속성을 사용합니다.

Example SAML 어설션의 코드 조각 예

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys">
  <AttributeValue>Project</AttributeValue>
  <AttributeValue>Department</AttributeValue>
</Attribute>
```

AssumeRoleWithWebIdentity를 사용하여 세션 태그 전달

OIDC(OpenID Connect) 호환 페더레이션을 사용하여 `AssumeRoleWithWebIdentity` 작업을 인증합니다. 이 작업은 AWS 리소스에 액세스하는 데 사용할 수 있는 임시 자격 증명 세트를 반환합니다. AWS Management Console 액세스를 위해 웹 아이덴티티 페더레이션을 사용하는 방법에 대한 자세한 내용은 OIDC 페더레이션 섹션을 참조하세요.

OIDC(OpenID Connect)에서 세션 태그를 전달하려면 JWT(JSON 웹 토큰)에 세션 태그를 포함해야 합니다. `AssumeRoleWithWebIdentity` 요청을 제출할 때 토큰의 `https://`

aws.amazon.com/ tags 네임스페이스에 세션 태그를 포함합니다. OIDC 토큰 및 클레임에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 사용자 풀과 함께 토큰 사용을 참조하세요.

예를 들어, 다음의 디코딩된 JWT는 토큰을 사용하여 Project, CostCenter 및 Department 세션 태그로 AssumeRoleWithWebIdentity를 호출합니다. 또한 토큰은 Project 및 CostCenter 태그를 전이적으로 설정합니다. 전이적 태그는 역할 체인 동안 지속됩니다. 자세한 내용은 세션 태그를 사용하는 역할 체인 섹션을 참조하세요.

Example 디코딩된 JSON 웹 토큰의 예

```
{
    "sub": "johndoe",
    "aud": "ac_oic_client",
    "jti": "ZYUCeRMQVtqHypVPWAN3VB",
    "iss": "https://xyz.com",
    "iat": 1566583294,
    "exp": 1566583354,
    "auth_time": 1566583292,
    "https://aws.amazon.com/tags": {
        "principal_tags": {
            "Project": ["Automation"],
            "CostCenter": ["987654"],
            "Department": ["Engineering"]
        },
        "transitive_tag_keys": [
            "Project",
            "CostCenter"
        ]
    }
}
```

GetFederationToken을 사용하여 세션 태그 전달

GetFederationToken을 사용하면 사용자를 페더레이션할 수 있습니다. 이 작업은 AWS 리소스에 액세스하는 데 사용할 수 있는 임시 자격 증명 세트를 반환합니다. 페더레이션 사용자 세션에 태그를 추가하려면 --tags AWS CLI 옵션 또는 Tags AWS API 파라미터를 사용합니다. GetFederationToken을 사용할 경우 세션 태그를 전이적으로 설정할 수 없습니다. 임시 자격 증명을 사용하여 역할을 수임할 수 없기 때문입니다. 이 경우 역할 체인을 사용할 수 없습니다.

다음 예제에서는 GetFederationToken을 사용하는 샘플 요청을 보여 줍니다. 이 예제에서는 토큰을 요청할 때 my-fed-user라는 세션을 생성합니다. 세션 태그 키 값 페어 Project = Automation 및 Department = Engineering을 추가합니다.

Example GetFederationToken CLI 요청의 예

```
aws sts get-federation-token \
--name my-fed-user \
--tags key=Project,value=Automation key=Department,value=Engineering
```

GetFederationToken 작업에서 반환되는 임시 자격 증명을 사용하는 경우 세션의 보안 주체 태그에 사용자의 태그와 전달된 세션 태그가 포함됩니다.

세션 태그를 사용하는 역할 체인

한 역할을 맡은 다음 임시 자격 증명을 사용하여 다른 역할을 맡을 수 있습니다. 이러한 작업을 세션 간에 계속할 수 있습니다. 이를 역할 체인이라고 합니다. 역할을 맡는 동안 세션 태그를 전달하면 키를 전이적으로 설정할 수 있습니다. 이렇게 하면 해당 세션 태그가 역할 체인의 후속 세션에 전달됩니다. 역할 태그를 전이적으로 설정할 수 없습니다. 이러한 태그를 후속 세션에 전달하려면 세션 태그로 지정합니다.

> **Note**
>
> 전이적 태그는 역할 체인 중 유지되고 역할 신뢰 정책 평가 이후 일치하는 ResourceTag 값을 대체합니다.

다음 예제에서는 AWS STS에서 세션 태그, 전이적 태그 및 역할 태그를 역할 체인의 후속 세션에 전달하는 방식을 보여줍니다.

이 역할 체인 시나리오 예제에서는 AWS CLI에서 IAM 사용자 액세스 키를 사용하여 Role1이라는 역할을 수임합니다. 그런 다음 결과 세션 자격 증명을 사용하여 Role2라는 두 번째 역할을 맡습니다. 그런 다음 두 번째 세션 자격 증명을 사용하여 Role3라는 세 번째 역할을 맡을 수 있습니다. 이러한 요청은 세 가지 개별 작업으로 발생합니다. IAM에서는 각 역할은 이미 태깅되어 있습니다. 그리고 각 요청 중에 추가 세션 태그를 전달합니다.

역할을 체인할 때 이전 세션의 태그가 이후 세션에서도 유지되도록 할 수 있습니다. assume-role CLI 명령을 사용하여 이 작업을 수행하려면 태그를 세션 태그로 전달하고 전이적으로 설정해야 합니다. 태그 Star = 1을 세션 태그로 전달합니다. 또한 이 명령은 태그 Heart = 1을 역할에 연결하고 세션을 사용할 때 보안 주체 태그로 적용합니다. 그러나 Heart = 1 태그가 두 번째 또는 세 번째 세션에 자동으로 전달되도록 할 수도 있습니다. 이를 수행하려면 수동으로 세션 태그로 포함합니다. 결과 세션 보안 주체 태그에는 이러한 두 태그가 모두 포함되며 전이적으로 설정됩니다.

다음 AWS CLI 명령을 사용하여 이 요청을 수행합니다.

Example AssumeRole CLI 요청의 예

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Role1 \
--role-session-name Session1 \
--tags Key=Star,Value=1 Key=Heart,Value=1 \
--transitive-tag-keys Star Heart
```

그런 다음 해당 세션에 대한 자격 증명을 사용하여 Role2를 맡습니다. 이 명령은 태그 Sun = 2를 두 번째 역할에 연결하고 두 번째 세션을 사용할 때 보안 주체 태그로 적용합니다. Heart 및 Star 태그는 첫 번째 세션의 전이적 세션 태그를 상속합니다. 두 번째 세션의 결과 보안 주체 태그는 Heart = 1, Star = 1 및 Sun = 2입니다. Heart 및 Star는 계속 전이적으로 유지됩니다. Role2에 연결된 Sun 태그는 세션 태그가 아니므로 전이적으로 표시되지 않습니다. 향후 세션에서는 이 태그를 상속하지 않습니다.

다음 AWS CLI 명령을 사용하여 이 두 번째 요청을 수행합니다.

Example AssumeRole CLI 요청의 예

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Role2 \
--role-session-name Session2
```

그런 다음 두 번째 세션 자격 증명을 사용하여 Role3를 맡습니다. 세 번째 세션의 보안 주체 태그는 새 세션 태그, 상속된 전이적 세션 태그 및 역할 태그에서 가져옵니다. 두 번째 세션의 Heart = 1 및 Star = 1 태그는 첫 번째 세션의 전이적 세션 태그에서 상속됩니다. Sun = 2 세션 태그를 전달하려고 하면 작업이 실패합니다. 상속된 Star = 1 세션 태그는 역할의 Star = 3 태그를 재정의합니다. 역할 체인에서 전이적 태그의 값은 역할 신뢰 정책 평가 후에 ResourceTag 값과 일치하는 역할을 재정의합니다. 이 예제에서는 Role3이 Star를 역할 신뢰 정책에서 ResourceTag로 사용하는 경우 ResourceTag 값을 호출 역할 세션의 전이 태그 값으로 설정합니다. 역할의 Lightning 태그는 세 번째 세션에도 적용되며 전이적으로 설정되지 않습니다.

다음 AWS CLI 명령을 사용하여 세 번째 요청을 수행합니다.

Example AssumeRole CLI 요청의 예

```
aws sts assume-role \
```

```
--role-arn arn:aws:iam::123456789012:role/Role3 \
--role-session-name Session3
```

ABAC에 세션 태그 사용

속성 기반 액세스 제어(ABAC)는 태그 속성을 기반으로 권한을 정의하는 권한 부여 전략을 사용합니다.

회사에서 OIDC 또는 SAML 기반 자격 증명 공급자(IdP)를 사용하여 사용자 자격 증명을 관리하는 경우 세션 태그를 AWS에 전달하도록 어설션을 구성할 수 있습니다. 예를 들어 회사 사용자 자격 증명을 사용하면 직원이 AWS로 페더레이션할 때 AWS는 해당 속성을 결과 보안 주체에 적용합니다. 그런 다음 ABAC를 사용하여 이러한 속성에 따라 권한을 허용하거나 거부할 수 있습니다. 자세한 내용은 IAM 튜토리얼: ABAC에 SAML 세션 태그 사용 섹션을 참조하세요.

ABAC에 IAM Identity Center를 사용하는 방법에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 액세스 제어를 위한 속성을 참조하세요.

CloudTrail에서 세션 태그 보기

AWS CloudTrail을 사용하여 역할을 수임하거나 사용자를 페더레이션하는 데 사용되는 요청을 볼 수 있습니다. CloudTrail 로그 파일에는 수임한 역할 또는 페더레이션 사용자 세션의 보안 주체 태그에 대한 정보가 포함됩니다. 자세한 내용은 AWS CloudTrail을 사용하여 IAM 및 AWS STS API 호출 로깅 섹션을 참조하세요.

예를 들어, AWS STS `AssumeRoleWithSAML` 요청을 하고 세션 태그를 전달하고 해당 태그를 전이적으로 설정한다고 가정합니다. CloudTrail 로그에서 다음 정보를 찾을 수 있습니다.

Example AssumeRoleWithSAML CloudTrail 로그의 예

```
    "requestParameters": {
        "sAMLAssertionID": "_c0046cEXAMPLEb9d4b8eEXAMPLE2619aEXAMPLE",
        "roleSessionName": "MyRoleSessionName",
        "principalTags": {
            "CostCenter": "987654",
            "Project": "Unicorn"
        },
        "transitiveTagKeys": [
            "CostCenter",
            "Project"
        ],
        "durationSeconds": 3600,
        "roleArn": "arn:aws:iam::123456789012:role/SAMLTestRoleShibboleth",
```

```
        "principalArn": "arn:aws:iam::123456789012:saml-provider/Shibboleth"
    },
```

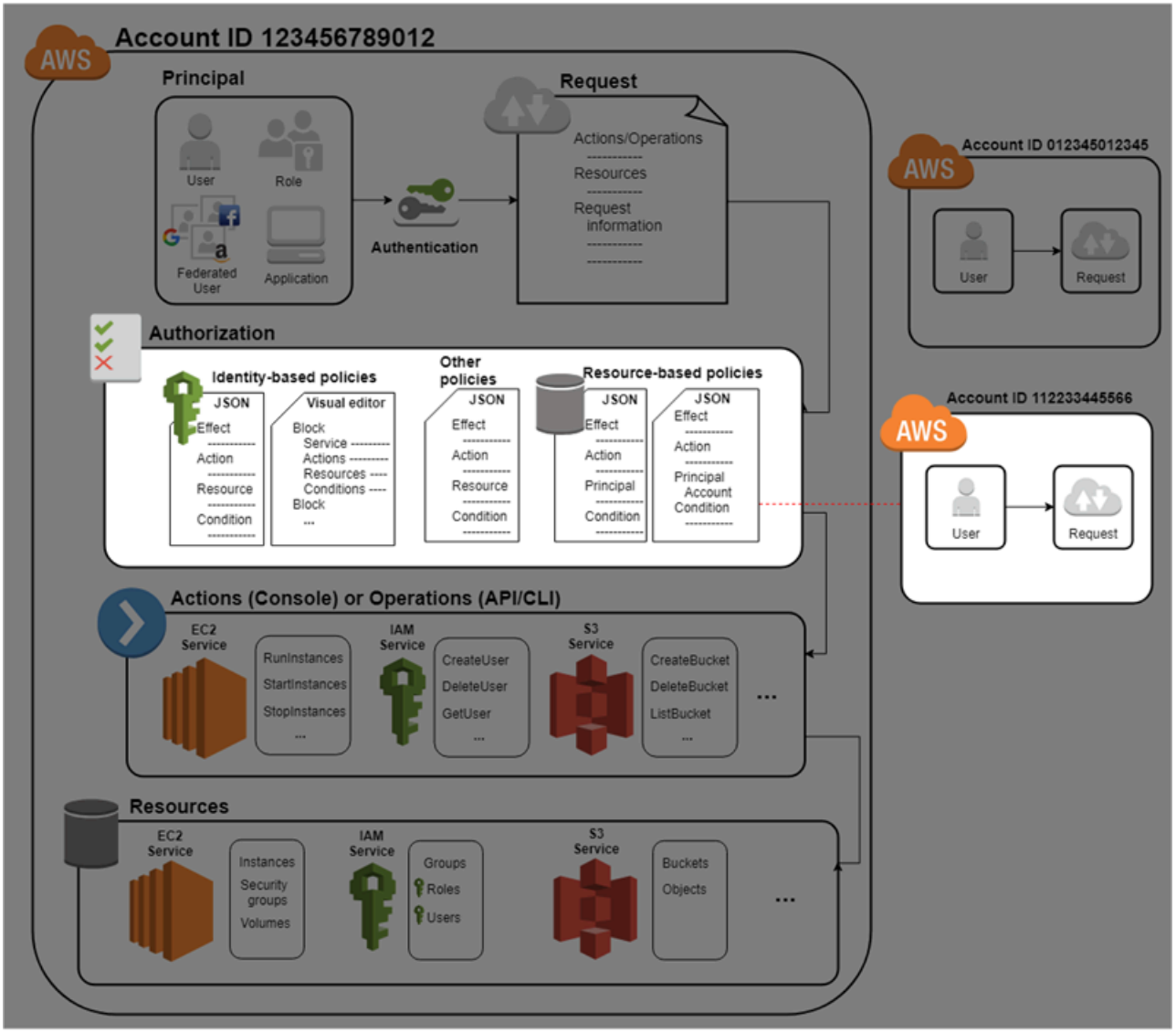
다음 CloudTrail 로그 예제에서 세션 태그를 사용하는 이벤트를 볼 수 있습니다.

- CloudTrail 로그 파일의 AWS STS 역할 체인 API 이벤트의 예
- CloudTrail 로그 파일의 SAML AWS STS API 이벤트의 예
- CloudTrail 로그 파일의 예시 OIDC AWS STS API 이벤트

AWS리소스에 대한 액세스 관리

AWS Identity and Access Management(IAM)은 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있는 웹 서비스입니다. 보안 주체가 AWS에 요청하면 AWS 적용 코드는 해당 보안 주체가 인증(로그인) 및 권한 부여(권한 있음)되었는지 확인합니다. 정책을 생성하고 IAM 자격 증명 또는 AWS 리소스에 연결하여 AWS 액세스를 관리합니다. 정책은 자격 증명 또는 리소스에 연결될 때 해당 권한을 정의하는 AWS의 JSON 정책 문서입니다. 정책 유형 및 활용에 대한 자세한 정보는 IAM의 정책 및 권한 섹션을 참조하세요.

인증 및 권한 부여 프로세스의 나머지 부분에 대한 자세한 정보는 IAM 작동 방식 섹션을 참조하세요.



권한 부여 중 AWS 적용 코드는 요청 콘텍스트의 값을 사용하여 일치하는 정책을 확인하고 요청을 허용할지 거부할지 여부를 결정합니다.

AWS은 요청 콘텍스트에 적용되는 각 정책을 확인합니다. 단일 정책이 요청을 거부한 경우 AWS는 전체 요청을 거부하고 정책 평가를 중지합니다. 이를 명시적 거부라고 합니다. 요청은 기본적으로 거부되므로 IAM은 사용 가능한 정책이 요청의 모든 부분을 허용하는 경우에만 요청에 권한을 부여합니다. 단일 계정 내 요청 평가 로직은 다음 규칙을 따릅니다.

- 기본적으로 모든 요청이 묵시적으로 거부됩니다. 또는 기본적으로 AWS 계정 루트 사용자에 모든 권한이 부여됩니다.
- 자격 증명 기반 또는 리소스 기반 정책에 포함된 명시적 허용은 이 기본 작동을 재정의합니다.
- 권한 경계, 조직 SCP 또는 세션 정책이 있는 경우 이러한 정책 유형이 명시적 거부로 허용을 재정의할 수도 있습니다.
- 어떠한 정책의 명시적 거부도 허용을 무시합니다.

요청이 인증 및 권한 부여된 후 AWS이 요청을 승인합니다. 다른 계정에서 요청해야 하는 경우 다른 계정의 정책에서 요청자에게 해당 리소스에 대한 액세스를 허용해야 합니다. 또한 요청하는 데 사용하는 IAM 엔터티에 해당 요청을 허용하는 자격 증명 기반 정책이 있어야 합니다.

액세스 관리 리소스

권한 및 정책 생성에 대한 자세한 정보는 다음 리소스를 참조하세요.

AWS 보안 블로그의 다음 게시물에서는 Amazon S3 버킷과 객체에 액세스하기 위한 정책을 작성하는 일반적인 방법을 소개합니다.

- IAM 정책 작성: Amazon S3 버킷에 대한 액세스 권한을 부여하는 방법
- IAM 정책 작성: Amazon S3 버킷의 사용자별 폴더에 대한 액세스 권한 부여
- IAM 정책 및 버킷 정책과 ACLs! ACL (S3 리소스에 대한 액세스 제어)
- RDS 리소스 수준 권한에 대한 소개
- EC2 리소스 수준 권한 설명

IAM의 정책 및 권한

정책을 생성하고 IAM 자격 증명(사용자, 사용자 그룹 또는 역할) 또는 AWS 리소스에 연결하여 AWS에서 액세스를 관리합니다. 정책은 자격 증명이나 리소스와 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. AWS는 IAM 보안 주체(사용자 또는 역할)가 요청을 보낼 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 AWS에 JSON 문서로 저

장됩니다. AWS에서는 자격 증명 기반 정책, 리소스 기반 정책, 권한 경계, Organizations SCP, ACL 및 세션 정책이라는 여섯 가지 정책 유형을 지원합니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, 정책이 GetUser 작업을 허용한다면 이 정책이 있는 사용자는 AWS Management Console, AWS CLI, 또는 AWS API에서 사용자 정보를 얻을 수 있습니다. IAM 사용자를 생성할 경우 콘솔 또는 프로그래밍 방식 액세스를 허용하도록 선택할 수 있습니다. 콘솔 액세스가 허용되는 경우 IAM 사용자는 로그인 보안 인증 정보를 사용하여 콘솔에 로그인할 수 있습니다. 프로그래밍 방식의 액세스가 허용되는 경우 사용자는 액세스 키를 사용하여 CLI 또는 API로 작업할 수 있습니다.

정책 유형

다음의 정책 유형은 AWS에서 사용할 수 있으며 가장 자주 사용하는 정책 유형에서 빈도가 낮은 정책 유형순으로 나열됩니다. 자세한 정보는 각 정책 유형에 따른 섹션을 참조하세요.

- 아이덴티티 기반 정책 - 관리형 및 인라인 정책을 IAM 아이덴티티(사용자, 사용자가 속한 그룹 또는 역할)에 연결합니다. 자격 증명 기반 정책은 자격 증명에 권한을 부여합니다.
- 리소스 기반 정책 - 인라인 정책을 리소스에 연결합니다. 리소스 기반 정책의 가장 일반적인 예제는 Amazon S3 버킷 정책 및 IAM 역할 신뢰 정책입니다. 리소스 기반 정책은 정책에 지정된 보안 주체에 권한을 부여합니다. 보안 주체는 리소스와 동일한 계정 또는 다른 계정에 있을 수 있습니다.
- 권한 경계 - 관리형 정책을 IAM 엔터티(사용자 또는 역할)에 대한 권한 경계로 사용합니다. 해당 정책은 자격 증명 기반 정책을 통해 엔터티에 부여할 수 있는 최대 권한을 정의하지만, 권한을 부여하지는 않습니다. 권한 경계는 리소스 기반 정책을 통해 엔터티에 부여할 수 있는 최대 권한을 정의하지 않습니다.
- Organizations SCP - AWS Organizations 서비스 제어 정책(SCP)을 사용하여 조직 또는 조직 단위(OU)의 계정 멤버에 대한 최대 권한을 정의합니다. SCP는 자격 증명 기반 정책이나 리소스 기반 정책을 통해 계정 내 엔터티(사용자나 역할)에 부여하는 권한을 제한하지만, 권한을 부여하지는 않습니다.
- 액세스 제어 목록(ACL) - ACL을 사용하여 ACL이 연결된 리소스에 액세스할 수 있는 다른 계정의 보안 주체를 제어합니다. ACL는 리소스 기반 정책과 비슷합니다. 다만 JSON 정책 문서 구조를 사용하지 않은 유일한 정책 유형입니다. ACL은 지정된 보안 주체에 권한을 부여하는 크로스 계정 권한 정책입니다. ACL은 동일 계정 내 엔터티에 권한을 부여할 수 없습니다.
- 세션 정책 - AWS CLI 또는 AWS API를 사용하여 역할이나 페더레이션 사용자를 수임할 때 고급 세션 정책을 전달합니다. 세션 정책은 역할이나 사용자의 자격 증명 기반 정책을 통해 세션에 부여하는 권한을 제한합니다. 세션 정책은 생성된 세션에 대한 권한을 제한하지 않지만, 권한을 부여하지도 않습니다. 자세한 정보는 세션 정책을 참조하세요.

보안 인증 기반 정책

자격 증명 기반 정책은 자격 증명(사용자, 사용자 그룹 및 역할)이 무슨 작업을 어느 리소스에서 어떤 조건에서 수행할 수 있는지를 제어하는 JSON 권한 정책 문서입니다. 자격 증명 기반 정책을 추가로 분류할 수 있습니다.

- 관리형 정책 - AWS 계정에 속한 다수의 사용자, 그룹 및 역할에 독립적으로 연결할 수 있는 자격 증명 기반 정책입니다. 두 가지 유형의 관리형 정책이 있습니다.
  - AWS 관리형 정책 - AWS에서 생성 및 관리하는 관리형 정책입니다.
  - 고객 관리형 정책 - 사용자가 자신의 AWS 계정에서 생성 및 관리하는 관리형 정책입니다. 고객 관리형 정책은 AWS 관리형 정책보다 정책에 대해 더욱 정밀하게 제어할 수 있습니다.
- 인라인 정책 - 단일 사용자, 그룹 또는 역할에 직접 추가하는 정책입니다. 인라인 정책은 정책과 자격 증명을 정확히 1대 1 관계로 유지합니다. 이는 자격 증명을 삭제하면 삭제됩니다.

관리형 정책을 사용할지 아니면 인라인 정책을 사용할지를 선택하는 방법은 관리형 정책과 인라인 정책의 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 Amazon S3 버킷과 같은 리소스에 연결하는 JSON 정책 문서입니다. 이러한 정책은 지정된 보안 주체에 해당 리소스에 대한 특정 작업을 수행할 수 있는 권한을 부여하고 이러한 권한이 적용되는 조건을 정의합니다. 리소스 기반 정책은 인라인 정책입니다. 관리형 리소스 기반 정책은 없습니다.

크로스 계정 액세스를 활성화하려는 경우 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 또한 보안 주체와 리소스가 별도의 AWS 계정에 있는 경우 자격 증명 기반 정책을 사용하여 보안 주체에 리소스에 대한 액세스 권한을 부여해야 합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 교차 서비스 액세스 권한을 부여하기 위한 단계별 지침은 튜토리얼: IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임 섹션을 참조하세요.

IAM 서비스는 역할 신뢰 정책이라고 하는 리소스 기반 정책 유형 하나만 지원하며, 이 유형은 IAM 역할에 연결됩니다. IAM 역할은 자격 증명이기도 하고 리소스 기반 정책을 지원하는 리소스이기도 합니다. 그러므로 IAM 역할에 신뢰 정책과 자격 증명 기반 정책을 모두 연결해야 합니다. 신뢰 정책은 역할을 수임할 수 있는 보안 주체 엔터티(계정, 사용자, 역할 및 페더레이션 사용자)를 정의합니다. IAM 역할과 다른 리소스 기반 정책 간의 차이에 대해 알아보려면 IAM의 크로스 계정 리소스 액세스 섹션을 참조하세요.

리소스 기반 정책을 지원하는 다른 서비스를 확인하려면 AWS IAM으로 작업하는 서비스 섹션을 참조하세요. 리소스 기반 정책에 대해 자세히 알아보려면 자격 증명 기반 정책 및 리소스 기반 정책 섹션을 참조하세요. 해당 신뢰 영역(신뢰할 수 있는 조직 또는 계정) 외의 계정 내 보안 주체가 역할을 수임하는 권한이 있는지 자세히 알고 싶다면, IAM Access Analyzer란 무엇일까요?를 참조하세요.

IAM 권한 경계

권한 경계는 자격 증명 기반 정책을 통해 IAM 엔터티에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 엔터티에 대한 권한 경계를 설정할 경우 해당 엔터티는 자격 증명 기반 정책 및 관련 권한 경계 모두에서 허용되는 작업만 수행할 수 있습니다. 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계에 제한을 받지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 엔터티의 권한 범위 섹션을 참조하세요.

서비스 제어 정책(SCP)

AWS Organizations는 기업이 소유하는 AWS 계정을 그룹화하고 중앙에서 관리할 수 있는 서비스입니다. 조직에서 모든 기능을 활성화할 경우, 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 조직 또는 조직 단위(OU)에 최대 권한을 지정하는 JSON 정책입니다. SCP는 각 AWS 계정 루트 사용자을 비롯하여 멤버 계정의 엔터티에 대한 권한을 제한합니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다.

Organizations 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하세요.

액세스 제어 목록(ACL)

ACL(액세스 제어 목록)은 리소스에 액세스할 수 있는 다른 계정의 보안 주체를 제어할 수 있는 서비스 정책입니다. ACL은 동일 계정 내에서 보안 주체에 대한 액세스를 제어하는 데 사용할 수 없습니다. ACL는 리소스 기반 정책과 비슷합니다. 다만 JSON 정책 문서 형식을 사용하지 않은 유일한 정책 유형입니다. Amazon S3, AWS WAF 및 Amazon VPC는 ACL을 지원하는 대표적인 서비스입니다. ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 안내서의 ACL(액세스 제어 목록) 개요를 참조하세요.

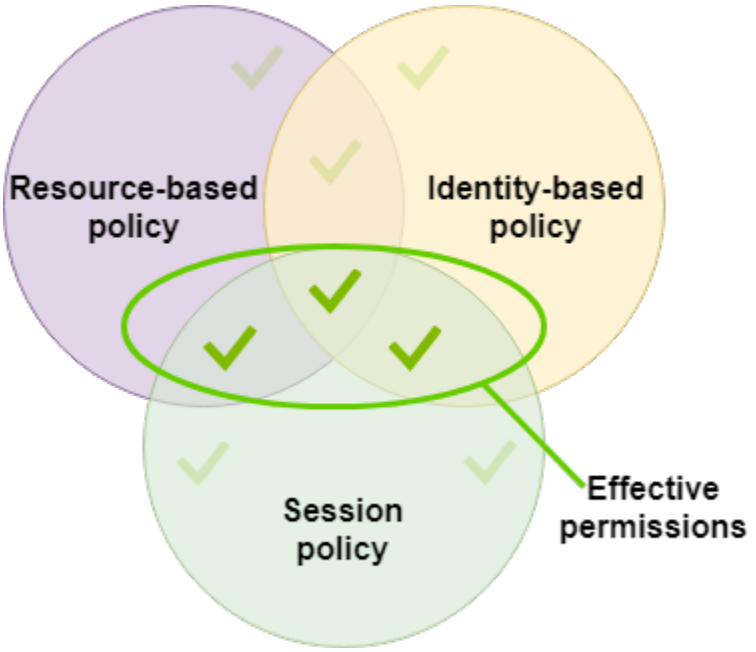
세션 정책

세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 세션에 대한 권한은 세션을 생성하는 데 사용되는 IAM 엔터티(사용자 또는 역할)에 대한 자격 증명 기반 정책과 세션 정책의 교집합입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다.

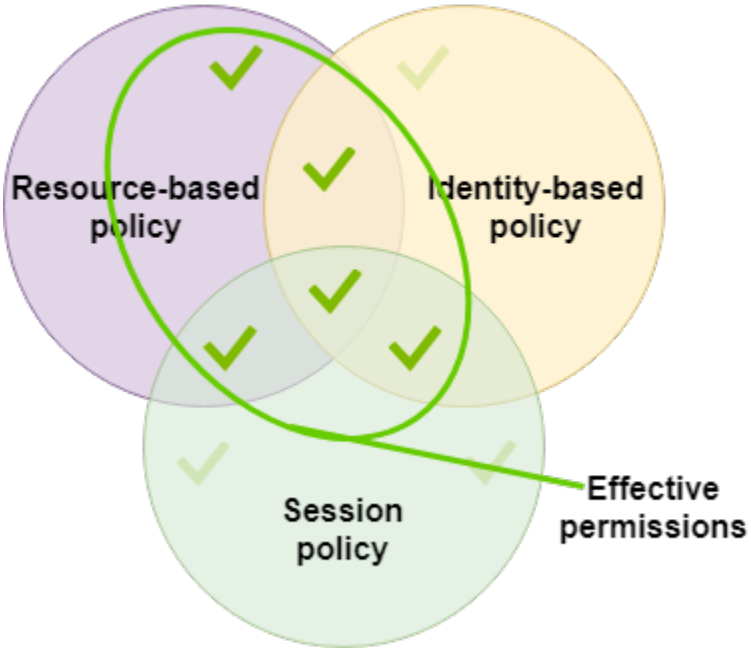
`AssumeRole`, `AssumeRoleWithSAML` 또는 `AssumeRoleWithWebIdentity` API 작업을 사용하여 프로그래밍 방식으로 역할 세션을 생성하고 세션 정책을 전달할 수 있습니다. `Policy` 파라미터를 사용하여 단일 JSON 인라인 세션 정책 문서를 전달할 수 있습니다. `PolicyArns` 파라미터를 사용하여 최대 10개까지 관리형 세션 정책을 지정할 수 있습니다. 역할 세션 생성에 대한 자세한 정보는 임시 보안 자격 증명 요청 섹션을 참조하세요.

페더레이션 사용자 세션을 생성할 경우 IAM 사용자의 액세스 키를 사용하여 `GetFederationToken` API 작업을 프로그래밍 방식으로 호출할 수 있습니다. 또한 세션 정책도 전달해야 합니다. 결과적으로 얻는 세션의 권한은 자격 증명 기반 정책과 세션 정책의 교집합입니다. 페더레이션 사용자 생성에 대한 자세한 정보는 GetFederationToken - 사용자 지정 아이덴티티 브로커를 통한 페더레이션 단원을 참조하십시오.

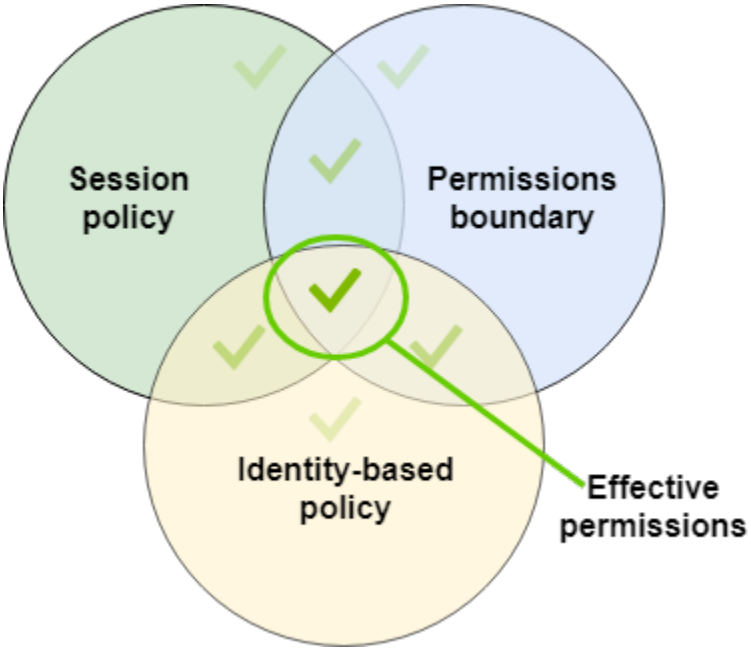
리소스 기반 정책에서 사용자 또는 역할의 ARN을 보안 주체로 지정할 수 있습니다. 이 경우, 세션이 생성되기 전에 리소스 기반 정책의 권한이 역할 또는 사용자의 자격 증명 기반 정책에 추가됩니다. 이 세션 정책은 리소스 기반 정책 및 자격 증명 기반 정책을 통해 부여되는 모든 권한을 제한합니다. 결과적으로 얻는 세션의 권한은 세션 정책과 리소스 기반 정책에 추가로 자격 증명 기반 정책의 교집합입니다.



리소스 기반 정책에서 세션의 ARN을 보안 주체로 지정할 수 있습니다. 이 경우, 세션이 생성된 후 리소스 기반 정책의 권한이 추가됩니다. 리소스 기반 정책 권한은 세션 정책에 제한을 받지 않습니다. 결과 세션에는 리소스 기반 정책의 모든 권한 + 자격 증명 기반 정책과 세션 정책의 권한 교집합이 부여됩니다.



권한 경계를 사용하여 세션을 생성하는 데 사용되는 사용자 또는 역할의 최대 권한 설정할 수 있습니다. 이 경우, 결과적으로 얻는 세션의 권한은 세션 정책, 권한 경계 및 자격 증명 기반 정책의 교집합입니다. 단, 권한 경계는 결과 세션의 ARN을 지정하는 리소스 기반 정책이 부여하는 권한을 제한할 수 없습니다.



정책 및 루트 사용자

AWS 계정 루트 사용자는 어떤 정책에는 영향을 받지만 이외의 정책에는 영향을 받지 않습니다. 자격 증명 기반 정책을 루트 사용자로 연결할 수 없고 루트 사용자에 대한 권한 경계를 설정할 수 없습니다. 그러나, 루트 사용자를 리소스 기반 정책 또는 ACL의 보안 주체로 지정할 수 있습니다. 루트 사용자는 여전히 계정의 멤버입니다. 계정이 AWS Organizations의 조직 멤버인 경우 루트 사용자는 계정의 SCP에 의해 영향을 받습니다.

JSON 정책 개요

대부분의 정책은 AWS에 JSON 문서로 저장됩니다. 자격 증명 기반 정책, 경계를 설정할 수 있는 정책은 사용자 또는 역할에 연결할 수 있는 JSON 정책 문서입니다. 리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. SCP는 AWS Organizations 조직 단위(OU)에 연결하는 제한된 구문이 있는 JSON 정책 문서입니다. ACL은 리소스에도 연결되지만 다른 구문을 사용해야 합니다. 세션 정책은 역할 또는 페더레이션 사용자 세션을 수임할 때 제공하는 JSON 정책입니다.

JSON 구문을 이해할 필요가 없습니다. AWS Management Console의 시각적 편집기를 사용하면 JSON을 사용하지 않고 고객 관리형 정책을 생성하고 편집할 수 있습니다. 그러나 그룹 또는 복잡한 정책에 대해 인라인 정책을 사용하는 경우에는 콘솔을 사용하여 JSON 편집기에서 해당 정책을 생성하고 편집해야 합니다. 시각적 편집기 사용에 대한 자세한 정보는 고객 관리형 정책으로 사용자 지정 IAM 권한 정의 및 IAM 정책 편집 섹션을 참조하세요.

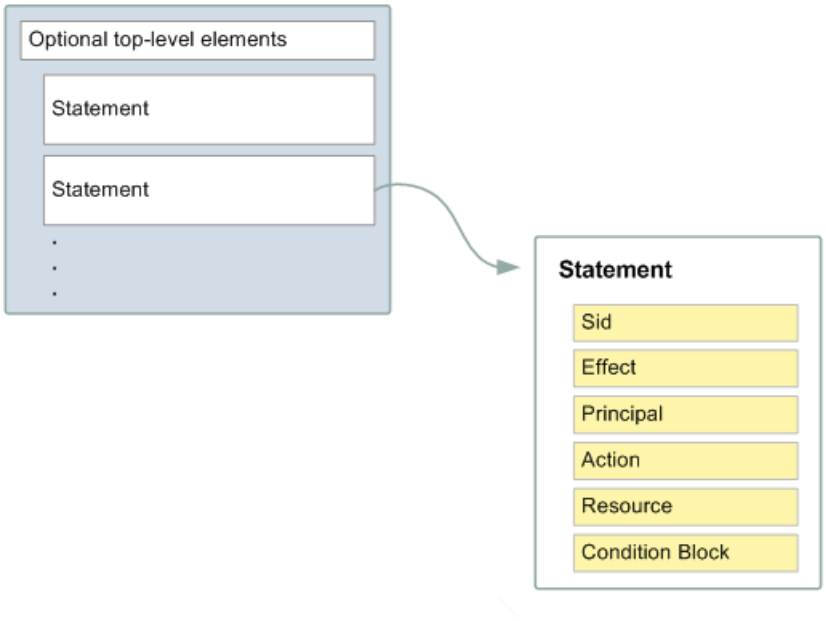
JSON 정책을 생성하거나 편집할 때 IAM은 효과적인 정책을 생성하는 데 도움이 되는 정책 검증을 수행할 수 있습니다. IAM은 JSON 구문 오류를 식별하는 반면, IAM Access Analyzer는 정책을 더욱 구체화하는 데 도움이 되는 권장 사항과 함께 추가 정책 검사를 제공합니다. 정책 검증에 대한 자세한 내용은 IAM 정책 검증 섹션을 참조하세요. IAM Access Analyzer 정책 확인 및 실행 가능한 권장 사항에 대한 자세한 내용은 IAM Access Analyzer 정책 검증을 참조하세요.

JSON 정책 문서 구조

다음 그림처럼 JSON 정책 문서는 이러한 요소를 포함합니다.

- 문서 상단에 위치하는 정책 전반의 선택적 정보
- 하나 이상의 개별 문

각 설명문에는 단일 권한에 대한 정보가 포함되어 있습니다. 정책에 설명문이 여러 개 포함되어 있는 경우, AWS는 설명문을 평가하는 동안 전체에 대해 논리 OR을 적용합니다. 요청 하나에 적용되는 정책이 여럿인 경우, AWS는 정책을 평가하는 동안 전체에 걸쳐 논리 OR을 적용합니다.



문의 정보는 일련의 요소 안에 포함되어 있습니다.

- Version - 사용하고자 하는 정책 언어의 버전을 지정합니다. 최신 `2012-10-17` 버전을 사용하는 것이 좋습니다. 자세한 내용은 IAM JSON 정책 요소: Version 단원을 참조하세요.
- Statement - 이 주요 정책 요소를 다음 요소의 컨테이너로 사용합니다. 정책에 설명문 둘 이상을 포함할 수 있습니다.
- Sid(선택 사항) - 선택 설명문 ID를 포함하여 설명문들을 구분합니다.
- Effect - `Allow` 또는 `Deny`를 사용하여 정책에서 액세스를 허용하는지 또는 거부하는지 여부를 설명합니다.
- Principal(일부 상황에서만 필요) - 리소스 기반 정책을 생성하는 경우 액세스를 허용하거나 거부할 계정, 사용자, 역할 또는 페더레이션 사용자를 표시해야 합니다. 사용자 또는 역할에 연결할 IAM 권한 정책을 생성하면 이 요소를 포함할 수 없습니다. 보안 주체는 사용자 또는 역할을 의미합니다.
- Action - 정책이 허용하거나 거부하는 작업 목록을 포함합니다.
- Resource(일부 상황에서만 필요) - IAM 권한 정책을 생성하는 경우 작업이 적용되는 리소스 목록을 지정해야 합니다. 리소스 기반 정책을 생성하는 경우 이 요소는 선택 사항입니다. 이 요소를 포함하지 않으면 작업이 적용되는 리소스는 정책이 연결된 리소스입니다.
- Condition(선택 사항) - 정책에서 권한을 부여하는 상황을 지정합니다.

이러한 요소와 기타 더 고급 정책 요소에 대한 자세한 정보는 IAM JSON 정책 요소 참조 섹션을 참조하세요.

복수의 문 및 복수의 정책

엔터티(사용자 또는 역할)에 부여할 권한을 하나 이상 정의하고자 할 경우, 단일 정책에 여러 설명문을 사용할 수 있습니다. 여러 정책을 연결할 수도 있습니다. 단일한 설명문에 여러 권한을 정의하고자 할 경우, 정책이 기대하는 액세스를 보장하지 않을 수 있습니다. 리소스 유형에 따라 정책을 나누는 것이 좋습니다.

정책의 제한된 크기로 인해 더 복잡한 권한에 대해서는 여러 정책을 사용해야 할 수도 있습니다. 개별 사용자 관리형 정책에 권한의 기능적 그룹화를 만드는 방법이 좋습니다. 예를 들어, IAM 사용자 관리용 정책 하나, 자기 관리용 하나 및 S3 버킷 관리용 기타 정책 하나를 생성합니다. 여러 설명문과 여러 정책의 조합과 상관없이 AWS는 동일한 방식으로 정책을 평가합니다.

예를 들어, 다음 정책에는 설명문이 세 개 있으며 각 설명문은 단일 계정에 별도의 권한 세트를 부여합니다. 설명문은 다음을 정의합니다.

- Sid(설명문 ID)의 FirstStatement 첫 번째 설명문은 연결된 정책으로 사용자가 자체 암호를 변경하도록 허용합니다. 이 문에서 Resource 요소는 “*”(“모든 리소스”를 의미)이지만 실제로 ChangePassword API 작업(또는 동등한 change-password CLI 명령)은 요청을 수행한 사용자의 암호에만 영향을 미칩니다.
- 두 번째 문은 사용자가 자신의 AWS 계정에 있는 모든 Amazon S3 버킷을 나열할 수 있도록 합니다. 이 문에서 Resource 요소는 "*"("모든 리소스"를 의미)이지만 정책에서 다른 계정의 리소스에 대한 액세스 권한을 부여하지 않으므로 사용자는 자신의 AWS 계정에 있는 버킷만 나열할 수 있습니다.
- 세 번째 설명문은 사용자가 confidential-data라는 버킷에 있는 객체를 나열 및 검색할 수 있도록 하지만, 이는 사용자가 멀티 팩터 인증(MFA)에서 인증한 경우에 한합니다. 정책의 Condition 요소는 MFA 인증을 수행합니다.

  정책 문에 Condition 요소가 포함된 경우, Condition 요소가 true로 평가된 경우에만 해당 문이 유효합니다. 이때 Condition은 사용자가 MFA 인증된 경우 true로 평가됩니다. 사용자가 MFA 인증되지 않은 경우, 이 Condition은 false로 평가됩니다. 이 경우 이 정책의 세 번째 설명문과 사용자는 confidential-data 버킷을 적용하지 않고 이에 액세스할 수 없습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FirstStatement",
      "Effect": "Allow",
```

```
      "Action": ["iam:ChangePassword"],
      "Resource": "*"
    },
    {
      "Sid": "SecondStatement",
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Sid": "ThirdStatement",
      "Effect": "Allow",
      "Action": [
        "s3:List*",
        "s3:Get*"
      ],
      "Resource": [
        "arn:aws:s3:::confidential-data",
        "arn:aws:s3:::confidential-data/*"
      ],
      "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
    }
  ]
}
```

JSON 정책 구문 예제

다음 자격 증명 기반 정책은 `example_bucket`이라는 하나의 Amazon S3 버킷 목록에 암시된 보안 주체를 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
}
```

다음 리소스 기반 정책은 Amazon S3 버킷에 연결될 수 있습니다. 이 정책에서는 특정 AWS 계정 구성원이 `mybucket`라는 버킷의 모든 Amazon S3 작업을 수행할 수 있도록 합니다. 작업 내 버킷 또는 객

체에 수행될 수 있는 모든 작업을 허용합니다. (이 정책은 계정에만 신뢰를 부여하므로, 해당 계정의 개별 사용자는 지정된 Amazon S3 작업에 대한 권한을 다시 부여받아야 합니다.)

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "1",
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::account-id:root"]},
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::mybucket",
      "arn:aws:s3:::mybucket/*"
    ]
  }]
}
```

공통 시나리오가 포함되는 예제 정책은 IAM 자격 증명 기반 정책의 예 섹션을 참조하세요.

최소 권한 부여

IAM 정책을 생성할 때는 최소 권한 부여의 표준 보안 조언을 따르거나, 작업 수행에 필요한 최소한의 권한만 부여합니다. 사용자(역할)가 수행해야 하는 작업을 파악한 후 사용자들이 해당 작업만 수행하도록 사용자에 대한 정책을 작성합니다.

최소한의 권한 조합으로 시작하여 필요에 따라 추가 권한을 부여합니다. 처음부터 권한을 많이 부여한 후 나중에 줄이는 방법보다 이 방법이 안전합니다.

최소 권한의 대안으로 AWS 관리형 정책 또는 와일드카드(*)를 사용하는 정책 권한을 사용하여 정책을 시작할 수 있습니다. 보안 주체가 작업을 수행하는 데 필요한 것보다 더 많은 권한을 부여할 경우 보안 위험을 고려하세요. 이러한 보안 주체를 모니터링하여 사용 중인 권한을 확인합니다. 그런 다음 최소 권한 정책을 작성합니다.

IAM에서는 부여하는 권한을 구체화하는 데 도움이 되는 몇 가지 옵션을 제공합니다.

- 액세스 수준 그룹화 이해 - 액세스 수준 그룹화를 사용하면 정책이 부여하는 액세스 수준을 이해할 수 있습니다. 정책 작업은 `List`, `Read`, `Write`, `Permissions management` 또는 `Tagging`으로 분류됩니다. 예를 들어 `List` 및 `Read` 액세스 레벨에서 작업을 선택하여 사용자에게 읽기 전용 액세스 권한을 부여할 수 있습니다. 정책 요약을 사용하여 액세스 레벨 권한을 이해하는 방법에 대해 알아보려면 정책 요약의 액세스 레벨 이해하기 섹션을 참조하세요.

- 정책 검증 - JSON 정책을 생성 및 편집할 때 IAM Access Analyzer를 사용하여 정책 검증을 수행할 수 있습니다. 기존 정책을 모두 검토하고 검증하는 것이 좋습니다. IAM Access Analyzer는 정책 검증을 위해 100개 이상의 정책 확인 항목을 제공합니다. 정책의 문이 지나치게 허용적이라고 생각되는 액세스를 허용하는 경우 보안 경고를 생성합니다. 최소 권한을 부여하기 위해 작업할 때 보안 경고를 통해 제공되는 실행 가능한 권장 사항을 사용할 수 있습니다. IAM Access Analyzer가 제공하는 정책 확인 사항에 대한 자세한 내용은 IAM Access Analyzer 정책 검증을 참조하세요.
- 액세스 활동에 기반한 정책 생성 - 부여하는 권한을 세분화할 수 있도록 IAM 엔터티(사용자 또는 역할)의 액세스 활동을 기반으로 IAM 정책을 생성할 수 있습니다. IAM Access Analyzer는 사용자의 AWS CloudTrail 로그를 검토하고 지정된 기간에 역할에 의해 사용된 권한이 포함된 정책 템플릿을 생성합니다. 이 템플릿을 사용하여 세분화된 권한이 포함된 관리형 정책을 생성한 다음 IAM 엔터티에 연결할 수 있습니다. 이렇게 하면 특정 사용 사례에 사용자나 역할이 AWS 리소스와 상호 작용하는 데 필요한 권한만 부여됩니다. 자세한 내용은 액세스 활동을 기반으로 정책 생성 섹션을 참조하세요.
- 마지막으로 액세스한 정보 사용 - 최소 권한으로 도울 수 있는 또 다른 기능은 마지막으로 액세스한 정보입니다. IAM 사용자, 그룹, 역할 또는 정책에 대한 IAM 콘솔 세부 정보 페이지의 액세스 관리자(Access Advisor) 탭에서 이 정보를 확인합니다. 마지막으로 액세스한 정보에는 Amazon EC2, IAM, Lambda, Amazon S3와 같은 일부 서비스에 마지막으로 액세스하여 작업한 정보가 포함됩니다. AWS Organizations 관리 계정 자격 증명을 사용하여 로그인하는 경우 IAM 콘솔의 AWS Organizations 섹션에서 마지막으로 액세스한 서비스 정보를 볼 수 있습니다. 또한 AWS CLI 또는 AWS API를 사용하여 IAM 또는 조직의 엔터티 또는 정책에 대해 마지막으로 액세스한 정보 보고서를 검색할 수 있습니다. 이 정보를 사용하여 불필요한 권한을 확인할 수 있으므로 IAM 또는 Organizations 정책을 미세 조정함으로써 최소 권한의 원칙을 보다 잘 준수할 수 있습니다. 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 단원을 참조하십시오.
- AWS CloudTrail에서 계정 이벤트 검토 - 권한을 추가로 줄이려면 AWS CloudTrail 이벤트 기록의 계정 이벤트를 볼 수 있습니다. CloudTrail 이벤트 로그에는 정책의 권한을 줄이는 데 사용할 수 있는 자세한 이벤트 정보가 포함되어 있습니다. 로그에는 IAM 엔터티에 필요한 작업 및 리소스만 포함되어 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 콘솔에서 CloudTrail 이벤트 보기를 참조하세요.

자세한 내용은 서비스별 리소스에 대해 정책을 작성하는 방법의 예제를 제공하는 각 서비스의 다음 정책 주제를 참조하세요.

- Amazon DynamoDB 개발자 안내서의 Amazon DynamoDB에 대한 인증 및 액세스 제어
- Amazon Simple Storage Service 사용 설명서의 버킷 정책 및 사용자 정책 사용
- Amazon Simple Storage Service 사용 설명서의 액세스 제어 목록(ACL) 개요

관리형 정책과 인라인 정책

IAM에서 자격 증명에 대한 권한을 설정하는 경우 AWS 관리형 정책, 고객 관리형 정책 또는 인라인 정책 중 어느 것을 사용할지를 결정해야 합니다. 다음 주제에서는 각 자격 증명 기반 정책 유형과 사용 시기에 대한 자세한 정보를 제공합니다.

주제

AWS 관리형 정책

AWS 관리형 정책은 AWS에서 생성 및 관리하는 독립적인 정책입니다. 여기에서 독립형 정책은 정책 스스로 정책 이름이 포함된 Amazon 리소스 이름(ARN)을 갖고 있다는 것을 의미합니다. 예를 들어 `arn:aws:iam::aws:policy/IAMReadOnlyAccess`는 AWS 관리형 정책입니다. ARN에 대한 자세한 내용은 IAM ARN 섹션을 참조하세요. AWS에 대한 AWS 서비스 관리형 정책의 목록은 AWS 관리형 정책을 참조하세요.

AWS 관리형 정책을 사용하면 사용자, 사용자 그룹 및 역할에 적절한 권한을 손쉽게 할당할 수 있습니다. 정책을 직접 작성하는 것보다 빠르고, 여러 가지 일반 사용 사례에 맞는 권한이 포함되어 있습니다.

AWS 관리형 정책에 정의되어 있는 권한은 변경할 수 없습니다. AWS가 AWS 관리형 정책에서 정의한 권한을 간혹 업데이트합니다. AWS에서 업데이트할 경우 정책이 추가되어 있는 모든 보안 주체 엔터티(사용자, 사용자 그룹 및 역할)에게도 업데이트가 적용됩니다. 새로운 AWS 제품을 실행하거나 새로운 API 호출을 기존 서비스에 이용하는 경우 AWS가 AWS 관리형 정책을 업데이트할 가능성이 높습니다. 예를 들어 ReadOnlyAccess라는 이름의 AWS 관리형 정책은 모든 AWS 서비스 및 리소스에 대한 읽기 전용 액세스 권한을 제공합니다. AWS에서 새로운 서비스가 실행될 때는 AWS가 ReadOnlyAccess 정책을 업데이트하여 새로운 서비스에 대한 읽기 전용 권한을 추가합니다. 이렇게 업데이트된 권한은 정책이 추가되는 모든 보안 주체 엔터티에게 적용됩니다.

전체 액세스 AWS 관리형 정책: 서비스에 대한 전체 액세스 권한을 부여하여 서비스 관리자에 대한 권한을 정의합니다. 그러한 예는 다음과 같습니다.

- AmazonDynamoDBFullAccess
- IAMFullAccess

파워 유저 AWS 관리형 정책: AWS 서비스 및 리소스에 대한 모든 액세스 권한을 제공하지만 사용자 및 사용자 그룹을 관리하도록 허용하지는 않습니다. 그러한 예는 다음과 같습니다.

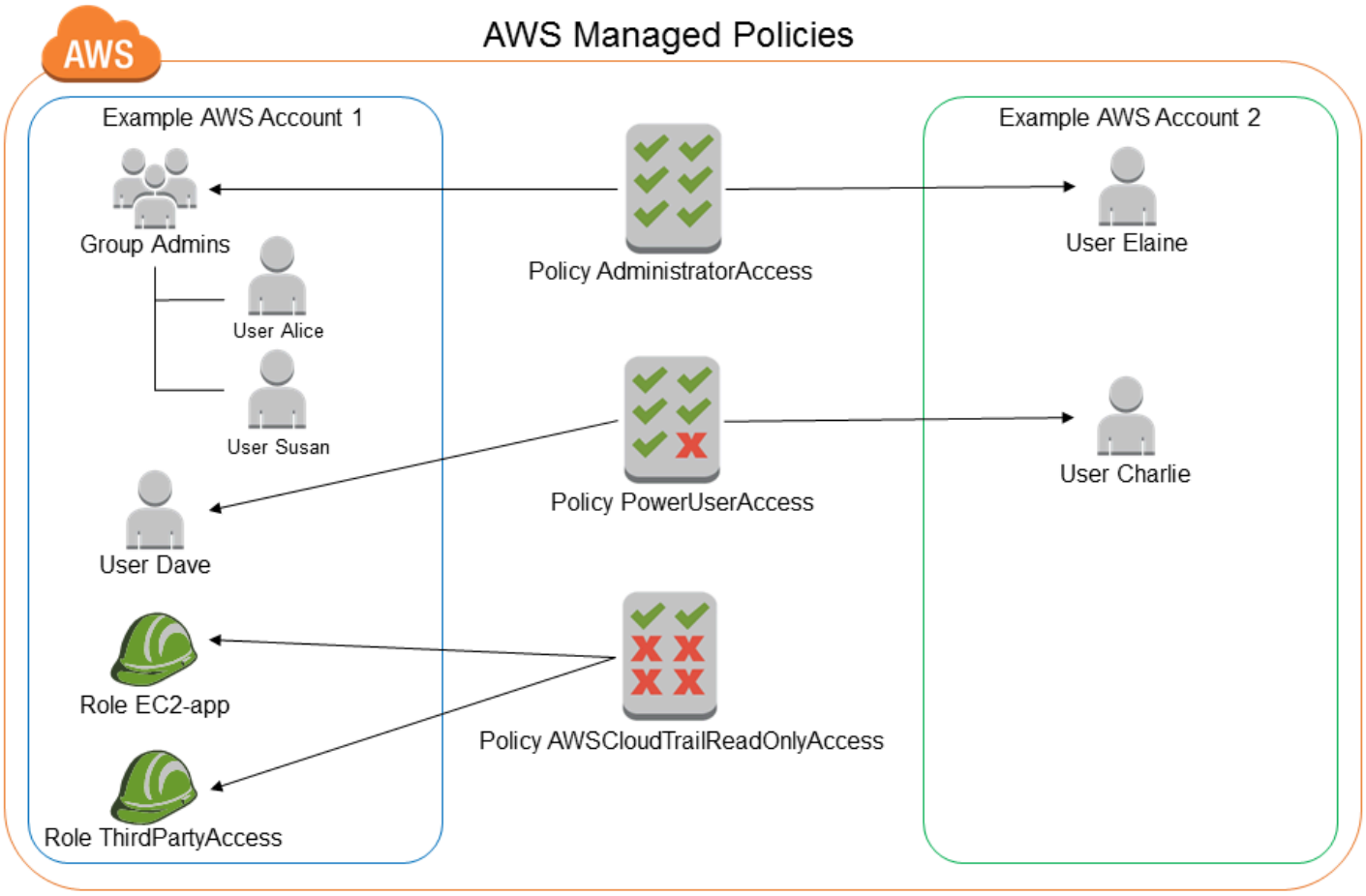
- AWSCodeCommitPowerUser
- AWSKeyManagementServicePowerUser

부분 액세스 AWS 관리형 정책: 권한 관리 액세스 수준 권한을 허용하지 않고 AWS 서비스에 대한 특정 수준의 액세스를 제공합니다. 그러한 예는 다음과 같습니다.

- AmazonMobileAnalyticsWriteOnlyAccess
- AmazonEC2ReadOnlyAccess

직능 AWS 관리형 정책: 이러한 정책은 IT 업계에서 일반적으로 사용되는 직능과 밀접한 관련이 있으며, 그러한 직능에 필요한 권한을 손쉽게 부여하도록 해줍니다. 직무 정책을 사용하는 큰 장점 중 하나는 새로운 서비스와 API 작업이 도입될 때마다 AWS가 이를 유지하고 업데이트할 수 있다는 점입니다. 예를 들어 AdministratorAccess 직무는 AWS의 모든 서비스 및 리소스에 대한 모든 액세스 권한 및 작업 권한을 위임합니다. 계정 관리자에게만 이 정책을 사용하는 것이 좋습니다. IAM 및 조직에 대해서는 제한적인 액세스 권한만 있으면 되지만 그 밖의 모든 서비스에 대해 모든 액세스 권한이 필요한 고급 사용자의 경우, PowerUserAccess 직무를 사용하세요. 직무 정책의 목록과 설명은 직무에 관한 AWS 관리형 정책 섹션을 참조하세요.

다음은 AWS 관리형 정책을 나타낸 다이어그램입니다. 다이어그램을 보면 AdministratorAccess, PowerUserAccess, 그리고 AWSCloudTrailReadOnlyAccess 등 3개의 AWS 관리형 정책이 있습니다. 다이어그램에도 나와 있지만 단일 AWS 관리형 정책을 다른 AWS 계정의 보안 주체 엔터티에 추가할 수도 있고, 단일 AWS 계정의 다른 보안 주체 엔터티에 추가할 수도 있습니다.



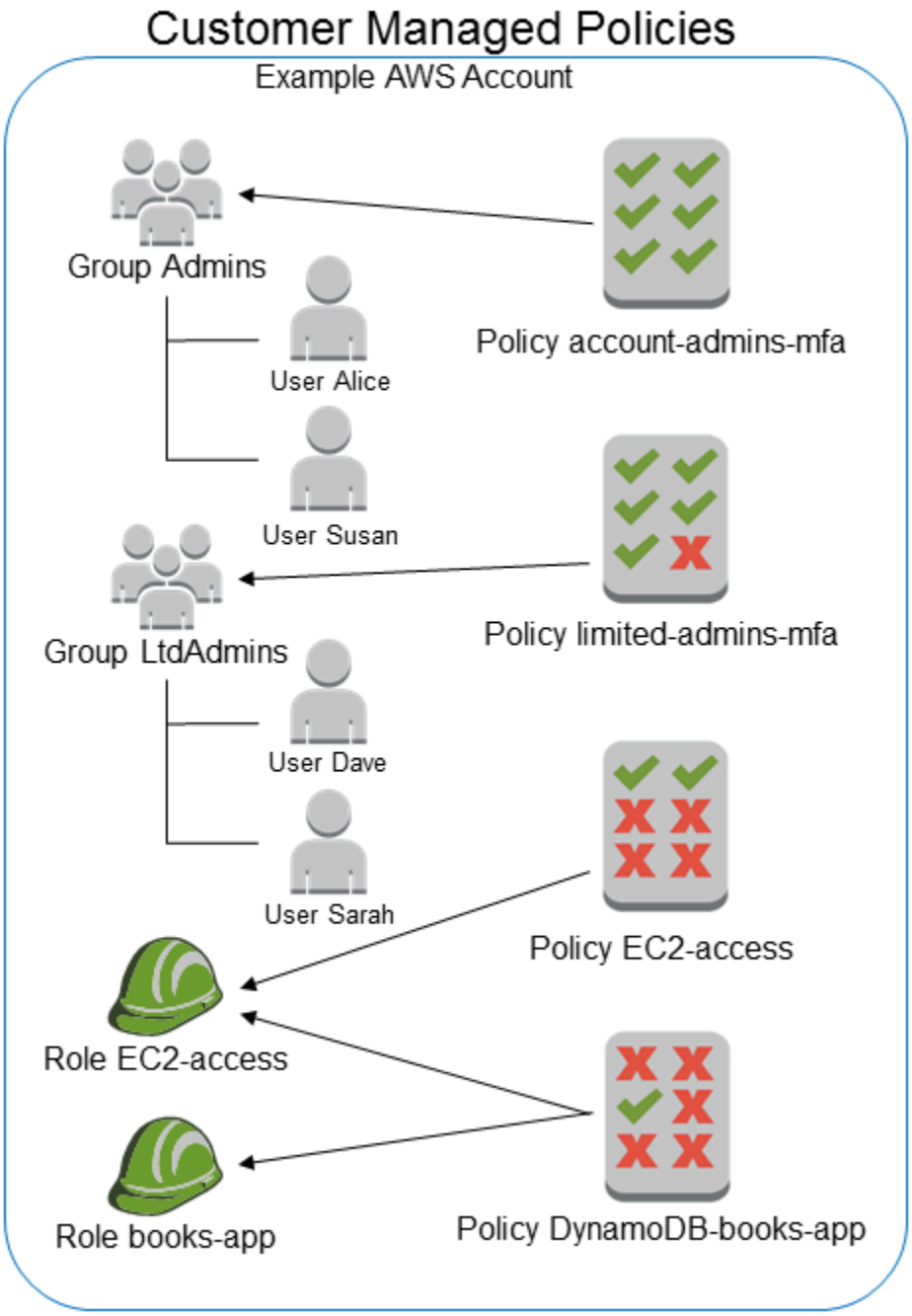
고객 관리형 정책

자체 AWS 계정에서 보안 주체 엔터티(사용자, 사용자 그룹 및 역할)에 연결할 수 있는 독립형 정책을 생성할 수 있습니다. 특정 사용 사례에 맞게 이러한 고객 관리형 정책을 생성하고, 원하는 만큼 자주 변경 및 업데이트할 수 있습니다. AWS 관리형 정책과 마찬가지로, 정책을 보안 주체 엔터티에 추가할 경우 정책에서 정의한 권한까지 엔터티에게 부여하게 됩니다. 정책의 권한을 업데이트할 경우, 정책이 추가되어 있는 모든 보안 주체 엔터티에 변경 사항이 적용됩니다.

고객이 관리하는 정책을 생성하는 좋은 방법은 AWS에서 관리하는 기존의 정책을 복사하여 시작하는 것입니다. 이렇게 하면 시작 시 올바른 정책으로 시작하므로 해당 환경에 맞게 사용자 지정만 하면 됩니다.

다음은 고객 관리형 정책을 나타낸 다이어그램입니다. 각 정책은 자체적으로 정책 이름이 포함된 Amazon 리소스 이름(ARN)을 갖고 있는 IAM 엔터티입니다. 다이어그램을 보면 동일한 정책을 여러 보안 주체 엔터티에 추가할 수 있습니다. 예를 들어 동일한 DynamoDB-books-app 정책이 두 개의 다른 IAM 역할에 추가됩니다.

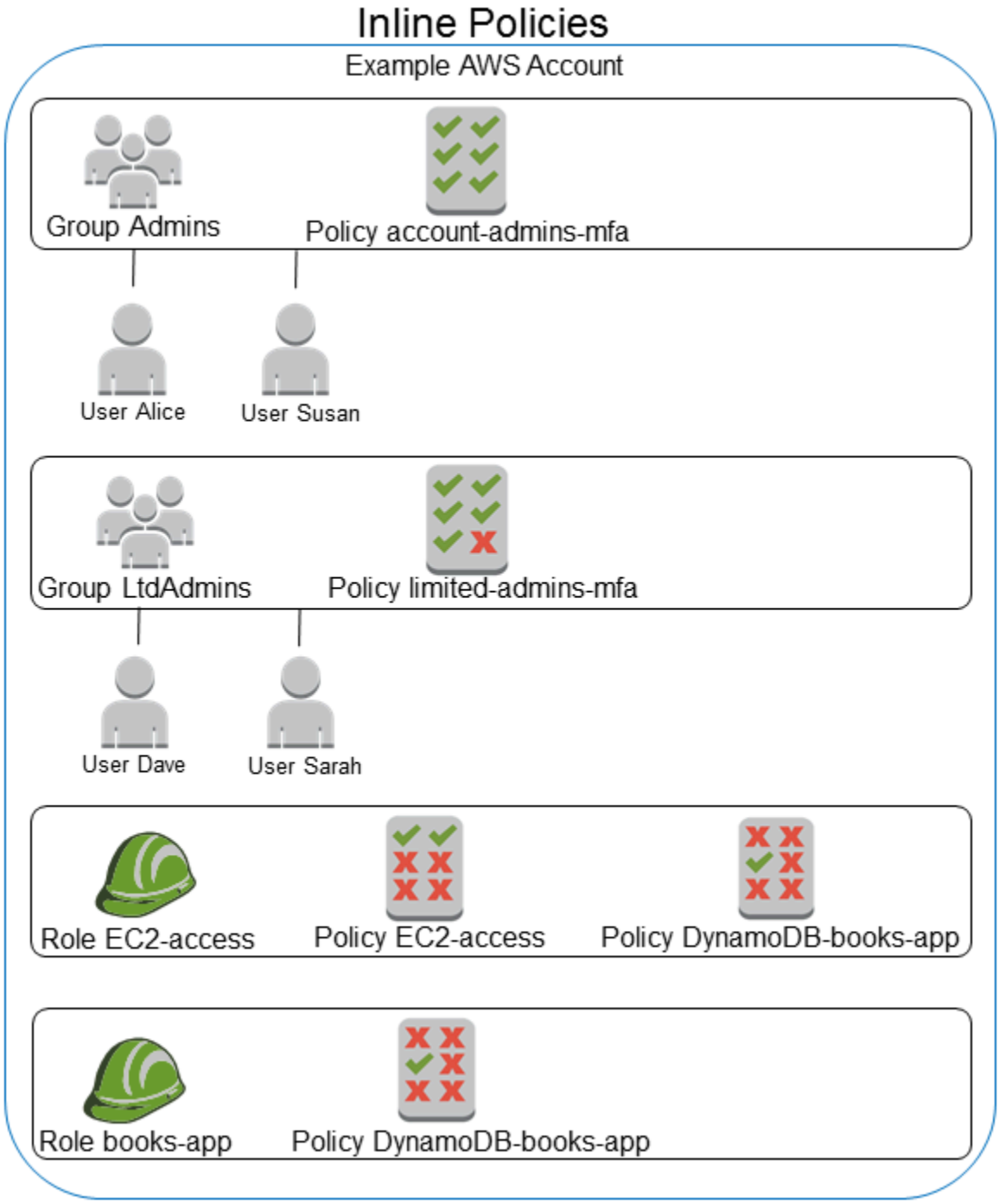
자세한 내용은 고객 관리형 정책으로 사용자 지정 IAM 권한 정의 단원을 참조하세요.



인라인 정책

인라인 정책은 단일 IAM ID(사용자, 사용자 그룹 또는 역할)에 대해 생성되는 정책입니다. 인라인 정책은 정책과 자격 증명을 정확히 1대 1 관계로 유지합니다. 이는 자격 증명을 삭제하면 삭제됩니다. 자격 증명을 생성하거나 이후에 생성할 때 정책을 생성하여 자격 증명에 삽입할 수 있습니다. 정책이 둘 이상의 엔터티에 적용될 수 있는 경우 관리형 정책을 사용하는 것이 좋습니다.

다음은 인라인 정책을 나타낸 다이어그램입니다. 각 정책은 사용자, 그룹 또는 역할에서 내재된 부분입니다. 다이어그램을 보면 2개의 역할에 동일한 정책(the DynamoDB-books-app 정책)이 추가되어 있지만, 단 하나의 정책도 공유하지 않습니다. 각 역할에는 고유한 정책 사본이 있습니다.



관리형 정책과 인라인 정책의 선택

관리형 정책과 인라인 정책 중 하나를 결정할 때는 사용 사례를 고려합니다. 대부분 경우 인라인 정책보다는 관리형 정책의 사용을 권장합니다.

> **Note**
>
> 관리형 정책과 인라인 정책을 함께 사용하여 보안 주체 엔터티에 대한 공통 및 고유 권한을 정의할 수 있습니다.

관리형 정책은 다음과 같은 기능을 제공합니다.

재사용성

단일 관리형 정책은 다수의 보안 주체 개체(사용자, 그룹 및 역할)에 추가할 수 있습니다. 정책 라이브러리를 생성하여 AWS 계정에 유용한 권한을 정의한 다음 필요에 따라 생성한 정책을 보안 주체 엔터티에 추가할 수 있습니다.

중앙 변경 관리

관리형 정책 변경 시 정책이 추가되어 있는 모든 보안 주체 엔터티에 변경 사항이 적용됩니다. 예를 들어 AWS API 권한을 추가할 경우 고객 관리형 정책을 업데이트하거나 AWS 관리형 정책을 연결하여 권한을 추가할 수 있습니다. AWS 관리형 정책을 사용하는 경우 AWS가 정책을 업데이트합니다. 관리형 정책이 업데이트되면 관리형 정책이 추가되어 있는 모든 보안 주체 엔터티에 변경 사항이 적용됩니다. 이와는 대조적으로 인라인 정책을 변경하려면 인라인 정책이 추가되어 있는 자격 증명을 일일이 편집해야 합니다. 예를 들어 그룹과 역할에 모두 동일한 인라인 정책이 추가되어 있더라도 정책을 변경하기 위해서는 두 보안 주체 개체를 개별적으로 편집해야만 합니다.

버전 관리 및 롤백

고객 관리 정책을 변경할 경우 변경된 정책은 기존 정책을 덮어쓰지 않습니다. 대신 IAM에서 관리형 정책의 새 버전을 생성합니다. IAM은 고객 관리형 정책을 최대 5개 버전까지 저장합니다. 정책 버전은 필요에 따라 정책을 이전 버전으로 되돌리는 데도 사용됩니다.

> **Note**
>
> 정책 버전은 `Version` 정책 요소와 다릅니다. `Version` 정책 요소는 정책 내에서 사용되며 정책 언어의 버전을 정의합니다. 정책 버전에 대한 자세한 정보는 the section called “IAM

정책 버전 관리" 섹션을 참조하세요. `Version` 정책 요소에 대한 자세한 정보는 IAM JSON 정책 요소: Version을 참조하세요.

권한 위임 관리

정책으로 정의한 권한을 지속적으로 제어하면서 AWS 계정에 속한 사용자가 정책을 추가 및 분리하도록 허용할 수 있습니다. 이렇게 하려면 일부 사용자에게는 전체 관리자 권한을 위임합니다. 다시 말해, 전체 관리자란 정책을 생성, 업데이트 및 삭제할 수 있는 것을 말합니다. 제한된 관리자로서 다른 사용자를 지정할 수 있습니다. 그러한 제한된 관리자는 다른 보안 주체 엔터티에 정책을 연결할 수 있지만 이때 정책은 연결하도록 허용된 정책으로 제한됩니다.

권한 위임 관리에 대한 자세한 내용은 정책에 대한 액세스 제어 섹션을 참조하세요.

더 큰 정책 문자 제한

관리형 정책의 최대 문자 크기 제한이 인라인 정책의 문자 제한보다 큽니다. 인라인 정책의 문자 크기 제한에 도달하면 더 많은 IAM 그룹을 생성하고 관리형 정책을 그룹에 연결할 수 있습니다.

할당량과 제한에 대한 자세한 내용은 IAM 및 AWS STS 할당량 섹션을 참조하세요.

AWS 관리형 정책의 자동 업데이트

AWS는 AWS 관리형 정책을 유지하면서 필요에 따라 자동으로 업데이트하기 때문에(예를 들어 새로운 AWS 서비스 권한을 추가하기 위해) 직접 변경할 필요가 없습니다. 업데이트는 AWS 관리형 정책을 추가한 보안 주체 엔터티에게 자동으로 적용됩니다.

인라인 정책 사용

인라인 정책은 정책과 정책이 추가된 자격 증명을 정확히 1대 1 관계로 유지할 때 유용합니다. 정책 권한을 의도하지 않은 자격 증명에 실수로 할당하는 일을 배제하려고 하는 경우를 예로 들어 보겠습니다. 이때 인라인 정책을 사용하면 정책 권한이 잘못된 자격 증명에 실수로 추가되는 일이 사라집니다. 그 밖에도 AWS Management Console을 사용하여 자격 증명을 삭제할 경우 자격 증명에 삽입된 정책 역시 삭제됩니다. 해당 정책은 보안 주체 엔터티의 일부이기 때문입니다.

관리형 정책 시작하기

최소 권한 부여 또는 작업 수행에 요구되는 권한만 부여하는 정책을 사용하는 것이 좋습니다. 최소 권한을 부여하는 가장 안전한 방법은 팀에 필요한 권한만 있는 고객 관리형 정책을 작성하는 것입니다. 필요한 경우 팀에서 추가 권한을 요청할 수 있는 프로세스를 만들어야 합니다. 팀이 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하는 데는 시간과 전문 지식이 필요합니다.

IAM 자격 증명(사용자, 사용자 그룹 및 역할)에 대한 권한을 추가하려면 AWS 관리형 정책을 사용할 수 있습니다. AWS 관리형 정책은 최소 권한을 부여하지 않습니다. 보안 주체가 작업을 수행하는 데 필요한 것보다 더 많은 권한을 부여할 경우 보안 위험을 고려해야 합니다.

AWS 관리형 정책(작업 기능 포함)을 모든 IAM 자격 증명에 연결할 수 있습니다. 자세한 내용은 IAM 자격 증명 권한 추가 및 제거 섹션을 참조하세요.

최소 권한으로 전환하려면 AWS Identity and Access Management Access Analyzer를 실행하여 AWS 관리형 정책으로 보안 주체를 모니터링합니다. 사용 권한을 학습한 후 팀에 필요한 권한만 있는 고객 관리형 정책을 작성하거나 생성할 수 있습니다. 이는 덜 안전하지만 팀에서 AWS를 어떻게 사용하는지 학습할 수록 더 많은 유연성을 제공합니다.. 자세한 내용은 IAM Access Analyzer 정책 생성 섹션을 참조하세요.

AWS 관리형 정책은 여러 가지 일반 사용 사례에서 권한을 제공할 목적으로 설계되었습니다. 특정 작업 기능을 위해 설계된 AWS 관리형 정책에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 섹션을 참조하세요.

AWS 관리형 정책의 목록은 AWS 관리형 정책 참조 가이드를 참조하세요.

인라인 정책을 관리형 정책으로 변환하기

계정에 인라인 정책이 있는 경우 이를 관리형 정책으로 변환할 수 있습니다. 이렇게 하려면 정책을 새 관리형 정책에 복사합니다. 그런 다음 인라인 정책이 있는 자격 증명에 새 정책을 연결합니다. 그런 다음 인라인 정책을 삭제합니다.

인라인 정책을 관리형 정책으로 변환하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자 그룹, 사용자 또는 역할을 선택합니다.
3. 목록에서 제거할 정책이 있는 사용자 그룹, 사용자 또는 역할 이름을 선택합니다.
4. 권한(Permissions) 탭을 선택합니다.
5. 사용자 그룹의 경우 제거할 인라인 정책의 이름을 선택합니다. 사용자 및 역할에 대해 필요한 경우 ***n***개 더 표시를 선택한 다음 제거할 인라인 정책 옆에 있는 화살표를 선택합니다.
6. 정책에 대한 JSON 정책 문서를 복사합니다.
7. 탐색 창에서 Policies(정책)을 선택합니다.
8. 정책 생성을 선택한 후 JSON 옵션을 선택합니다.

9. 기존 텍스트를 JSON 정책 텍스트로 바꾸고 다음을 선택합니다.
10. 정책의 이름과 설명(선택 사항)을 입력한 다음에 정책 생성을 선택합니다.
11. 탐색 창에서 사용자 그룹, 사용자 또는 역할을 선택한 다음 제거하려는 정책이 있는 사용자 그룹, 사용자 또는 역할의 이름을 다시 선택합니다.
12. 권한 탭을 선택한 다음 권한 추가를 선택합니다.
13. 사용자 그룹에 대해 새 정책 이름 옆의 확인란을 선택한 다음 권한 추가와 정책 연결을 차례로 선택합니다. 사용자 또는 역할의 경우 권한 추가를 선택합니다. 다음 페이지에서 기존 정책 직접 연결을 선택하고 새 정책 이름 옆의 확인란을 선택한 다음, 다음을 선택하고 권한 추가를 선택합니다.

    사용자 그룹, 사용자 또는 역할에 대한 요약 페이지로 돌아갑니다.
14. 제거할 인라인 정책 옆의 확인란을 선택한 다음 제거를 선택합니다.

사용되지 않는 AWS 관리형 정책

사용 권한 할당을 단순화하기 위해 AWS가 제공하는 관리형 정책 - IAM 사용자, 그룹 및 역할에 연결하도록 준비된 사전 정의된 정책입니다.

새로운 서비스가 나왔을 때와 같이 AWS는 때때로 기존 정책에 새 권한을 추가해야 합니다. 기존 정책에 새 권한을 추가해도 특성이나 권한이 제거되거나 방해를 받지는 않습니다.

하지만 AWS는 필요한 변경이 기존 정책에 적용될 경우 고객에게 영향을 줄 수 있기 때문에 새로 정책을 만듭니다. 예를 들어 기존 정책에서 권한을 제거하면 이 정책을 사용하는 IAM 주체나 애플리케이션의 권한이 손상되어 중요한 작업에 방해가 될 수 있습니다.

따라서 이러한 변경이 필요할 경우 AWS는 해당 사항을 변경한 정책을 새로 만들어서 고객에게 제공합니다. 기존 정책은 사용되지 않음으로 표시됩니다. 사용 중단된 관리형 정책은 IAM 콘솔의 정책 목록에서 해당 정책 옆에 경고 아이콘이 표시됩니다.

사용되지 않는 정책은 다음과 같은 특성을 갖습니다.

- 현재 연결된 모든 사용자, 그룹 및 역할에 계속 적용됩니다. 연결이 해제되지 않습니다.
- 새로운 사용자, 그룹 또는 역할에 연결할 수 없습니다. 현재 주체에서 연결을 해제할 경우 다시 연결할 수 없습니다.
- 현재의 모든 주체로부터 연결을 해제하면 더 이상 표시되지 않으며 어떤 경우에도 다시 사용할 수 없습니다.

사용자, 그룹 또는 역할에 정책이 필요할 경우 새로운 정책을 연결해야 합니다. 정책이 사용되지 않음으로 설정되었다고 알림을 받으면 모든 사용자, 그룹 및 역할을 대체 정책에 연결하고 사용되지 않는 정책으로부터 연결을 해제하는 것이 좋습니다. 사용되지 않는 정책을 계속 사용하면 위험이 수반될 수 있으므로 대체 정책으로 전환하는 것이 좋습니다.

데이터 경계를 사용하여 권한 가드레일 설정

데이터 경계 가드레일은 광범위한 AWS 계정 및 리소스 전반에서 데이터를 보호하는 데 도움이 되는 상시 경계 역할을 하는 것입니다. 데이터 경계는 IAM 보안 모범 사례에 따라 여러 계정에 권한 가드레일을 설정합니다. 이러한 조직 차원의 권한 가드레일은 기존의 세분화된 액세스 제어를 대체하지 않습니다. 대신 사용자, 역할, 리소스가 일련의 정의된 보안 표준을 준수하도록 보장함으로써 보안 전략을 개선하는 데 도움이 되는 대략적인 액세스 제어 역할을 합니다.

데이터 경계는 신뢰할 수 있는 ID만 예상 네트워크에서 신뢰할 수 있는 리소스에 액세스하도록 하는 AWS 환경의 권한 가드레일 세트입니다.

- 신뢰할 수 있는 ID: 사용자의 AWS 계정 보안 주체(IAM 역할 또는 사용자) 및 사용자를 대신하는 AWS 서비스입니다.
- 신뢰할 수 있는 리소스: 사용자의 AWS 계정 또는 사용자를 대신하는 AWS 서비스가 소유한 리소스입니다.
- 예상 네트워크: 온프레미스 데이터 센터 및 Virtual Private Cloud(VPC) 또는 사용자를 대신하는 AWS 서비스 네트워크입니다.

> **Note**
>
> 경우에 따라 신뢰할 수 있는 비즈니스 파트너의 액세스도 포함하도록 데이터 경계를 확장해야 할 수 있습니다. 신뢰할 수 있는 ID, 신뢰할 수 있는 리소스, 회사와 AWS 서비스 용도에 맞는 예상 네트워크에 대한 정의를 생성할 때 의도된 모든 데이터 액세스 패턴을 고려해야 합니다.

데이터 경계 제어는 정보 보안 및 위험 관리 프로그램 내의 다른 보안 제어와 마찬가지로 취급해야 합니다. 즉, 위협 분석을 수행하여 클라우드 환경 내의 잠재적 위험을 식별한 다음, 자체 위험 수용 기준에 따라 적절한 데이터 경계 제어를 선택하고 구현합니다. 데이터 경계 구현에 대한 반복적인 위험 기반 접근 방식에 더 나은 정보를 제공하려면 데이터 경계 제어를 통해 해결되는 보안 위험 및 위협 벡터뿐만 아니라 보안 우선 순위도 이해해야 합니다.

데이터 경계 제어

대략적인 데이터 경계 제어를 통해 정책 유형 및 조건 키의 다양한 조합을 구현하고 세 가지 데이터 경계에 걸쳐 여섯 가지 뚜렷한 보안 목표를 달성할 수 있습니다.

경계	제어 목표	사용	적용 대상	전역 조건 컨텍스트 키
자격 증명	신뢰할 수 있는 ID만 내 리소스에 액세스할 수 있습니다.	리소스 기반 정책	리소스	aws:PrincipalOrgID aws:PrincipalOrgPaths aws:PrincipalAccount aws:PrincipalIsAwsService
	네트워크에서는 신뢰할 수 있는 ID만 허용됩니다.	VPC 엔드포인트 정책	네트워크	
리소스	ID는 신뢰할 수 있는 리소스에만 액세스할 수 있습니다.	SCP	ID	aws:ResourceOrgID aws:ResourceOrgPaths aws:ResourceAccount
	네트워크에서는 신뢰할 수 있는 리소스에만 액세스할 수 있습니다.	VPC 엔드포인트 정책	네트워크	
네트워크	ID는 예상 네트워크에서만 리소스에 액세스할 수 있습니다.	SCP	ID	aws:SourceIp aws:SourceVpc aws:SourceVpce aws:ViaAWSService
	리소스는 예상 네트워크에서만 액	리소스 기반 정책	리소스	

경계	제어 목표	사용	적용 대상	전역 조건 컨텍스트 키
	세스할 수 있습니다.			aws:PrincipalIsAwsService

데이터 경계는 의도하지 않은 액세스 패턴을 방지하기 위해 데이터 주변에 확고한 경계를 생성하는 것이라고 생각할 수 있습니다. 데이터 경계는 의도하지 않은 광범위한 액세스를 방지할 수 있지만 여전히 세분화된 액세스 제어 결정을 내려야 합니다. 데이터 경계를 설정하더라도 최소 권한을 위한 여정의 일환으로 IAM Access Analyzer와 같은 도구를 사용하여 권한을 지속적으로 미세 조정해야 합니다.

ID 경계

ID 경계는 신뢰할 수 있는 ID만 리소스에 액세스하고 네트워크에서는 신뢰할 수 있는 ID만 허용하는 일련의 세밀한 예방적 액세스 제어입니다. 신뢰할 수 있는 ID에는 사용자의 AWS 계정 및 사용자를 대신하는 AWS 서비스의 보안 주체(역할 또는 사용자)가 포함됩니다. 다른 모든 ID는 신뢰할 수 없는 것으로 간주되며 명시적인 예외가 부여되지 않는 한 ID 경계에 의해 차단됩니다.

다음 전역 조건 키는 ID 경계 제어를 적용하는 데 도움이 됩니다. 리소스 기반 정책에서 이러한 키를 사용하여 리소스에 대한 액세스를 제한하거나 VPC 엔드포인트 정책에서 네트워크에 대한 액세스를 제한할 수 있습니다.

- aws:PrincipalOrgID-이 조건 키를 사용하여 요청하는 IAM 보안 주체가 AWS Organizations의 지정된 조직에 속하도록 할 수 있습니다.
- aws:PrincipalOrgPaths-이 조건 키를 사용하여 요청하는 IAM 사용자, IAM 역할, 페더레이션 사용자 또는 AWS 계정 루트 사용자가 AWS Organizations의 지정된 조직 단위(OU)에 속하도록 할 수 있습니다.
- aws:PrincipalAccount-이 조건 키를 사용하여 정책에서 지정한 보안 주체 계정만 리소스에 액세스하도록 할 수 있습니다.
- aws:PrincipalIsAWSService 및 aws:SourceOrgID(또는 aws:SourceOrgPaths 및 aws:SourceAccount)-이 조건 키를 사용하여 AWS 서비스 보안 주체가 리소스에 액세스할 때 지정된 조직, 조직 구성 단위 또는 AWS Organizations의 계정에 있는 리소스를 대신해서만 액세스하도록 할 수 있습니다.

자세한 내용은 AWS에 데이터 경계 설정: 신뢰할 수 있는 ID만 회사 데이터에 액세스하도록 허용을 참조하세요.

리소스 경계

리소스 경계는 사용자 ID가 신뢰할 수 있는 리소스에만 액세스하고 네트워크에서 신뢰할 수 있는 리소스에만 액세스할 수 있는 일련의 대략적인 예방적 액세스 제어입니다. 신뢰할 수 있는 리소스에 사용자의 AWS 계정 또는 사용자를 대신하는 AWS 서비스가 소유한 리소스가 포함됩니다.

다음 전역 조건 키는 리소스 경계 제어를 적용하는 데 도움이 됩니다. 서비스 제어 정책(SCP)에서 이러한 키를 사용하여 사용자의 ID로 액세스할 수 있는 리소스를 제한하거나 VPC 엔드포인트 정책에서 네트워크를 통해 액세스할 수 있는 리소스를 제한할 수 있습니다.

- aws:ResourceOrgID-이 조건 키를 사용하여 액세스 중인 리소스가 AWS Organizations의 지정된 조직에 속하도록 할 수 있습니다.
- aws:ResourceOrgPaths-이 조건 키를 사용하여 액세스 중인 리소스가 AWS Organizations의 지정된 조직 단위(OU)에 속하도록 할 수 있습니다.
- aws:ResourceAccount-이 조건 키를 사용하여 액세스 중인 리소스가 AWS Organizations의 지정된 계정에 속하도록 할 수 있습니다.

경우에 따라 AWS가 소유한 리소스, 즉 사용자의 조직에 속하지 않고 보안 주체 또는 사용자를 대신하는 AWS 서비스가 액세스하는 리소스에 대한 액세스를 허용해야 할 수도 있습니다. 이러한 시나리오에 대한 자세한 내용은 AWS에 데이터 경계 설정: 내 조직의 신뢰할 수 있는 리소스만 허용을 참조하세요.

네트워크 경계

네트워크 경계는 사용자 ID가 예상 네트워크에서만 리소스에 액세스하고 리소스는 예상 네트워크에서만 액세스할 수 있는 일련의 대략적인 예방적 액세스 제어입니다. 예상 네트워크에는 온프레미스 데이터 센터 및 Virtual Private Cloud(VPC) 그리고 사용자를 대신하는 AWS 서비스 네트워크가 포함됩니다.

다음 전역 조건 키는 네트워크 경계 제어를 적용하는 데 도움이 됩니다. 서비스 제어 정책(SCP)에서 이러한 키를 사용하여 ID가 통신할 수 있는 네트워크를 제한하거나 리소스 기반 정책에서 예상 네트워크에 대한 리소스 액세스를 제한할 수 있습니다.

- aws:SourceIp-이 조건 키를 사용하여 요청자의 IP 주소가 지정된 IP 범위 내에 있는지 확인할 수 있습니다.
- aws:SourceVpc-이 조건 키를 사용하여 요청이 통과하는 VPC 엔드포인트가 지정된 VPC에 속하는지 확인할 수 있습니다.
- aws:SourceVpce-이 조건 키를 사용하여 요청이 지정된 VPC 엔드포인트를 통과하도록 할 수 있습니다.

- aws:ViaAWSService-이 조건 키를 사용하여 AWS 서비스가 전달 액세스 세션(FAS)를 통해 주체를 대신하여 요청을 보내도록 할 수 있습니다.
- aws:PrincipalIsAWSService-이 조건 키를 사용하여 AWS 서비스가 AWS 서비스 보안 주체를 통해 리소스에 액세스하도록 할 수 있습니다.

네트워크 외부에서 해당 리소스에 액세스할 수 있도록 AWS 서비스에 액세스 권한을 부여해야 하는 추가 시나리오가 있습니다. 자세한 내용은 AWS에 데이터 경계 설정: 예상 네트워크에서만 회사 데이터 액세스 허용을 참조하세요.

데이터 경계를 자세히 알아보기 위한 리소스

다음 리소스는 AWS 전반의 데이터 경계를 자세히 알아보는 데 도움이 됩니다.

- AWS의 데이터 경계-데이터 경계와 그 이점 및 사용 사례에 대해 알아봅니다.
- 백서: AWS에 데이터 경계 구축-이 백서에서는 AWS에서 ID, 리소스, 네트워크를 중심으로 경계를 만들기 위한 모범 사례와 사용 가능한 서비스를 간략하게 설명합니다.
- 웨비나: AWS에 데이터 경계 구축-다양한 위험 시나리오를기반으로 데이터 경계 제어를 구현하는 위치와 방법을 알아봅니다.
- 블로그 게시물 시리즈: AWS에 데이터 경계 설정-이 블로그 게시물에서는 주요 보안 및 구현 고려 사항을 포함하여 대규모로 데이터 경계를 설정하는 방법에 대한 규범적 지침을 다룹니다.
- 데이터 경계 정책 예제-이 GitHub 리포지토리에는 AWS에서 데이터 경계를 구현하는 데 도움이 되는 몇 가지 일반적인 패턴을 다루는 예제 정책이 포함되어 있습니다.
- 데이터 경계 도우미-이 도구를 사용하면 AWS CloudTrail 로그의 액세스 활동을 분석하여 데이터 경계 제어의 영향을 설계하고 예측할 수 있습니다.

IAM 엔터티의 권한 범위

에서는 IAM 엔터티(사용자 또는 역할)에 대한 권한 경계를 AWS지원합니다. 권한 경계는 관리형 정책을 사용하여 자격 증명 기반 정책을 통해 IAM 엔터티에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 엔터티의 권한 경계는 자격 증명 기반 정책 및 관련 권한 경계 모두에서 허용되는 작업만 수행하도록 허용합니다.

정책 유형에 대한 자세한 정보는 정책 유형 섹션을 참조하세요.

> **Important**
>
> 권한 경계 정책이 연결된 IAM 사용자 또는 역할에는 Deny 효과를 포함하는 NotPrincipal 정책 요소가 있는 리소스 기반 정책 문을 사용하지 않도록 합니다. Deny 효과를 포함하는 NotPrincipal 요소는 NotPrincipal 요소에 지정된 값에 관계없이 권한 경계 정책이 연결된 IAM 보안 주체를 항상 거부합니다. 이로 인해 원래 리소스에 액세스할 수 있는 일부 IAM 사용자 또는 역할이 해당 리소스에 더 이상 액세스할 수 없습니다. NotPrincipal 요소 대신 액세스를 제한하기 위해 ArnNotEquals 조건 연산자를 aws:PrincipalArn 컨텍스트 키와 함께 사용하도록 리소스 기반 정책 문을 변경하는 것이 좋습니다. NotPrincipal 요소에 대한 자세한 내용은 AWS JSON 정책 요소: NotPrincipal 섹션을 참조하세요.

AWS 관리형 정책 또는 고객 관리형 정책을 사용하여 IAM 엔터티(사용자 또는 역할) 경계를 설정할 수 있습니다. 이 정책은 사용자 또는 역할에 대해 최대 권한을 제한합니다.

예를 들어, ShirleyRodriguez라는 이름의 IAM 사용자는 Amazon S3, Amazon CloudWatch 및 Amazon EC2만 관리할 수 있어야 합니다. 이 규칙을 시행하려면 다은 정책을 사용하여 ShirleyRodriguez 사용자의 권한 경계를 설정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "cloudwatch:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

정책을 사용하여 사용자 권한 경계를 설정하면 이 정책은 사용자 권한을 제한하지만 자체적으로 권한을 제공하지 않습니다. 이 예제에서 정책은 ShirleyRodriguez의 최대 권한을 Amazon S3, CloudWatch 및 Amazon EC2의 모든 작업으로 설정합니다. Shirley가 작업을 허용하는 권한 정책이 있다고 해도 IAM을 포함한 다른 서비스에서는 이 작업을 절대 수행할 수 없습니다. 예를 들어 다음 정책을 ShirleyRodriguez 사용자에게 추가할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:CreateUser",
    "Resource": "*"
  }
}
```

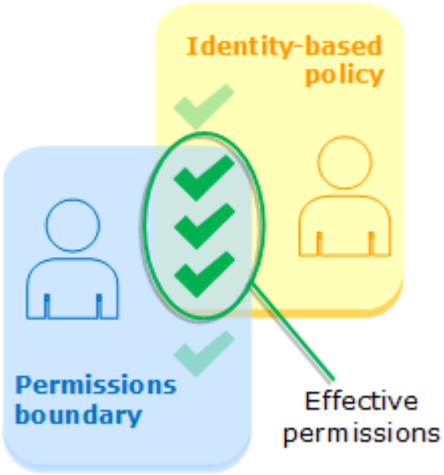
이 정책은 IAM에서 사용자 생성을 허용합니다. 이 권한 정책을 `ShirleyRodriguez` 사용자에 연결하고 Shirley가 사용자를 생성하고자 할 경우 작업은 실패합니다. 그 이유는 권한 경계가 `iam:CreateUser` 작업을 허용하지 않기 때문입니다. 이 두 가지 정책을 감안할 때 Shirley는 AWS에서 작업을 수행할 권한이 없습니다. 다른 서비스(예: Amazon S3)에서 작업을 허용하려면 다른 권한 정책을 추가해야 합니다. 또는 권한 경계를 업데이트하여 그녀에게 IAM에서 사용자를 생성하도록 허용할 수도 있습니다.

경계가 있는 효과적인 권한 평가

IAM 엔터티(사용자 또는 역할)에 대한 권한 경계는 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 사용자 또는 역할에 대한 효과적 권한을 변경할 수 있습니다. 사용자 또는 역할에 영향을 주는 모든 정책을 통해 부여되는 권한이 개체에 대한 유효 권한입니다. 계정 내에서 엔터티에 대한 권한은 자격 증명 기반 정책, 리소스 기반 정책, 권한 경계, 조직 SCP 또는 세션 정책에 영향을 받을 수 있습니다. 다양한 유형의 정책에 대한 자세한 정보는 IAM의 정책 및 권한 섹션을 참조하세요.

이러한 정책 유형 중 하나에서 작업에 대한 액세스가 명시적으로 거부된 경우 해당 요청이 거부됩니다. 여러 권한 유형에 의해 엔터티에 부여된 권한은 훨씬 더 복잡합니다. AWS의 정책 평가에 대한 자세한 정보는 정책 평가 로직 섹션을 참조하세요.

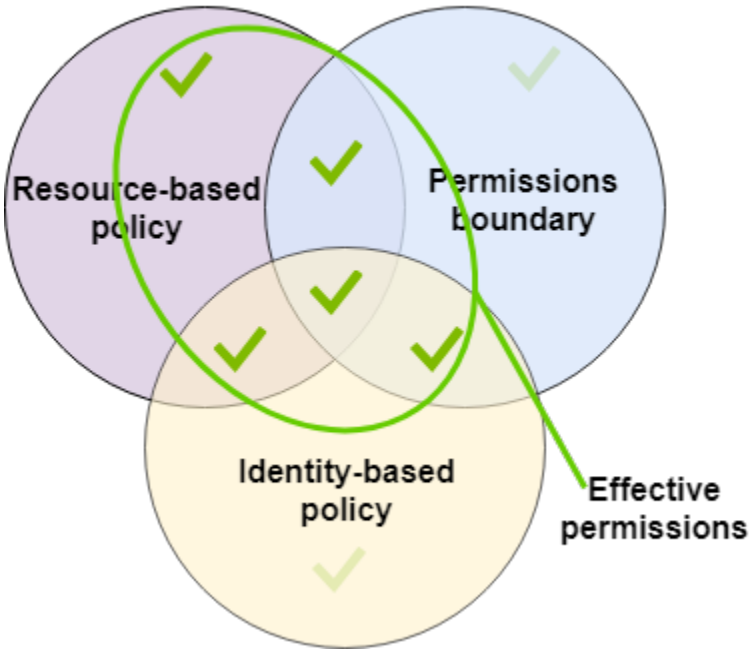
자격 증명 기반 정책과 경계 - 자격 증명 기반 정책은 사용자, 사용자 그룹 또는 역할에 연결된 인라인 또는 관리형 정책입니다. 자격 증명 기반 정책은 엔터티에 권한을 부여하며, 권한 경계는 이러한 권한을 제한합니다. 유효 권한은 두 정책 유형의 교집합입니다. 이들 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다.



리소스 기반 정책 - 리소스 기반 정책은 지정된 보안 주체가 정책이 연결된 리소스에 액세스하는 방식을 제어합니다.

IAM 사용자를 위한 리소스 기반 정책

동일한 계정 내에서 IAM 사용자 ARN(페더레이션 사용자 세션이 아님)에게 권한을 부여하는 리소스 기반 정책은 자격 증명 기반 정책 또는 권한 경계의 암시적 거부에 의해 제한되지 않습니다.



IAM 역할에 대한 리소스 기반 정책

IAM 역할 - IAM 역할 ARN에 권한을 부여하는 리소스 기반 정책은 권한 경계 또는 세션 정책의 암시적 거부에 의해 제한됩니다.

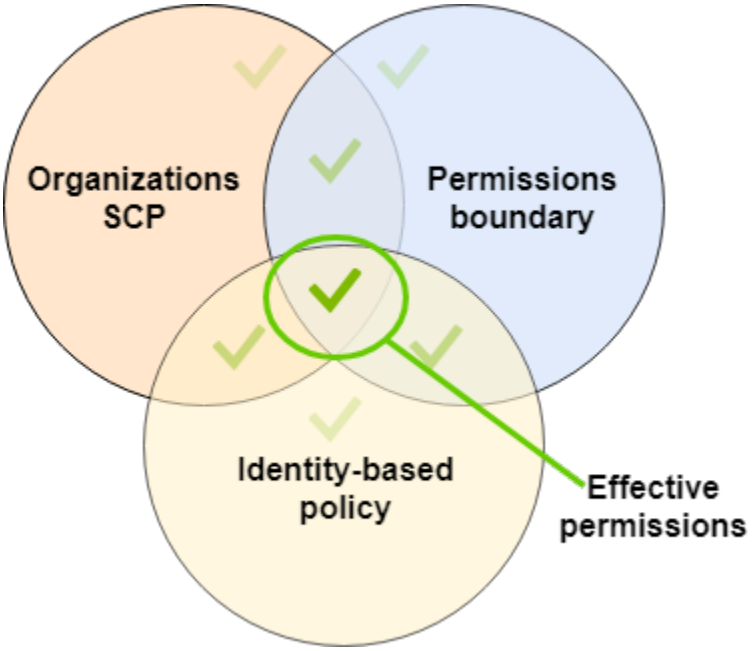
IAM 역할 세션 - 동일한 계정 내에서 IAM 역할 세션 ARN에 권한을 부여하는 리소스 기반 정책은 수임된 역할 세션에 직접 권한을 부여합니다. 세션에 직접 부여된 사용 권한은 자격 증명 기반 정책, 권한 경계 또는 세션 정책의 암시적 거부에 의해 제한되지 않습니다. 역할을 맡고 요청을 할 때 요청을 수행하는 보안 주체는 역할 자체의 ARN이 아니라 IAM 역할 세션 ARN입니다.

IAM 역할 및 페더레이션 사용자에 대한 리소스 기반 정책

IAM 페더레이션 사용자 세션 - IAM 페더레이션 사용자 세션은 `GetFederationToken` 호출을 통해 생성된 세션입니다. 페더레이션 사용자가 요청을 할 때 요청을 수행하는 보안 주체는 페더레이션된 IAM 사용자의 ARN이 아니라 페더레이션 사용자 ARN입니다. 동일한 계정 내에서 페더레이션 사용자 ARN에게 권한을 부여하는 리소스 기반 정책은 세션에 직접 권한을 부여합니다. 세션에 직접 부여된 사용 권한은 자격 증명 기반 정책, 권한 경계 또는 세션 정책의 암시적 거부에 의해 제한되지 않습니다.

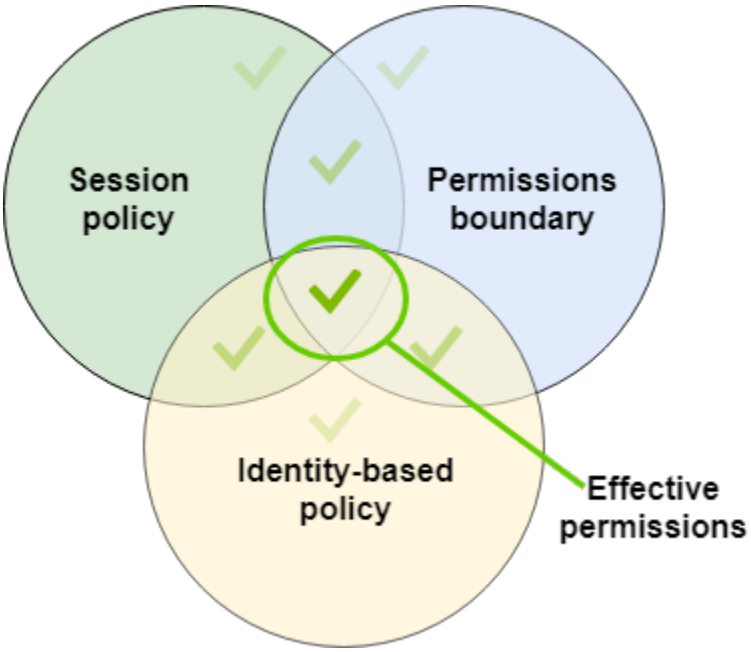
그러나 리소스 기반 정책이 페더레이션한 IAM 사용자의 ARN에 권한을 부여하는 경우, 세션 중에 페더레이션 사용자가 요청한 요청은 권한 경계 또는 세션 정책의 암시적 거부에 의해 제한됩니다.

조직 SCP - SCP는 전체 AWS 계정에 적용됩니다. SCP는 해당 계정 내 보안 주체가 보낸 모든 요청에 대한 권한을 제한합니다. IAM 엔터티(사용자 또는 역할)는 SCP, 권한 경계 및 자격 증명 기반 정책의 영향을 받는 요청을 수행할 수 있습니다. 이 경우, 세 정책 유형 모두에서 허용하는 경우에만 해당 요청이 허용됩니다. 유효 권한은 세 정책 유형 모두의 교집합입니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다.



AWS Organizations에서 계정이 조직의 멤버인지 여부를 알아볼 수 있습니다. 조직 멤버가 SCP의 영향을 받을 수 있습니다. AWS CLI 명령 또는 AWS API 작업을 사용하여 이 데이터를 보려면 조직 엔터티에 대해 `organizations:DescribeOrganization` 작업 권한이 있어야 합니다. 조직 콘솔에서 작업을 수행할 추가 권한이 있어야 합니다. SCP가 특정 요청에 대한 액세스를 거부하는지 여부를 확인하거나 유효 권한을 변경하려면 AWS Organizations 관리자에게 문의하세요.

세션 정책 - 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 세션에 대한 권한은 세션을 생성하는 데 사용되는 IAM 엔터티(사용자 또는 역할)와 세션 정책에서 가져옵니다. 엔터티의 자격 증명 기반 정책 권한은 세션 정책과 권한 경계에 제한을 받습니다. 이 정책 유형 집합의 유효 권한은 세 정책 유형 모두의 교집합입니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 세션 정책에 대한 자세한 정보는 세션 정책을 참조하세요.



권한 경계를 사용하여 다른 사용자에게 책임 위임

권한 경계를 사용하여 사용자 생성과 같은 권한 관리 작업을 계정의 IAM 사용자에 위임할 수 있습니다. 이로써 권한의 특정 경계 내에서 다른 사용자가 작업을 대신 수행할 수 있는 권한이 부여됩니다.

예를 들어, María가 X-Company AWS 계정 관리자라고 가정하세요. Maria가 Zhang에게 사용자 생성 업무를 위임하고자 합니다. 하지만 Zhang이 다음 회사 규칙에 따라 사용자를 생성하는지 확신해야 합니다.

- 사용자는 IAM을 사용하여 사용자, 그룹, 역할 또는 정책을 생성하고 관리할 수 없습니다.

- 사용자의 Amazon S3 `logs` 버킷 액세스가 거부되고 사용자가 `i-1234567890abcdef0` Amazon EC2 인스턴스로 액세스할 수 없습니다.
- 사용자는 사용자 자체 경계 정책을 제거할 수 없습니다.

이런 규칙을 시행하기 위해서는 María는 아래와 같은 세부 정보가 포함된 작업을 완료합니다.

1. María는 `XCompanyBoundaries` 관리형 정책을 생성하여 계정의 모든 새로운 사용자에 대한 권한 경계로서 사용할 수 있습니다.
2. María는 `DelegatedUserBoundary` 관리형 정책을 생성하여 Zhang에 대한 권한 경계로서 할당합니다. Maria는 관리자 사용자의 ARN을 기록하고 정책에서 사용하여 Zhang이 해당 ARN에 액세스하지 못하도록 합니다.
3. María는 `DelegatedUserPermissions` 관리형 정책을 생성하여 Zhang에 대한 권한 정책으로서 연결합니다.
4. María가 Zhang에서 그의 새로운 책임과 제한을 알려줍니다.

작업 1: María는 먼저 관리형 정책을 생성하여 새로운 사용자에 대한 경계를 정의해야 합니다. María는 Zhang이 사용자에게 필요한 권한 정책을 사용자에게 부여할 수 있도록 허용하지만 사용자를 제한하고자 합니다. 이렇게 하기 위해서는 마리아는 `XCompanyBoundaries`라는 다음 고객 관리형 정책을 생성합니다. 이 정책은 다음을 수행합니다.

- 사용자에게 여러 서비스에 대한 전체 액세스 권한을 부여
- IAM 콘솔에서 제한된 자체 관리 액세스 허용 즉, 콘솔에 로그인한 후 암호를 변경할 수 있습니다. 초기 암호를 설정할 수 없습니다. 이 작업을 허용하려면 `"*LoginProfile"` 작업을 `AllowManageOwnPasswordAndAccessKeys` 문에 추가합니다.
- Amazon S3 로그 버킷 또는 `i-1234567890abcdef0` Amazon EC2 인스턴스로의 사용자 액세스 거부

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ServiceBoundaries",
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "cloudwatch:*",
```

```
                "ec2:*",
                "dynamodb:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowIAMConsoleForCredentials",
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers",
                "iam:GetAccountPasswordPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswordAndAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:*AccessKey*",
                "iam:ChangePassword",
                "iam:GetUser",
                "iam:*ServiceSpecificCredential*",
                "iam:*SigningCertificate*"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "DenyS3Logs",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::logs",
                "arn:aws:s3:::logs/*"
            ]
        },
        {
            "Sid": "DenyEC2Production",
            "Effect": "Deny",
            "Action": "ec2:*",
            "Resource": "arn:aws:ec2:*:*:instance/i-1234567890abcdef0"
        }
    ]
}
```

각 설명문은 다른 목적이 있습니다.

1. 이 정책의 `ServiceBoundaries` 설명문은 지정된 AWS 서비스에 대한 완전한 액세스를 허용합니다. 이런 서비스의 새로운 사용자 작업이 사용자에 연결된 권한 정책에 따라서만 제한된다는 의미입니다.
2. `AllowIAMConsoleForCredentials` 문에서 모든 IAM 사용자를 나열할 수 있는 액세스를 허용합니다. 이 액세스는 AWS Management Console의 사용자 페이지를 탐색하는 데 필요합니다. 또한 계정의 암호 요구 사항을 확인하도록 허용합니다. 이 액세스는 자신의 고유 암호를 변경할 때 필요합니다.
3. `AllowManageOwnPasswordAndAccessKeys` 문을 통해 사용자가 고유한 콘솔 암호와 프로그래밍 방식의 액세스 키만 관리할 수 있습니다. 이는 Zhang 또는 다른 관리자가 새로운 사용자에게 전체 IAM 액세스를 포함하는 권한 정책을 할당할 때 중요합니다. 이 경우, 해당 사용자가 자신 또는 다른 사용자의 권한을 변경할 수 있습니다. 이 설명문은 이런 상황을 방지할 수 있습니다.
4. `DenyS3Logs` 설명문은 `logs` 버킷 액세스를 명시적으로 거부합니다.
5. `DenyEC2Production` 설명문은 `i-1234567890abcdef0` 인스턴스 액세스를 명시적으로 거부합니다.

작업 2: María는 Zhang이 모든 X-Company 사용자를 생성하도록 허용하지만 `XCompanyBoundaries` 권한 경계를 통해서만 허용하고자 합니다. 마리아는 `DelegatedUserBoundary`라는 다음 고객 관리형 정책을 생성합니다. 이런 정책은 Zhang이 가질 수 있는 최대 권한을 정의합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateOrChangeOnlyWithBoundary",
            "Effect": "Allow",
            "Action": [
                "iam:AttachUserPolicy",
                "iam:CreateUser",
                "iam:DeleteUserPolicy",
                "iam:DetachUserPolicy",
                "iam:PutUserPermissionsBoundary",
                "iam:PutUserPolicy"
            ],
            "Resource": "*",
            "Condition": {
               "StringEquals": {
```

```
                "iam:PermissionsBoundary": "arn:aws:iam::123456789012:policy/
XCompanyBoundaries"
                }
            }
        },
        {
            "Sid": "CloudWatchAndOtherIAMTasks",
            "Effect": "Allow",
            "Action": [
              "cloudwatch:*",
              "iam:CreateAccessKey",
              "iam:CreateGroup",
              "iam:CreateLoginProfile",
              "iam:CreatePolicy",
              "iam:DeleteGroup",
              "iam:DeletePolicy",
              "iam:DeletePolicyVersion",
              "iam:DeleteUser",
              "iam:GetAccountPasswordPolicy",
              "iam:GetGroup",
              "iam:GetLoginProfile",
              "iam:GetPolicy",
              "iam:GetPolicyVersion",
              "iam:GetRolePolicy",
              "iam:GetUser",
              "iam:GetUserPolicy",
              "iam:ListAccessKeys",
              "iam:ListAttachedRolePolicies",
              "iam:ListAttachedUserPolicies",
              "iam:ListEntitiesForPolicy",
              "iam:ListGroups",
              "iam:ListGroupsForUser",
              "iam:ListMFADevices",
              "iam:ListPolicies",
              "iam:ListPolicyVersions",
              "iam:ListRolePolicies",
              "iam:ListSSHPublicKeys",
              "iam:ListServiceSpecificCredentials",
              "iam:ListSigningCertificates",
              "iam:ListUserPolicies",
              "iam:ListUsers",
              "iam:SetDefaultPolicyVersion",
              "iam:SimulateCustomPolicy",
              "iam:SimulatePrincipalPolicy",
```

```
                "iam:UpdateGroup",
                "iam:UpdateLoginProfile",
                "iam:UpdateUser"
            ],
            "NotResource": "arn:aws:iam::123456789012:user/Maria"
        },
        {
            "Sid": "NoBoundaryPolicyEdit",
            "Effect": "Deny",
            "Action": [
                "iam:CreatePolicyVersion",
                "iam:DeletePolicy",
                "iam:DeletePolicyVersion",
                "iam:SetDefaultPolicyVersion"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:policy/XCompanyBoundaries",
                "arn:aws:iam::123456789012:policy/DelegatedUserBoundary"
            ]
        },
        {
            "Sid": "NoBoundaryUserDelete",
            "Effect": "Deny",
            "Action": "iam:DeleteUserPermissionsBoundary",
            "Resource": "*"
        }
    ]
}
```

각 설명문은 다른 목적이 있습니다.

1. `CreateOrChangeOnlyWithBoundary` 설명문은 Zhang이 IAM 사용자를 생성하도록 허용하지만 Zhang이 권한 경계를 설정할 때 `XCompanyBoundaries` 정책을 사용할 경우에만 가능합니다. 이 설명문은 또한 장이 기존 사용자에 대한 권한 경계를 설정하도록 허용하지만 장이 동일한 정책을 사용할 때만 가능합니다. 마지막으로, 이 설명문은 Zhang이 이 권한 경계 설정을 통해 사용자에 대한 권한 정책을 관리하도록 허용합니다.
2. `CloudWatchAndOtherIAMTasks` 설명문은 Zhang이 사용자, 그룹 및 정책 관리 작업을 완료하도록 허용합니다. 그는 `NotResource` 정책 요소에 나열되지 않은 IAM 사용자의 암호를 재설정하고 액세스 키를 생성할 수 있는 권한이 있습니다. 따라서 그는 사용자가 로그인 문제를 해결하도록 지원할 수 있습니다.

3. NoBoundaryPolicyEdit 설명문은 Zhang이 XCompanyBoundaries 정책을 업데이트할 수 있는 액세스를 거부합니다. 장은 자신 또는 다른 사용자에 대한 권한 경계를 설정하는 데 사용되는 어떤 정책도 변경할 수 없습니다.
4. NoBoundaryUserDelete 문에서는 Zhang이 자신 또는 다른 사용자에 대해 권한 경계를 삭제하기 위해 액세스할 때 이를 거부합니다.

그런 다음 María는 Zhang 사용자에 대한 권한 경계로서 DelegatedUserBoundary 정책을 할당합니다.

작업 3: 권한 경계가 최대 권한을 제한하지만 자체 액세스를 허용하지 않기 때문에 Maria는 Zhang에 대한 권한 정책을 생성해야 합니다. 마리아는 DelegatedUserPermissions라는 다음 정책을 생성합니다. 이 정책은 정의된 경계 내에서 Zhang이 수행할 수 있는 작업을 정의합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "IAM",
            "Effect": "Allow",
            "Action": "iam:*",
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchLimited",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetDashboard",
                "cloudwatch:GetMetricData",
                "cloudwatch:ListDashboards",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "S3BucketContents",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::ZhangBucket"
        }
    ]
```

```
}
```

각 설명문은 다른 목적이 있습니다.

1. 정책의 `IAM` 설명문은 IAM에 대한 Zhang의 완전한 액세스를 허용합니다. 그러나, Zhang 권한 경계가 몇 가지 IAM 작업만 허용하기 때문에 Zhang의 효과적인 IAM 권한은 그의 권한 경계에 의해서만 제한됩니다.
2. `CloudWatchLimited` 설명문은 Zhang이 CloudWatch에서 5가지 작업을 수행할 수 있도록 허용합니다. Zhang 권한 경계는 CloudWatch의 모든 작업을 허용하기 때문에 그의 효과적인 CloudWatch 권한은 그의 권한 정책에 의해서만 제한됩니다.
3. `S3BucketContents` 설명문은 Zhang이 `ZhangBucket` Amazon S3 버킷을 나열할 수 있도록 허용합니다. 그러나, Zhang 권한 경계는 어떠한 Amazon S3 작업도 허용하지 않기 때문에 Zhang은 그의 권한 정책과 상관없이 어떠한 S3 작업도 수행할 수 없습니다.

> **Note**
>
> Zhang의 정책을 통해 자신은 액세스할 수 없는 Amazon S3 리소스에 액세스할 수 있는 사용자를 생성할 수 있습니다. Maria는 이러한 관리 작업을 위임하여 Amazon S3에 대한 액세스 권한이 있는 Zhang을 효과적으로 신뢰합니다.

그러면 María는 `DelegatedUserPermissions` 정책을 Zhang 사용자에 대한 권한 정책으로서 연결합니다.

작업 4: María는 새로운 사용자를 생성하도록 Zhang에게 지침을 내립니다. María는 Zhang에게 새로운 사용자가 원하는 모든 권한을 통해 새로운 사용자를 생성할 수 있지만 `XCompanyBoundaries` 정책을 권한 경계로서 할당해야 한다고 말합니다.

Zhang은 다음 작업을 완료합니다.

1. Zhang은 AWS Management Console로 사용자를 생성합니다. 그는 사용자 이름 `Nikhil`를 입력하고 사용자에 대한 콘솔 액세스를 가능하게 합니다. 위의 정책에서는 사용자가 IAM 콘솔에 로그인한 후에만 암호를 변경할 수 있으므로 암호 재설정 필요 옆의 확인란 선택을 취소합니다.
2. 권한 설정 페이지에서 Zhang은 Nikhil가 업무를 할 수 있도록 허용하는 IAMFullAccess 및 AmazonS3ReadOnlyAccess 권한 정책을 선택합니다.
3. Zhang은 María의 지침을 잊고 Set permissions boundary(권한 경계 설정) 섹션을 넘깁니다.
4. Zhang은 사용자 세부 정보를 검토하고 사용자 생성을 선택합니다.

작업은 실패하고 액세스는 거부됩니다. Zhang의 `DelegatedUserBoundary` 권한 경계는 그가 생성하는 어떠한 사용자도`XCompanyBoundaries` 정책을 권한 경계로서 가지고 있어야 합니다.

5. Zhang은 이전 페이지로 돌아갑니다. 그는 권한 경계 설정 페이지에서 `XCompanyBoundaries` 정책을 선택합니다.
6. Zhang은 사용자 세부 정보를 검토하고 사용자 생성을 선택합니다.

   사용자가 생성됩니다.

Nikhil이 로그인할 경우, 그는 권한 경계에 의해 거부된 작업 이외의 IAM 및 Amazon S3에 액세스할 수 있습니다. 예를 들어, 그는 IAM에 자신의 암호를 변경할 수 있지만 다른 사용자를 생성하거나 그의 정책을 편집할 수 없습니다. Nikhil은 Amazon S3에 대한 읽기 전용 액세스 권한이 있습니다.

누군가가 `logs` 버킷에 Nikhil이 버킷에 객체를 넣을 수 있도록 허용하는 리소스 기반 정책을 추가하더라도 그는 여전히 이 버킷에 액세스할 수 없습니다 `logs` 버킷에 대한 작업이 권한 경계에 의해 명시적으로 거부되었기 때문입니다. 정책 유형에 포함된 명시적 거부로 인해 요청이 거부됩니다. 하지만 Secrets Manager 암호에 연결된 리소스 기반 정책이 Nikhil이 `secretsmanager:GetSecretValue` 작업을 수행하도록 허용하는 경우 Nikhil은 암호를 불러와서 암호화를 해제할 수 있습니다. 그 이유는 Secrets Manager 작업이 Nikhill의 권한 경계에 의해 명시적으로 거부되지 않았고 권한 경계에서의 묵시적 거부가 리소스 기반 정책을 제한하지 않기 때문입니다.

자격 증명 기반 정책 및 리소스 기반 정책

정책은 자격 증명 또는 리소스에 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. 리소스에 대한 액세스를 제한하는 권한 정책을 생성할 때 자격 증명 기반 정책 또는 리소스 기반 정책을 선택할 수 있습니다.

자격 증명 기반 정책은 IAM 사용자, 그룹 또는 역할에 연결됩니다. 이러한 정책으로 자격 증명이 수행할 수 있는 작업(권한)을 지정할 수 있습니다. 예를 들어, John이라는 IAM 사용자에게 Amazon EC2 `RunInstances` 작업을 수행하도록 허용하는 정책을 연결할 수 있습니다. 이 정책은 John이 `MyCompany`이라는 Amazon DynamoDB 테이블에서 항목을 가져오도록 허용되었다는 내용도 명시할 수 있습니다. 또한 John에게 자신의 IAM 보안 자격 증명을 관리하도록 허용할 수도 있습니다. 자격 증명 기반 정책은 관리형 권한 또는 인라인 권한이 될 수 있습니다.

리소스 기반 정책은 리소스에 연결됩니다. 예를 들어, 리소스 기반 정책을 Amazon S3 버킷, Amazon SQS 대기열, VPC 엔드포인트, AWS Key Management Service 암호화 키, Amazon DynamoDB 테이블 및 스트림에 연결할 수 있습니다. 리소스 기반 정책을 지원하는 서비스 목록은 AWS IAM으로 작업하는 서비스 섹션을 참조하세요.

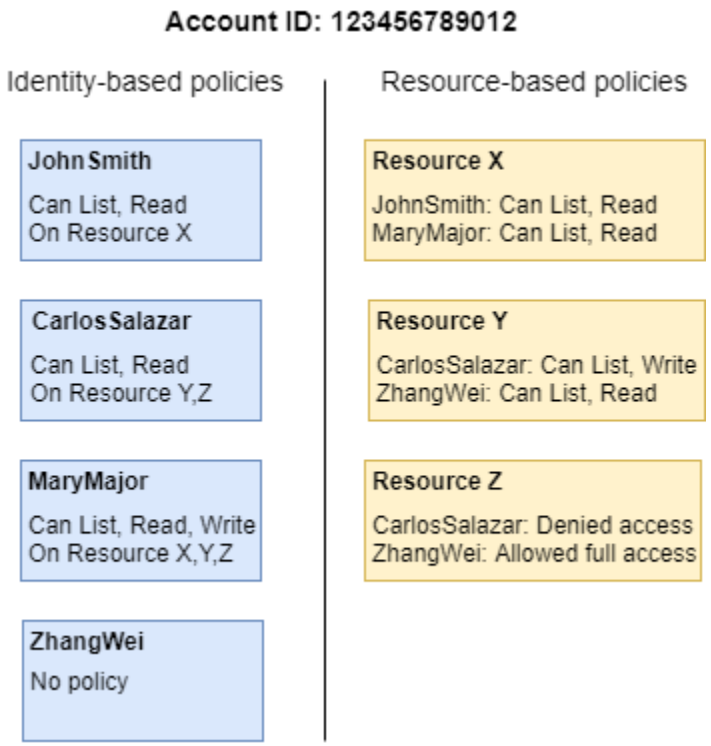
리소스 기반 정책을 사용하면 이러한 리소스에 액세스할 수 있는 대상 및 해당 대상이 리소스에서 수행할 수 있는 작업을 지정할 수 있습니다. 해당 신뢰 영역(신뢰할 수 있는 조직 또는 계정) 외의 계정 내 보안 주체가 역할을 수임하는 권한이 있는지 자세히 알고 싶다면, IAM Access Analyzer란 무엇일까요?를 참조하세요. 리소스 기반 정책은 인라인만 있고 관리형은 없습니다.

> **Note**
>
> 리소스 기반 정책은 리소스 수준 권한과 다릅니다. 이 주제에서 설명한 바와 같이 리소스 기반 정책을 리소스에 직접 연결할 수 있습니다. 리소스 수준 권한이란 ARN을 사용하여 정책에서 개별 리소스를 지정하는 기능을 말합니다. 리소스 기반 정책은 일부 AWS 서비스에서만 지원됩니다. 리소스 기반 정책 및 리소스 수준 권한을 지원하는 서비스 목록은 AWS IAM으로 작업하는 서비스 섹션을 참조하세요.

자격 증명 기반 정책 및 리소스 기반 정책이 동일한 계정 내에서 상호 작용하는 방법을 알아보려면 단일 계정 내에서 정책 평가 섹션을 참조하세요.

정책이 계정 간에 상호 작용하는 방식을 알아보려면 교차 계정 정책 평가 로직 섹션을 참조하세요.

이러한 개념에 대한 이해도를 높이려면 다음 그림 섹션을 참조하세요. `123456789012` 계정의 관리자는 자격 증명 기반 정책을 `JohnSmith`, `CarlosSalazar` 및 `MaryMajor` 사용자에게 연결합니다. 이 정책의 일부 작업은 특정 리소스에서 수행할 수 있습니다. 예를 들어 사용자 `JohnSmith`는 `Resource X`에 대해 일부 작업을 수행할 수 있습니다. 이는 자격 증명 기반 정책에서 리소스 수준 권한입니다. 관리자는 또한 리소스 기반 정책을 `Resource X`, `Resource Y` 및 `Resource Z`에 추가했습니다. 리소스 기반 정책을 통해 해당 리소스에 액세스할 수 있는 사용자를 지정할 수 있습니다. 예를 들어 `Resource X`의 리소스 기반 정책은 `JohnSmith` 및 `MaryMajor` 사용자 목록을 표시하고 리소스에 대한 읽기 권한을 허용합니다.



123456789012 계정의 예를 사용하면 다음 사용자가 나열된 작업을 수행할 수 있습니다.

- JohnSmith - John은 Resource X에서 나열 및 읽기 작업을 수행할 수 있습니다. John은 사용자에 대한 자격 증명 기반 정책과 Resource X에 대한 리소스 기반 정책을 통해 이 권한을 부여 받습니다.
- CarlosSalazar - Carlos는 Resource Y에서 나열, 읽기 및 쓰기 작업을 수행할 수 있지만 Resource Z에 대한 액세스는 거부됩니다. Carlos의 자격 증명 기반 정책을 통해 Resource Y에서 나열 및 읽기 작업을 수행할 수 있습니다. Resource Y 리소스 기반 정책을 사용하면 Carlos에게 쓰기 권한도 허용됩니다. 그러나 자격 증명 기반 정책을 통해 Resource Z에 대한 액세스가 허용되더라도 Resource Z 리소스 기반 정책으로 인해 해당 액세스가 거부됩니다. 명시적 Deny는 Allow를 재정의하므로 Carlos의 Resource Z에 대한 액세스가 거부됩니다. 자세한 내용은 정책 평가 로직 단원을 참조하십시오.
- MaryMajor - Mary는 Resource X, Resource Y 및 Resource Z에 대해 나열, 읽기 및 쓰기 작업을 수행할 수 있습니다. Mary의 자격 증명 기반 정책을 통해 리소스 기반 정책보다 더 많은 리소스에 대해 더 많은 작업을 수행할 수 있지만 액세스를 거부하는 정책은 없습니다.
- ZhangWei - Zhang에게는 Resource Z에 대한 모든 액세스 권한이 있습니다. Zhang은 자격 증명 기반 정책이 없지만 Resource Z 리소스 기반 정책을 사용하면 리소스에 대한 전체 액세스 권한을 가질 수 있습니다. Zhang은 Resource Y에서 나열 및 읽기 작업을 수행할 수도 있습니다.

자격 증명 기반 정책과 리소스 기반 정책은 모두 권한 정책이며 함께 평가됩니다. 권한 정책만 적용되는 요청의 경우 AWS는 먼저 모든 정책에서 Deny를 확인합니다. 이 정책이 존재하는 경우 요청이 거부됩니다. 그런 다음 AWS는 각 Allow를 확인합니다. 적어도 하나의 정책 설명이 요청의 작업을 허용하는 경우 요청이 허용됩니다. Allow가 자격 증명 기반 정책인지 리소스 기반 정책인지는 중요하지 않습니다.

> **Important**
>
> 이 논리는 요청이 하나의 AWS 계정에서 이루어진 경우에만 적용됩니다. 하나의 계정에서 다른 계정으로 요청한 경우 Account A의 요청자는 Account B의 리소스에 대한 요청을 허용하는 자격 증명 기반 정책을 가지고 있어야 합니다. 또한 Account B의 리소스 기반 정책은 Account A의 요청자가 리소스에 액세스할 수 있도록 허용해야 합니다. 두 계정 모두에 작업을 허용하는 정책이 있어야 합니다. 그렇지 않으면 요청이 실패합니다. 크로스 계정 액세스에 대해 리소스 기반 정책을 사용하는 방법에 대한 자세한 정보는 IAM의 크로스 계정 리소스 액세스 섹션을 참조하세요.

특정 권한이 있는 사용자는 해당 권한에 연결된 권한 정책이 있는 리소스를 요청할 수 있습니다. 이 경우 AWS는 해당 리소스에 대한 액세스 권한을 부여할지 여부를 결정할 때 두 권한 세트를 모두 평가합니다. 정책이 평가되는 방식에 대한 자세한 정보는 정책 평가 로직 섹션을 참조하세요.

> **Note**
>
> Amazon S3는 자격 증명 기반 정책 및 리소스 기반 정책(버킷 정책이라고 함)을 지원합니다. 또한 Amazon S3는 IAM 정책 및 권한과 무관한, 액세스 제어 목록(ACL)으로 알려진 권한 메커니즘을 지원합니다. IAM 정책을 Amazon S3 ACL과 함께 사용할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 액세스 제어를 참조하세요.

정책을 사용한 AWS 리소스 액세스 제어

정책을 사용하여 IAM 또는 모든 AWS 내 리소스에 대한 액세스를 제어할 수 있습니다.

정책을 사용하여 AWS에서 액세스를 제어하려면 AWS이 액세스를 부여하는 방식을 이해해야 합니다. AWS는 리소스 모음으로 구성되어 있습니다. IAM 사용자는 리소스입니다. Amazon S3 버킷은 리소스입니다. AWS API, AWS CLI 또는 AWS Management Console을 사용하여 작업을 수행할 경우(예: 사용자 생성) 해당 작업에 대한 요청을 전송합니다. 이 요청은 작업, 리소스, 보안 주체 엔터티(사용자 또

는 역할), 보안 주체 계정 및 필요한 요청 정보를 지정합니다. 이러한 모든 정보는 콘텍스트를 제공합니다.

그런 다음 AWS은 사용자(보안 주체)가 지정된 리소스에 대해 지정된 작업을 수행할 수 있도록 인증(로그인) 및 권한 부여(권한 있음)되었는지 확인합니다. 권한을 부여하는 동안 AWS는 요청 콘텍스트에 적용되는 모든 정책을 확인합니다. 대부분의 정책은 AWS에 JSON 문서로 저장되며 보안 주체 엔터티에 대한 권한을 지정합니다. 정책 유형 및 활용에 대한 자세한 정보는 IAM의 정책 및 권한 섹션을 참조하세요.

AWS는 정책이 요청의 각 부분을 허용한 경우에만 요청에 권한을 부여합니다. 이러한 프로세스의 다이어그램을 보려면 IAM 작동 방식 섹션을 참조하세요. AWS가 요청이 허용되는지 여부를 결정하는 방법에 대한 자세한 내용은 정책 평가 로직 섹션을 참조하세요.

IAM 정책을 생성하면 다음에 대한 액세스를 제어할 수 있습니다.

- 보안 주체 - 요청하는 사용자(보안 주체)가 수행하도록 허용된 사항을 제어합니다.
- IAM 자격 증명 - 어떤 IAM 자격 증명(사용자 그룹, 사용자 및 역할)에 액세스할 수 있는지 및 그 방법을 제어합니다.
- IAM 정책 - 고객 관리형 정책을 생성, 편집 및 삭제할 수 있는 대상과 모든 관리형 정책을 연결하고 분리할 수 있는 대상을 제어합니다.
- AWS 리소스 - 자격 증명 기반 정책 또는 리소스 기반 정책을 사용하여 리소스에 액세스할 수 있는 대상을 제어합니다.
- AWS 계정 - 요청이 특정 계정의 멤버에만 허용되는지 여부를 제어합니다.

이러한 정책을 사용하여 AWS 리소스에 액세스할 수 있는 대상과 액세스한 대상이 리소스에서 수행할 수 있는 작업을 지정할 수 있습니다. 모든 IAM 사용자는 처음에 권한이 없습니다. 다시 말해, 기본적으로 사용자는 아무 작업도 할 수 없으며, 심지어 자신의 액세스 키를 볼 수도 없습니다. 사용자에게 작업을 수행할 권한을 부여하기 위해 사용자에게 권한을 추가(즉 사용자에게 정책 연결)하거나 의도한 권한을 보유한 사용자 그룹에 사용자를 추가할 수 있습니다.

예를 들어, 자신의 액세스 키를 나열할 사용자 권한을 부여할 수 있습니다. 해당 권한을 확장하여 각 사용자가 자신의 키를 생성, 업데이트 및 삭제하도록 할 수도 있습니다.

사용자 그룹에 권한을 부여하면 사용자 그룹에 속한 모든 사용자가 해당 권한을 얻습니다. 예를 들어, Administrators 사용자그룹에 AWS 계정 리소스에서 IAM 작업을 수행할 권한을 부여할 수 있습니다. 또 다른 예로 Managers 사용자 그룹에 AWS 계정의 Amazon EC2 인스턴스를 설명할 권한을 부여할 수 있습니다.

사용자, 사용자 그룹 및 역할에 기본 권한을 위임하는 방법에 대한 자세한 정보는 IAM 리소스에 액세스하는 데 필요한 권한 섹션을 참조하세요. 기본 권한을 보여주는 정책의 예를 더 보려면 IAM 리소스를 관리하기 위한 정책의 예 섹션을 참조하세요.

보안 주체에 대한 액세스 제어

정책을 사용하여 요청하는 사용자(보안 주체)가 수행하도록 허용된 사항을 제어할 수 있습니다. 이렇게 하려면 자격 증명 기반 정책을 해당 사용자의 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결해야 합니다. 또한 권한 경계를 사용하여 엔터티(사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정할 수 있습니다.

예를 들어 사용자 Zhang Wei의 CloudWatch, Amazon DynamoDB, Amazon EC2 및 Amazon S3에 대한 완전한 액세스를 허용하고자 한다고 가정해 봅시다. 다른 사용자에 대해 권한 세트 한 개가 필요한 경우 나중에 분리할 수 있도록 두 가지 다른 정책을 생성할 수 있습니다. 또는 모든 권한을 단일 정책으로 모은 다음 이 정책을 이름이 Zhang Wei인 IAM 사용자에게 연결할 수 있습니다. 정책을 Zhang Wei가 속한 사용자 그룹 또는 Zhang Wei가 수임하는 역할에 연결할 수도 있습니다. 그 결과, Zhang이 S3 버킷의 내용을 볼 경우 해당 요청이 허용됩니다. 새 IAM 사용자를 생성하려고 시도할 경우에는 권한이 없으므로 요청이 거부됩니다.

Zhang의 권한 경계를 사용하여 Zhang에게 `amzn-s3-demo-bucket1` S3 버킷으로의 액세스 권한을 부여해야 합니다. 이렇게 하기 위해서는 Zhang에게 부여하고자 하는 최대 권한을 결정합니다. 이런 경우, Zhang이 그의 권한 정책으로 하는 일을 제어합니다. 여기서는 Zhang이 기밀 버킷으로 액세스하지만 않도록 신경 씁니다. 따라서 다음 정책을 사용하여 Zhang의 경계를 정의하여 Amazon S3에 대한 모든 AWS 작업 및 몇 가지 기타 서비스를 허용하지만 `amzn-s3-demo-bucket1` S3 버킷으로의 액세스는 거부합니다. 권한 경계가 모든 IAM 작업을 허용하지 않기 때문에 권한 경계는 Zhang이 그의(또는 어떠한 사람의) 경계를 삭제하지 못하도록 방지합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionsBoundarySomeServices",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:*",
                "dynamodb:*",
                "ec2:*",
                "s3:*"
            ],
            "Resource": "*"
        },
```

```
        {
            "Sid": "PermissionsBoundaryNoConfidentialBucket",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1",
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ]
        }
    ]
}
```

이 사용자에 대한 권한 경계처럼 정책을 할당할 경우 어떠한 권한도 허용하지 않는다는 점을 유의하세요. 권한 경계는 자격 증명 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 권한 경계에 대한 자세한 정보는 IAM 엔터티의 권한 범위 섹션을 참조하세요.

이전 절차에 대한 자세한 정보는 이러한 리소스 섹션을 참조하세요.

- 보안 주체에 연결할 수 있는 IAM 정책의 생성에 대해 자세히 알아보려면 고객 관리형 정책으로 사용자 지정 IAM 권한 정의 섹션을 참조하세요.
- 보안 주체에 IAM 정책을 연결하는 방법에 대해 자세히 알아보려면 IAM 자격 증명 권한 추가 및 제거 섹션을 참조하세요.
- EC2에 전체 액세스 권한을 부여하는 예시 정책을 보려면 Amazon EC2: 특정 리전 내에서의 모든 EC2 액세스를 프로그래밍 방식으로 콘솔에서 허용 섹션을 참조하세요.
- S3 버킷에 읽기 전용 액세스를 허용하려면 Amazon S3: S3 버킷에 있는 객체에 대한 읽기 및 쓰기 액세스 권한을 프로그래밍 방식으로 콘솔에서 허용 예시 정책의 첫 번째 두 설명문을 사용하세요.
- 사용자에게 콘솔 암호, 프로그래밍 방식의 액세스 키, MFA 디바이스 등 사용자의 보안 인증을 설정하도록 허용하는 정책 예제는 AWS: MFA 인증 IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용 섹션을 참조하세요.

자격 증명에 대한 액세스 제어

IAM 정책에서 사용자 그룹 전반의 모든 사용자에게 연결하는 정책을 생성함으로써 사용자가 자격 증명에 대해 수행할 수 있는 사항을 제어할 수 있습니다. 이렇게 하려면 자격 증명에 수행할 수 있는 사항 또는 자격 증명에 액세스할 수 있는 대상을 제어하는 정책을 생성합니다.

예를 들어, 이름이 AllUsers인 사용자 그룹을 생성한 다음 해당 사용자 그룹을 모든 사용자에 연결할 수 있습니다. 사용자 그룹을 생성할 때 이전 섹션에서 설명한 대로 모든 사용자에게 보안 인증을 설정하기 위한 액세스 권한을 부여할 수 있습니다. 그런 다음 정책 조건에 사용자 이름이 포함되지 않은 경우 사

용자 그룹을 변경하는 액세스를 거부하는 정책을 생성할 수 있습니다. 그러나 정책에서 이 부분은 나열된 사용자를 제외한 모든 사용자의 액세스만 거부합니다. 또한 사용자 그룹 사용자 모두에 대한 모든 사용자 그룹 관리 작업을 허용하는 권한을 포함해야 합니다. 마지막으로, 모든 사용자에게 적용되도록 이 정책을 사용자 그룹에 연결합니다. 그 결과, 정책에 지정되지 않은 사용자가 사용자 그룹을 변경하려고 하면 해당 요청이 거부됩니다.

시각적 편집기를 사용하여 이 정책을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 정책을 선택합니다.

   정책을 처음으로 선택하는 경우 관리형 정책 소개 페이지가 나타납니다. 시작하기(Get Started)를 선택합니다.
3. 정책 생성을 선택합니다.
4. 정책 편집기 섹션에서 시각적 편집기 옵션을 선택합니다.
5. 서비스 선택에서 IAM을 선택합니다.
6. 허용된 작업에서 검색 상자에 **group**을 입력합니다. 시각적 편집기에 group이라는 단어가 포함된 모든 IAM 작업이 표시됩니다. 모든 확인란을 선택합니다.
7. 리소스를 선택하여 정책에 대한 리소스를 지정합니다. 선택한 작업에 따라 group과 user 리소스 유형이 표시됩니다.

   - 그룹 - ARN 추가를 선택합니다. 리소스 내부의 경우 모든 계정 옵션을 선택합니다. 경로를 포함하는 모든 그룹 이름 확인란을 선택한 다음, 사용자 그룹 이름 **AllUsers**를 입력합니다. 그런 다음 ARN 추가를 선택합니다.
   - 사용자 - 이 계정에서 모두 옆에 있는 확인란을 선택합니다.

   선택한 작업 중 하나인 ListGroups는 특정 리소스 사용을 지원하지 않습니다. 해당 작업에서 All resources(모든 리소스)를 선택할 필요가 없습니다. 정책을 저장하거나 JSON 편집기에서 정책을 보는 경우 이 모든 리소스에 대해 IAM이 작업 권한을 부여하는 새 권한 블록을 자동으로 생성하는 것을 확인할 수 있습니다.
8. 다른 권한 블록을 추가하려면 더 많은 권한 추가를 선택합니다.
9. 서비스 선택을 선택한 다음 IAM을 선택합니다.
10. 허용된 작업을 선택한 다음 권한 거부로 전환을 선택합니다. 이렇게 하면 권한을 거부할 때 전체 블록이 사용됩니다.

11. 검색 상자에 **group**을 입력합니다. 시각적 편집기에 group이라는 단어가 포함된 모든 IAM 작업이 표시됩니다. 다음 작업 옆에 있는 확인란을 선택합니다.

    - CreateGroup
    - DeleteGroup
    - RemoveUserFromGroup
    - AttachGroupPolicy
    - DeleteGroupPolicy
    - DetachGroupPolicy
    - PutGroupPolicy
    - UpdateGroup

12. 리소스를 선택하여 정책에 대한 리소스를 지정합니다. 선택한 작업에 따라 group 리소스 유형이 표시됩니다. ARN 추가를 선택합니다. 리소스 내부의 경우 모든 계정 옵션을 선택합니다. 그룹 이름과 경로에서 사용자 그룹 이름 **AllUsers**를 입력합니다. 그런 다음 ARN 추가를 선택합니다.
13. 요청 조건 - 선택 사항을 선택한 다음, 다른 조건 추가를 선택합니다. 다음 값을 사용하여 양식 입력을 완료합니다.

    - 조건 키 — aws:username 선택
    - 한정어 - 기본값을 선택합니다.
    - 연산자 - StringNotEquals를 선택합니다.
    - 값 - **srodriguez** 를 입력한 다음 추가를 선택합니다. **mjackson**를 입력한 다음, 추가를 선택하여 다른 값을 추가합니다. **adesai**를 입력한 다음 조건 추가를 선택합니다.

    이 조건은 호출한 사용자가 목록에 포함되지 않은 경우 지정된 사용자 그룹 관리 작업 액세스가 거부됩니다. 이는 명시적으로 권한을 거부하므로 해당 사용자가 작업을 호출할 수 있도록 허용된 이전 블록을 무시합니다. 목록에 있는 사용자는 액세스가 거부되지 않으며 첫 번째 권한 블록의 권한이 부여되므로 사용자 그룹을 전체적으로 관리할 수 있습니다.

14. 마쳤으면 [Next]를 선택합니다.

> **Note**
>
> 언제든지 시각적 편집기 옵션과 JSON 편집기 옵션을 서로 전환할 수 있습니다. 그러나 변경을 적용하거나 시각적 편집기 옵션에서 다음을 선택한 경우 IAM은 시각적 편집기에 최

적화되도록 정책을 재구성할 수 있습니다. 자세한 내용은 정책 재구성 단원을 참조하십시오.

15. 검토 및 생성 페이지에서 정책 이름에 **`LimitAllUserGroupManagement`**를 입력합니다. 설명에 **`Allows all users read-only access to a specific user group, and allows only specific users access to make changes to the user group`**을 입력합니다. 이 정책에 정의된 권한을 검토하여 의도한 권한을 부여했는지 확인합니다. 그런 다음 정책 생성을 선택하여 새 정책을 저장합니다.
16. 사용자 그룹에 정책을 연결합니다. 자세한 내용은 IAM 자격 증명 권한 추가 및 제거 단원을 참조하십시오.

또는 이러한 예시 JSON 정책 문서를 사용하여 동일한 정책을 생성할 수 있습니다. 이 JSON 정책을 보려면 IAM: 특정 IAM 사용자가 프로그래밍 방식으로, 그리고 콘솔에서 그룹을 관리하도록 허용 섹션을 참조하세요. JSON 문서를 사용하여 정책을 생성하는 자세한 지침은 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

정책에 대한 액세스 제어

사용자가 AWS 관리형 정책을 적용하는 방식을 제어할 수 있습니다. 이렇게 하려면 이 정책을 모든 사용자에게 연결합니다. 이 작업에 사용자 그룹을 사용하는 것이 좋습니다.

예를 들어, 사용자가 새 IAM 사용자, 사용자 그룹 또는 역할에 IAMUserChangePassword 및 PowerUserAccess AWS 관리형 정책만 연결하도록 허용하는 정책을 생성할 수 있습니다.

고객 관리형 정책의 경우 이러한 정책을 생성, 업데이트 및 삭제할 수 있는 대상을 제어할 수 있습니다. 정책을 보안 주체 개체(사용자 그룹, 사용자 및 역할)에 연결하고 해당 개체에서 분리할 수 있는 대상을 제어할 수 있습니다. 또한 사용자가 어떤 정책을 어떤 주체에 연결하거나 분리할지 제어할 수 있습니다.

예를 들어 계정 관리자에게 정책을 생성, 업데이트 및 삭제할 권한을 부여할 수 있습니다. 그런 다음 팀 리더 또는 기타 제한된 관리자에게 제한된 관리자가 관리하는 보안 주체 개체에 이러한 정책을 연결하고 분리할 권한을 부여합니다.

자세한 정보는 다음 리소스 섹션을 참조하세요.

- 보안 주체에 연결할 수 있는 IAM 정책의 생성에 대해 자세히 알아보려면 고객 관리형 정책으로 사용자 지정 IAM 권한 정의 섹션을 참조하세요.
- 보안 주체에 IAM 정책을 연결하는 방법에 대해 자세히 알아보려면 IAM 자격 증명 권한 추가 및 제거 섹션을 참조하세요.

- 관리형 정책 사용을 제한하는 예시 정책을 보려면 IAM: IAM 사용자, 그룹 또는 역할에 적용 가능한 관리형 정책을 제한 섹션을 참조하세요.

고객 관리형 정책을 생성, 업데이트 및 삭제할 권한 제어

IAM 정책을 사용하여 AWS 계정에서 고객 관리형 정책을 생성, 업데이트 및 삭제할 수 있는 대상을 제어할 수 있습니다. 다음 목록에는 정책 또는 정책 버전을 생성, 업데이트 및 삭제하는 것과 직접적으로 관련된 API 작업이 포함되어 있습니다.

- CreatePolicy
- CreatePolicyVersion
- DeletePolicy
- DeletePolicyVersion
- SetDefaultPolicyVersion

위 목록의 API 작업은 IAM 정책을 사용하여 허용하거나 거부할 수 있는 작업, 즉 부여할 수 있는 권한에 해당합니다.

다음 예시 정책을 고려하세요. 사용자가 AWS 계정에서 모든 고객 관리형 정책의 기본 버전을 생성, 업데이트(즉, 새 정책 버전 생성), 삭제 및 설정하도록 허용합니다. 또한 이 정책 예에서는 사용자가 정책을 나열하고 가져오도록 허용합니다. 이 예시 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

Example 모든 정책의 기본 버전을 생성, 업데이트, 삭제, 나열, 가져오기 및 설정하도록 허용하는 정책 예

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:CreatePolicy",
      "iam:CreatePolicyVersion",
      "iam:DeletePolicy",
      "iam:DeletePolicyVersion",
      "iam:GetPolicy",
      "iam:GetPolicyVersion",
      "iam:ListPolicies",
      "iam:ListPolicyVersions",
```

```
      "iam:SetDefaultPolicyVersion"
    ],
    "Resource": "*"
  }
}
```

이러한 API 작업의 사용을 제한하는 정책을 생성하여 지정하는 관리형 정책에만 영향을 줄 수 있습니다. 예를 들어, 특정 고객 관리형 정책의 경우에만 사용자가 기본 버전을 설정하고 정책 버전을 삭제하도록 허용해야 할 수 있습니다. 이렇게 하려면 이러한 권한을 부여하는 정책의 `Resource` 요소에 정책 ARN을 지정합니다.

다음 예시는 사용자가 정책 버전을 삭제하고 기본 버전을 설정할 수 있는 정책을 보여줍니다. 이런 작업은 /TEAM-A/ 경로를 포함하는 고객 관리형 정책에만 허용됩니다. 고객 관리형 정책 ARN은 그 정책의 `Resource` 요소에 지정되어 있습니다. (이 예에서 ARN에는 경로와 와일드카드가 포함되어 있으므로 경로 /TEAM-A/를 포함하는 모든 고객 관리형 정책과 일치합니다.) 이 예시 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

고객 관리형 정책 이름에서 경로를 사용하는 방법에 대한 자세한 정보는 표시 이름 및 경로 섹션을 참조하세요.

Example 특정 정책의 경우에만 정책 버전 삭제와 기본 버전 설정을 허용하는 정책 예

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:DeletePolicyVersion",
      "iam:SetDefaultPolicyVersion"
    ],
    "Resource": "arn:aws:iam::account-id:policy/TEAM-A/*"
  }
}
```

관리형 정책을 연결 및 분리하는 권한 제어

IAM 정책을 사용하여 사용자가 특정 관리형 정책만 사용하도록 허용할 수도 있습니다. 사실상 사용자가 다른 보안 주체에 부여할 수 있는 권한을 제어할 수 있습니다.

다음은 보안 주체 개체에 관리형 정책을 연결하고 분리하는 것과 직접적으로 관련된 API 작업 목록입니다.

- AttachGroupPolicy
- AttachRolePolicy
- AttachUserPolicy
- DetachGroupPolicy
- DetachRolePolicy
- DetachUserPolicy

이러한 API 작업의 사용을 제한하는 정책을 생성하여 지정하는 특정 관리형 정책 및/또는 보안 주체 개체에만 영향을 줄 수 있습니다. 예를 들어, 사용자가 지정하는 관리형 정책에만 연결하도록 허용해야 할 수 있습니다. 또는 사용자가 지정하는 보안 주체 엔터티에만 관리형 정책을 연결하도록 허용해야 할 수 있습니다.

다음 정책 예에서는 사용자가 경로 /TEAM-A/를 포함하는 사용자 그룹 및 역할에만 관리형 정책을 연결하도록 허용합니다. 사용자 그룹 및 역할 ARN은 정책의 `Resource` 요소에서 지정됩니다. (이 예에서 ARN에는 경로와 와일드카드 문자가 포함되어 있으므로 경로 /TEAM-A/를 포함하는 모든 사용자 그룹 및 역할과 일치합니다.) 이 예시 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

Example 특정 사용자 그룹 또는 역할에만 관리형 정책 연결을 허용하는 정책 예

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:AttachGroupPolicy",
      "iam:AttachRolePolicy"
    ],
    "Resource": [
      "arn:aws:iam::account-id:group/TEAM-A/*",
      "arn:aws:iam::account-id:role/TEAM-A/*"
    ]
  }
}
```

앞 예에서 지정하는 정책에만 영향을 주도록 작업의 사용을 제한할 수 있습니다. 즉, 정책에 조건을 추가하여 사용자가 다른 보안 주체 엔터티에 연결하도록 허용할 권한을 제어할 수 있습니다.

다음 예에서 조건은 연결된 정책이 지정된 정책 가운데 하나와 일치하는 경우에만 `AttachGroupPolicy` 및 `AttachRolePolicy` 권한이 허용되도록 합니다. 이 조건은 `iam:PolicyARN` 조건 키를 사용하여 연결할 수 있는 정책을 결정합니다. 다음은 위의 예시를 확장한 예시 정책입니다. 사용자가 경로 /TEAM-A/ 경로를 포함하는 사용자 그룹 및 역할에만 /TEAM-A/ 경로를 포함하는 관리형 정책만 연결하도록 허용합니다. 이 예시 JSON 정책 문서를 사용하여 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:AttachGroupPolicy",
      "iam:AttachRolePolicy"
    ],
    "Resource": [
      "arn:aws:iam::account-id:group/TEAM-A/*",
      "arn:aws:iam::account-id:role/TEAM-A/*"
    ],
    "Condition": {"ArnLike":
      {"iam:PolicyARN": "arn:aws:iam::account-id:policy/TEAM-A/*"}
    }
  }
}
```

이 정책은 `ArnLike` 조건 연산자를 사용하지만 두 조건 연산자가 동일하게 작동하므로 `ArnEquals` 조건 연산자도 사용할 수 있습니다. `ArnLike` 및 `ArnEquals`에 대한 자세한 내용은 정책 요소 참조의 조건 유형에서 Amazon 리소스 이름(ARN) 조건 연산자 섹션을 참조하세요.

예를 들어, 지정하는 관리형 정책만 포함하도록 작업 사용을 제한할 수 있습니다. 이렇게 하려면 이러한 권한을 부여하는 정책의 `Condition` 요소에 정책 ARN을 지정합니다. 예를 들어, 고객 관리형 정책의 ARN을 지정하려면:

```
"Condition": {"ArnEquals":
  {"iam:PolicyARN": "arn:aws:iam::123456789012:policy/POLICY-NAME"}
}
```

AWS 관리형 정책의 `Condition` 요소에서도 정책의 ARN을 지정할 수 있습니다. AWS 관리형 정책의 ARN은 다음 예와 같이 정책 ARN에 계정 ID 대신 `aws`라는 특별한 별칭을 사용합니다.

```
"Condition": {"ArnEquals":
  {"iam:PolicyARN": "arn:aws:iam::aws:policy/AmazonEC2FullAccess"}
}
```

리소스에 대한 액세스 제어

자격 증명 기반 정책 또는 리소스 기반 정책을 사용하여 리소스에 대한 액세스를 제어할 수 있습니다. 자격 증명 기반 정책에서 자격 증명에 정책을 연결하고 자격 증명이 액세스할 수 있는 리소스를 지정합니다. 리소스 기반 정책에서 제어하려는 리소스에 정책을 연결합니다. 정책에서 해당 리소스에 액세스할 수 있는 보안 주체를 지정합니다. 두 정책 유형에 대한 자세한 정보는 자격 증명 기반 정책 및 리소스 기반 정책 섹션을 참조하세요.

자세한 정보는 다음 리소스 섹션을 참조하세요.

- 보안 주체에 연결할 수 있는 IAM 정책의 생성에 대해 자세히 알아보려면 고객 관리형 정책으로 사용자 지정 IAM 권한 정의 섹션을 참조하세요.
- 보안 주체에 IAM 정책을 연결하는 방법에 대해 자세히 알아보려면 IAM 자격 증명 권한 추가 및 제거 섹션을 참조하세요.
- Amazon S3는 해당 버킷의 리소스 기반 정책 사용을 지원합니다. 자세한 정보는 버킷 정책 예시 섹션을 참조하세요.

리소스 생성자에게 권한이 자동으로 부여되지는 않음

AWS 계정 루트 사용자 자격 증명을 사용하여 로그인한 경우 해당 계정에 속한 리소스에서 모든 작업을 수행할 수 있는 권한이 부여됩니다. 그러나 IAM 사용자의 경우에는 그렇지 않습니다. IAM 사용자는 리소스를 생성할 권한을 받을 수 있지만, 그러한 리소스에 대한 권한은 명시적으로 부여받은 권한으로 제한됩니다. 즉, IAM 역할과 같은 리소스를 생성했다는 이유만으로 해당 역할을 편집 또는 삭제할 권한이 자동으로 부여되지는 않습니다. 또한 사용자의 권한은 계정 소유자 또는 해당 권한을 관리할 권한이 있는 다른 사용자가 언제든지 취소할 수 있습니다.

특정 계정에서 보안 주체에 대한 액세스 제어

계정의 IAM 사용자에게 리소스에 대한 액세스 권한을 직접 부여할 수 있습니다. 다른 계정의 사용자가 리소스에 액세스할 필요가 있다면 IAM 역할을 생성합니다. 역할은 권한을 포함한 개체이지만 특정 사용자와 관련이 없습니다. 다른 계정의 사용자는 해당 역할을 수임하여 해당 역할에 할당된 권한에 따라 리소스에 액세스할 수 있습니다. 자세한 내용은 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 단원을 참조하십시오.

> **Note**
>
> 일부 서비스만이 자격 증명 기반 정책 및 리소스 기반 정책(예: Amazon S3, Amazon SNS, Amazon SQS)에 설명된 리소스 기반 정책을 지원합니다. 그런 서비스의 역할 사용 대안은 공유할 리소스(버킷, 주제 또는 대기열)에 정책을 연결하는 것입니다. 리소스 기반 정책은 리소스에 대한 액세스 허가를 받은 AWS 계정을 지정할 수 있습니다.

태그를 사용하여 IAM 사용자 및 역할에 대한 액세스 제어

다음 섹션의 정보를 사용하여 IAM 사용자 및 역할에 액세스할 수 있는 사람과 사용자 및 역할이 액세스할 수 있는 리소스를 제어합니다. 다른 IAM 리소스를 비롯하여, 다른 AWS 리소스에 대한 액세스 제어와 관련한 자세한 일반 정보 및 예는 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요.

> **Note**
>
> 태그 키 및 태그 키 값의 대소문자 구분에 대한 자세한 내용은 Case sensitivity을(를) 참조하세요.

태그는 IAM 리소스에 연결하거나 요청에 전달하거나 요청을 하는 보안 주체에 연결할 수 있습니다. IAM 사용자 또는 역할은 리소스 및 보안 주체 둘 다일 수 있습니다. 예를 들어 사용자가 사용자 그룹을 나열하도록 허용하는 정책을 작성할 수 있습니다. 이 작업은 요청을 하는 사용자(보안 주체)가 보려는 사용자와 동일한 `project=blue` 태그를 갖고 있는 경우에만 허용됩니다. 이 예에서 사용자는 동일한 프로젝트에서 작업하는 동안 자신을 비롯하여 모든 사용자에 대한 그룹 구성원 자격을 볼 수 있습니다.

태그를 기반으로 액세스를 제어하려면 정책의 조건 요소에 태그 정보를 제공하세요. IAM 정책을 생성할 때 IAM 태그 및 연관된 태그 조건 키를 사용하여 다음 중 하나에 대한 액세스를 제어할 수 있습니다.

- 리소스 - 태그를 기반으로 사용자 또는 역할 리소스에 대한 액세스를 제어합니다. 이를 위해 aws:ResourceTag/***key-name*** 조건 키를 사용하여 리소스에 연결해야 하는 태그 키 값 페어를 지정합니다. 자세한 내용은 AWS 리소스에 대한 액세스 제어 단원을 참조하십시오.
- 요청 – IAM 요청에 어떤 태그가 전달될 수 있는지를 제어합니다. 이를 수행하려면 aws:RequestTag/***key-name*** 조건 키를 사용하여 어떤 태그를 IAM 사용자 또는 역할에서 추가, 변경 또는 제거할 수 있는지 지정합니다. 이 키는 IAM 리소스 및 기타 AWS 리소스에서 동일한 방식으로 사용됩니다. 자세한 내용은 AWS 요청 중 액세스 제어 단원을 참조하십시오.

- 보안 주체 - 요청을 하는 사람(보안 주체)이 자신의 IAM 사용자 또는 역할에 연결된 태그를 기반으로 수행할 수 있는 권한을 제어합니다. 이를 위해 aws:PrincipalTag/***key-name*** 조건 키를 사용하여 요청이 허용되기 전에 IAM 사용자 또는 역할에 연결해야 하는 태그를 지정합니다.
- 권한 부여 프로세스의 일부 - aws:TagKeys 조건 키를 사용하여 특정 태그 키를 요청 또는 보안 주체에서 사용할 수 있는지 여부를 제어합니다. 이 경우 키 값은 중요하지 않습니다. 이 키는 IAM 및 기타 AWS 서비스에 대해서도 유사하게 동작합니다. 그러나 IAM에서 사용자를 태그 지정하면 보안 주체가 모든 서비스에 대한 요청을 할 수 있는지 여부도 제어할 수 있습니다. 자세한 내용은 태그 키를 기반으로 액세스 제어 단원을 참조하십시오.

시각적 편집기를 사용하거나 JSON을 사용하거나 기존 관리형 정책을 가져와서 IAM 정책을 생성할 수 있습니다. 세부 정보는 고객 관리형 정책으로 사용자 지정 IAM 권한 정의을 참조하세요.

> **Note**
>
> IAM 역할을 맡거나 사용자를 연동할 때 세션 태그를 전달할 수도 있습니다. 세션 중에만 유효합니다.

IAM 보안 주체에 대한 액세스 제어

보안 주체가 자신의 자격 증명에 연결된 태그를 기반으로 수행할 수 있는 권한을 제어할 수 있습니다.

이 예제는 이 계정의 모든 사용자가 동일한 프로젝트에서 작업하는 동안 자신을 포함한 모든 사용자의 그룹 멤버십을 볼 수 있도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 작업은 사용자의 리소스 태그와 보안 주체의 태그가 태그 키 `project`와 동일한 값을 가질 경우에만 허용됩니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "iam:ListGroupsForUser",
            "Resource": "arn:aws:iam::111222333444:user/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/project":
 "${aws:PrincipalTag/project}"}
```

```
            }
        }]
}
```

태그 키를 기반으로 액세스 제어

IAM 정책에서 태그를 사용하여 요청 또는 보안 주체에 특정 태그 키를 사용할 수 있는지 여부를 제어할 수 있습니다.

이 예제는 사용자의 `temporary` 키를 포함한 태그만 제거할 수 있는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "iam:UntagUser",
        "Resource": "*",
        "Condition": {"ForAllValues:StringEquals": {"aws:TagKeys": ["temporary"]}}
    }]
}
```

태그를 사용한 AWS 리소스 액세스 제어

태그를 사용하여, IAM 리소스를 비롯한 태깅을 지원하는 AWS 리소스에 대한 액세스를 제어할 수 있습니다. IAM 사용자 및 역할을 태깅하여 액세스할 수 있는 권한을 제어할 수 있습니다. IAM 사용자 및 역할을 태그 지정하는 방법을 알아보려면 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요. 또한 고객 관리형 정책, IAM 자격 증명 공급자, 인스턴스 프로파일, 서버 인증서, 가상 MFA 디바이스 등의 IAM 리소스에 대한 액세스를 제어할 수 있습니다. 보안 주체 태그가 있는 IAM 역할이 일치하는 태그가 있는 리소스에 액세스할 수 있도록 허용하는 정책을 만들고 테스트하는 자습서를 보려면 IAM 튜토리얼: 태그를 기반으로 AWS 리소스에 액세스할 수 있는 권한 정의 섹션을 참조하세요. 다음 섹션의 정보를 사용하여 IAM 사용자 또는 역할을 태깅하지 않고 IAM 리소스를 비롯한 다른 AWS 리소스에 대한 액세스를 제어합니다.

태그를 사용하여 AWS 리소스에 대한 액세스를 제어하기 전에 AWS의 액세스 허용 방식을 이해해야 합니다. 또한 AWS는 리소스의 컬렉션으로 구성되어 있습니다. Amazon EC2 인스턴스는 리소스입니다. Amazon S3 버킷은 리소스입니다. AWS API, AWS CLI 또는 AWS Management Console을 사용하여 작업(예: Amazon S3에서 버킷 생성)을 수행할 수 있습니다. 그렇게 하면 해당 작업에 대한 요청을

보냅니다. 이 요청은 작업, 리소스, 보안 주체 엔터티(사용자 또는 역할), 보안 주체 계정 및 필요한 요청 정보를 지정합니다. 이러한 모든 정보는 콘텍스트를 제공합니다.

그런 다음 AWS은 사용자(보안 주체 엔터티)가 지정된 리소스에 대해 지정된 작업을 수행할 수 있도록 인증(로그인) 및 권한 부여(권한 있음)되었는지 확인합니다. 권한을 부여하는 동안 AWS는 요청 콘텍스트에 적용되는 모든 정책을 확인합니다. 대부분의 정책은 AWS에 JSON 문서로 저장되며 보안 주체 엔터티에 대한 권한을 지정합니다. 정책 유형 및 활용에 대한 자세한 정보는 IAM의 정책 및 권한 섹션을 참조하세요.

AWS는 정책이 요청의 각 부분을 허용한 경우에만 요청에 권한을 부여합니다. 다이어그램을 보고 IAM 인프라에 대해 자세히 알아보려면 IAM 작동 방식 섹션을 참조하세요. IAM 요청이 허용되는지 여부를 결정하는 방법에 대한 자세한 내용은 정책 평가 로직 섹션을 참조하세요.

태그는 이 프로세스에서 고려해야 할 또 다른 요소인데, 리소스에 태그가 연결되거나 태그 지정을 지원하는 서비스에 대한 요청에 전달될 수 있기 때문입니다. 태그를 기반으로 액세스를 제어하려면 정책의 조건 요소에 태그 정보를 제공하세요. AWS 서비스에서 태그를 사용한 액세스 제어를 지원하는지 여부를 알아보려면 AWS IAM으로 작업하는 서비스 섹션을 참조하고, ABAC 열이 예인 서비스를 찾아보세요. 서비스의 이름을 선택하여 해당 서비스에 대한 권한 부여 및 액세스 제어 문서를 봅니다.

그러면 리소스의 태그를 기반으로 리소스에 대한 액세스를 허용하거나 거부하는 IAM 정책을 생성할 수 있습니다. 해당 정책에서는 태그 조건 키를 사용하여 다음 중 하나에 대한 액세스를 제어할 수 있습니다.

- 리소스 - 리소스에 대한 태그를 기반으로 AWS 서비스 리소스에 대한 액세스를 제어합니다. 이를 수행하려면 ResourceTag/***key-name*** 조건 키를 사용하여 리소스에 연결된 태그를 기반으로 리소스에 대한 액세스를 허용할지 여부를 결정합니다.
- 요청 - 어떤 태그가 요청에 전달될 수 있는지 제어합니다. 이를 수행하려면 aws:RequestTag/***key-name*** 조건 키를 사용하여 AWS 리소스에 태그를 지정하는 요청에서 어떤 태그 키 값 페어를 전달할 수 있는지 지정합니다.
- 권한 부여 프로세스의 일부 - aws:TagKeys 조건 키를 사용하여 특정 태그 키를 요청에서 사용할 수 있는지 여부를 제어합니다.

JSON을 사용하거나 기존 관리형 정책을 가져와서 시각적으로 IAM 정책을 생성할 수 있습니다. 세부 정보는 고객 관리형 정책으로 사용자 지정 IAM 권한 정의을 참조하세요.

> **Note**
>
> 일부 서비스에서는 사용자가 리소스를 생성하는 작업을 사용할 권한이 있는 경우 리소스 생성 시 태그를 지정할 수 있도록 허용합니다.

AWS 리소스에 대한 액세스 제어

IAM 정책의 조건을 사용하여 해당 리소스의 태그를 기반으로 AWS 리소스에 대한 액세스를 제어할 수 있습니다. 전역 `aws:ResourceTag/tag-key` 조건 키 또는 서비스별 키를 사용하여 이 작업을 수행할 수 있습니다. 일부 서비스는 이 키의 서비스별 버전만 지원하며 전역 버전은 지원하지 않습니다.

> **Warning**
>
> 역할에 태그를 지정한 다음 `iam:PassRole` 작업과 함께 정책의 `ResourceTag` 조건 키를 사용하는 것으로 역할을 전달할 수 있는 사람을 제어하려고 하지 마십시오. 이 접근법은 신뢰할 수 있는 결과를 가져오지 못합니다. 서비스로 역할을 전달하는 데 필요한 권한에 대한 자세한 내용은 사용자에게 AWS 서비스에 역할을 전달할 권한 부여 섹션을 참조하세요.

이 예제는 Amazon EC2 인스턴스의 시작 또는 중지를 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이러한 작업은 인스턴스 태그 `Owner`가 사용자 이름 값을 포함한 경우에만 허용됩니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
            }
        },
        {
            "Effect": "Allow",
```

```
            "Action": "ec2:DescribeInstances",
            "Resource": "*"
        }
    ]
}
```

이 정책을 계정의 IAM 사용자에게 연결할 수 있습니다. 이름이 `richard-roe`인 사용자가 Amazon EC2 인스턴스를 시작하려 하는 경우 인스턴스에 `Owner=richard-roe` 또는 `owner=richard-roe` 태그가 지정되어야 합니다. 그렇지 않은 경우 액세스가 거부됩니다. 태그 키 `Owner`는 `Owner` 및 `owner` 모두와 일치하는데, 조건 키가 대/소문자를 구분하지 않기 때문입니다. 자세한 내용은 IAM JSON 정책 요소: Condition 단원을 참조하십시오.

이 예제는 리소스 ARN에 `team` 보안 주체 태그를 사용하는 ID 기반 정책을 생성하는 방법을 보여줍니다. 정책은 Amazon Simple Queue Service 대기열을 삭제할 수 있는 권한을 부여하지만, 팀 이름으로 시작하고 뒤에 `-queue`이 붙은 대기열 이름에만 해당됩니다. 예를 들어, `qa`가 `team` 보안 주체 태그의 팀 이름인 경우의 `qa-queue`입니다.

```
{
      "Version": "2012-10-17",
      "Statement": {
        "Sid": "AllQueueActions",
        "Effect": "Allow",
        "Action": "sqs:DeleteQueue",
        "Resource": "arn:aws:sqs:us-east-2::${aws:PrincipalTag/team}-queue"
      }
}
```

AWS 요청 중 액세스 제어

IAM 정책의 조건을 사용하여 AWS 리소스에 태그를 적용하는 요청에서 어떤 태그 키 값 페어를 전달할 수 있는지를 제어할 수 있습니다.

이 예제는 태그를 인스턴스에 연결하는 Amazon EC2 `CreateTags` 작업 사용을 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 태그에 `environment` 키 및 `preprod` 또는 `production` 값이 포함된 경우에만 태그를 연결할 수 있습니다. 원하는 경우 `ForAllValues` 변경자를 `aws:TagKeys` 조건 키와 함께 사용하여 요청에서 키 `environment`만 허용됨을 표시할 수 있습니다. 이를 통해 사용자가 `environment` 대신 `Environment`를 실수로 사용하는 것과 같이 다른 키를 포함시키는 것을 방지합니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "ec2:CreateTags",
        "Resource": "arn:aws:ec2:*:*:instance/*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/environment": [
                    "preprod",
                    "production"
                ]
            },
            "ForAllValues:StringEquals": {"aws:TagKeys": "environment"}
        }
    }
}
```

태그 키를 기반으로 액세스 제어

IAM 정책에서 조건을 사용하여 요청에 특정 태그 키를 사용할 수 있는지 여부를 제어할 수 있습니다.

정책을 사용하여 태그를 사용한 액세스를 제어할 때 aws:TagKeys 조건 키를 사용하는 것이 좋습니다. 태그를 지원하는 AWS 서비스를 통해 대소문자만 다른 여러 태그 키 이름을 생성할 수 있습니다(예: Amazon EC2 인스턴스에 stack=production 및 Stack=test 태그 지정). 정책 조건에서 키 이름은 대/소문자를 구분하지 않습니다. 따라서 정책의 조건 요소에서 "aws:ResourceTag/TagKey1": "Value1" 지정을 완료한 경우 조건은 이름이 TagKey1 또는 tagkey1인 리소스 태그 키와 일치하지만 두 가지 모두와 일치하지는 않습니다. 대소문자만 다른 키를 포함한 중복 태그를 방지하려면 aws:TagKeys 조건을 사용하여 사용자가 적용할 수 있는 태그 키를 정의하거나 AWS Organizations과(와) 함께 사용할 수 있는 태그 정책을 사용합니다. 자세한 내용은 Organizations 사용 설명서의 태그 정책을 참조하세요.

이 예제에서는 Secrets Manager 비밀 생성 태그 지정을 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다(태그 키가 environment 또는 cost-center인 경우만). Null 조건은 요청에 태그가 없는 경우 조건이 false로 평가되도록 합니다.

```
{
        "Effect": "Allow",
        "Action": [
            "secretsmanager:CreateSecret",
            "secretsmanager:TagResource"
        ],
        "Resource": "*",
```

```
        "Condition": {
            "Null": {
                "aws:TagKeys": "false"
            },
            "ForAllValues:StringEquals": {
                "aws:TagKeys": [
                    "environment",
                    "cost-center"
                ]
            }
        }
}
```

IAM의 크로스 계정 리소스 액세스

일부 AWS 서비스의 경우, IAM을 사용하여 리소스에 대한 크로스 계정 액세스 권한을 부여할 수 있습니다. 이렇게 하려면 공유하려는 리소스에 리소스 정책을 직접 연결하거나 역할을 프록시로 사용하면 됩니다.

리소스를 직접 공유하려면, 공유하려는 리소스가 반드시 리소스 기반 정책을 지원해야 합니다. 역할의 ID 기반 정책과 달리 리소스 기반 정책은 해당 리소스에 액세스할 수 있는 사용자(보안 주체)를 지정합니다.

리소스 기반 정책을 지원하지 않는 다른 계정의 리소스에 액세스하려는 경우, 역할을 프록시로 사용합니다.

이러한 정책 유형 간의 차이에 대한 자세한 내용은 자격 증명 기반 정책 및 리소스 기반 정책 섹션을 참조하세요.

> **Note**
>
> IAM 역할 및 리소스 기반 정책은 단일 파티션 내에서만 계정 간에 액세스 권한을 위임합니다. 미국 서부(캘리포니아 북부)의 표준 aws 파티션에 계정이 있는 경우를 예로 들어보겠습니다. 중국의 aws-cn 파티션에도 계정이 있습니다. 이 경우 중국의 계정에서 리소스 기반 정책을 사용하여 AWS 표준 계정의 사용자에 대한 액세스를 허용할 수 없습니다.

역할을 사용한 크로스 계정 액세스

모든 AWS 서비스에서 리소스 기반 정책을 지원하는 것은 아닙니다. 이러한 서비스의 경우 여러 서비스에 대한 크로스 계정 액세스를 제공할 때 크로스 계정 IAM 역할을 사용하여 권한 관리를 중앙 집중화

할 수 있습니다. 크로스 계정 IAM 역할은 다른 AWS 계정의 IAM 보안 주체가 역할을 수임할 수 있도록 하는 신뢰 정책을 포함하는 IAM 역할입니다. 간단히 설명하자면, 특정 권한을 다른 AWS 계정에 위임하는 역할을 단일 AWS 계정에 생성할 수 있습니다.

IAM 자격 증명에 정책을 연결하는 방법에 대한 자세한 내용은 IAM 정책 관리 섹션을 참조하세요.

> **Note**
>
> 보안 주체가 역할의 권한을 일시적으로 사용하기 위해 특정 역할로 전환하면, 원래 권한을 포기하고 수임하는 역할에 할당된 권한을 갖게 됩니다.

고객 계정에 액세스해야 하는 APN 파트너 소프트웨어에 적용되는 전체 프로세스를 살펴보겠습니다.

1. 고객이 자신의 계정에 APN 파트너가 요구하는 Amazon S3 리소스에 대한 액세스를 허용하는 정책을 포함하여 IAM 역할을 생성합니다. 이 예에서 역할의 이름은 `APNPartner`입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::bucket-name"
            ]
        }
    ]
}
```

2. 이후 고객은 `APNPartner` 역할에 대한 신뢰 정책에 APN 파트너의 AWS 계정 ID를 제공하여 파트너의 AWS 계정에서 역할을 수임할 수 있도록 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::APN-account-ID:role/APN-user-name"
            },
            "Action": "sts:AssumeRole"
```

```
        }
    ]
}
```

3. 고객이 역할의 Amazon 리소스 이름(ARN)을 APN 파트너에게 제공합니다. ARN은 역할의 정규화된 이름입니다.

```
arn:aws:iam::APN-ACCOUNT-ID:role/APNPartner
```

> **Note**
>
> 멀티 테넌트 상황에서는 외부 ID를 사용하는 것이 좋습니다. 세부 정보는 타사가 소유한 AWS 계정에 액세스을 참조하세요.

4. APN 파트너의 소프트웨어가 고객 계정에 액세스해야 하는 경우, 해당 소프트웨어는 고객 계정 내 역할의 ARN을 사용하여 AWS Security Token Service에서 AssumeRole API를 호출합니다. STS는 소프트웨어가 작업을 수행할 수 있도록 하는 임시 AWS 보안 인증 정보를 반환합니다.

역할을 사용하여 크로스 계정 액세스 권한을 부여하는 또 다른 예는 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 섹션을 참조하세요. 튜토리얼: IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임에 접속해도 됩니다.

리소스 기반 정책을 사용한 크로스 계정 액세스

계정에서 리소스 기반 정책을 사용하여 다른 계정을 통해 리소스에 액세스할 경우, 보안 주체는 여전히 신뢰할 수 있는 계정에서 작업을 할 수 있고, 역할 권한을 수신하기 위해 자신의 권한을 포기할 필요가 없습니다. 즉, 보안 주체는 신뢰하는 계정의 리소스에 대한 액세스 권한을 유지하면서 신뢰할 수 있는 계정의 리소스에 계속 액세스할 수 있습니다. 다른 계정의 공유 리소스로 정보를 복사하거나 공유 리소스의 정보를 복사하는 등의 작업에서 이는 특히 유용합니다.

리소스 기반 정책에서 지정할 수 있는 보안 주체에는 계정, IAM 사용자, 페더레이션 사용자, IAM 역할, 수임된 역할 세션 또는 AWS 서비스가 포함됩니다. 자세한 내용은 보안 주체 지정을 참조하세요.

해당 신뢰 영역(신뢰할 수 있는 조직 또는 계정) 외의 계정 내 보안 주체가 역할을 수임하는 권한이 있는지 자세히 알고 싶다면, 외부 엔티티와 공유되는 리소스 식별을 참조하세요.

다음 목록에는 리소스 기반 정책을 지원하는 일부 AWS 서비스가 나와 있습니다. 보안 주체 대신 리소스에 권한 정책을 연결할 수 있도록 지원하는 AWS 서비스는 늘어나고 있습니다. 해당 서비스의 전체

목록은 AWS IAM으로 작업하는 서비스 섹션을 참조하고 리소스 기반 열의 값이 예인 서비스를 찾아보세요.

- Amazon S3 버킷 - 정책은 버킷과 연결되지만, 버킷과 그 안에 포함된 객체에 대한 액세스를 모두 제어합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon S3의 버킷 정책을 참조하세요. 일부의 경우, Amazon S3의 크로스 계정 액세스를 위한 역할을 사용하는 것이 최선일 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 연습 예제를 참조하세요.
- Amazon Simple Notification Service(SNS) 주제 - 자세한 내용은 Amazon Simple Notification Service 개발자 안내서에서 Amazon SNS 액세스 제어 사례를 참조하세요.
- Amazon Simple Queue Service(Amazon SQS) 대기열 - 자세한 내용은 Amazon Simple Queue Service 개발자 안내서의 부록: 액세스 정책 언어를 참조하세요.

AWS 권한 위임을 위한 리소스 기반 정책

리소스가 계정의 보안 주체에 권한을 부여하는 경우 이러한 권한을 특정 IAM 자격 증명에 위임할 수 있습니다. 자격 증명은 사용자, 사용자 그룹 또는 계정의 역할입니다. 자격 증명에 정책을 연결하여 권한을 위임합니다. 리소스 소유 계정에서 허용하는 최대 권한을 부여할 수 있습니다.

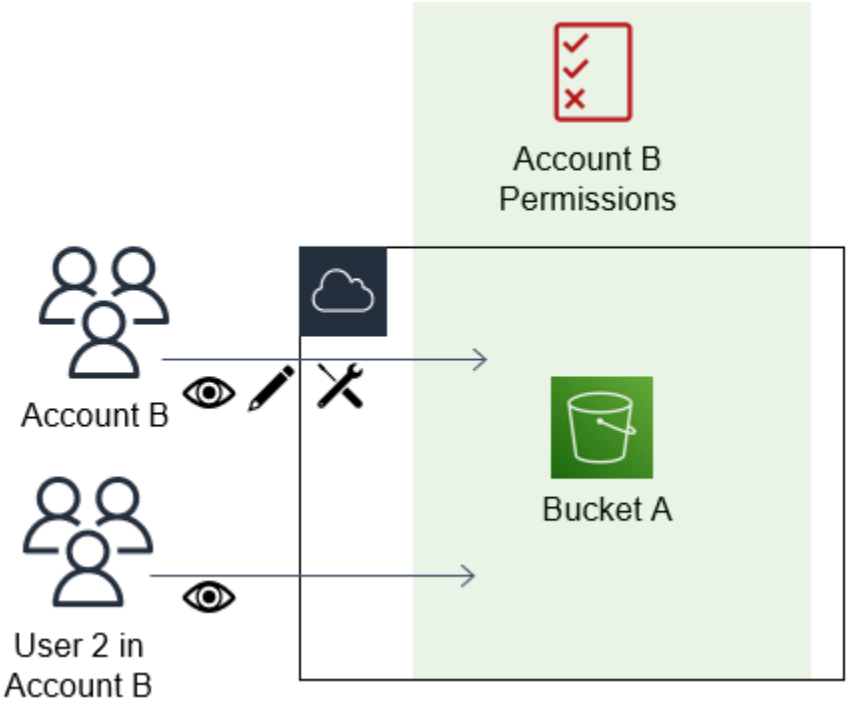
> **Important**
>
> 크로스 계정 액세스에서 보안 주체는 ID 정책 및 리소스 기반 정책에 `Allow` 항목이 있어야 합니다.

리소스 기반 정책에서는 계정의 모든 보안 주체에게 리소스에 대한 전체 관리 액세스 권한을 허용한다고 가정합니다. 이제 AWS 계정의 보안 주체에게 전체 액세스 권한, 읽기 전용 액세스 권한 또는 기타 부분적 액세스 권한을 위임할 수 있습니다. 또는 리소스 기반 정책에서 목록 액세스 권한만 허용하는 경우에는 목록 액세스 권한만 위임할 수 있습니다. 계정에 부여된 것보다 더 많은 권한을 위임하려고 해도 보안 주체는 목록 액세스 권한만 갖게 됩니다.

이러한 결정을 내리는 방법에 대한 자세한 내용은 계정 내에서 요청 허용 여부 결정을 참조하세요.

> **Note**
>
> IAM 역할 및 리소스 기반 정책은 단일 파티션 내에서만 계정 간에 액세스 권한을 위임합니다. 예를 들어 표준 aws 파티션의 계정과 aws-cn 파티션의 계정 간에 크로스 계정 액세스를 추가할 수 없습니다.

예를 들어 AccountA 및 AccountB를 관리한다고 가정합니다. AccountA에 BucketA라는 Amazon S3 버킷이 있습니다.



1. AccountB의 모든 보안 주체에게 버킷의 객체에 대한 전체 액세스 권한을 허용하는 리소스 기반 정책을 BucketA에 연결합니다. 해당 버킷의 모든 객체를 생성, 읽기 또는 삭제할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PrincipalAccess",
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::AccountB:root"},
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::BucketA/*"
            }
```

```
        ]
}
```

AccountA는 리소스 기반 정책에서 보안 주체로서 AccountB를 명명하여 AccountB에게 BucketA에 대한 전체 액세스 권한을 제공합니다. 그 결과 AccountB는 계정 BucketA에 대해 모든 작업을 수행할 권한이 있으며, AccountB 관리자는 AccountB에 속한 사용자에게 액세스 권한을 위임할 수 있습니다.

AccountB 루트 사용자는 계정에 부여된 모든 권한을 가집니다. 따라서 루트 사용자는 BucketA에 대한 전체 액세스 권한을 가집니다.

2. AccountB에서 User2라는 IAM 사용자에게 정책을 연결합니다. 이 정책은 사용자에게 BucketA의 객체에 대한 읽기 전용 액세스를 허용합니다. 즉, User2는 객체를 볼 수 있지만 객체를 생성, 편집 또는 삭제할 수는 없습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect" : "Allow",
            "Action" : [
                "s3:Get*",
                "s3:List*" ],
                "Resource" : "arn:aws:s3:::BucketA/*"
        }
    ]
}
```

AccountB가 위임할 수 있는 최대 액세스 수준은 계정에 부여된 액세스 수준입니다. 이 경우 리소스 기반 정책에서는 AccountB에 대한 전체 액세스 권한을 부여하지만 User2에는 읽기 전용 액세스 권한만 부여됩니다.

AccountB 관리자는 User1에게 액세스 권한을 부여하지 않습니다. 기본적으로 사용자는 명시적으로 부여된 권한을 제외하고는 어떤 권한도 없으므로 User1은 BucketA에 대한 액세스 권한이 없습니다.

IAM은 보안 주체가 요청을 할 때 보안 주체의 권한을 평가합니다. 와일드카드(*)를 사용하여 사용자에게 리소스에 대한 전체 액세스 권한을 부여하면 보안 주체는 AWS 계정이 액세스 권한을 가지고 있는 모든 리소스에 액세스할 수 있습니다. 사용자 정책을 생성한 이후에 추가하거나 액세스 권한을 획득한 리소스의 경우에도 마찬가지입니다.

앞의 예에서 AccountB가 모든 계정의 모든 리소스에 대한 전체 액세스 권한을 허용하는 정책을 User2에 연결했다면 User2는 AccountB에 액세스 권한이 있는 모든 리소스에 자동으로 액세스할 수 있었을 것입니다. 여기에는 BucketA 액세스 권한과 AccountA의 리소스 기반 정책에서 부여한 다른 리소스에 대한 액세스 권한이 포함됩니다 .

애플리케이션 및 서비스에 대한 액세스 권한 부여 등, 역할의 복잡한 사용 사례에 대한 자세한 내용은 IAM 역할 관련 일반 시나리오 섹션을 참조하세요.

> **Important**
>
> 신뢰 관계가 설정된 엔터티에만 액세스 권한을 부여하고 필요한 최소 수준의 액세스 권한만 부여합니다. 신뢰할 수 있는 엔터티가 다른 AWS 계정인 경우에는 어떠한 IAM 보안 주체에게든 리소스에 대한 액세스 권한을 부여할 수 있습니다. 신뢰하는 AWS 계정은 권한이 부여된 액세스 범위 내에서만 권한을 위임할 수 있으며, 계정에 부여된 권한보다 더 많은 액세스 권한을 위임할 수 없습니다.

권한, 정책 및 정책 작성에 사용하는 권한 정책 언어에 대한 자세한 내용은 AWS리소스에 대한 액세스 관리 섹션을 참조하세요.

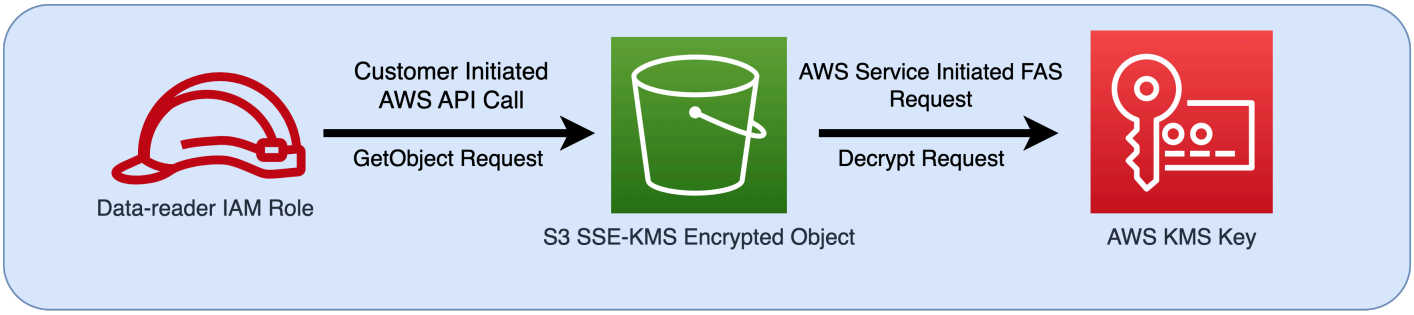
전달 액세스 세션

FAS(전달 액세스 세션)는 AWS 서비스가 사용자를 대신하여 요청할 때 ID, 권한 및 세션 속성을 전달하는 데 사용하기 위해 AWS 서비스에서 사용하는 IAM 기술입니다. FAS는 AWS 서비스를 직접적으로 호출하는 ID의 권한을 AWS 서비스의 ID와 결합하여 사용해 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 이후 IAM 보안 주체를 대신하여 AWS 서비스에 이루어집니다. FAS 요청이 이루어진 경우:

- IAM 보안 주체로부터 최초 요청을 받는 서비스는 IAM 보안 주체의 권한을 확인합니다.
- 후속 FAS 요청을 받는 서비스는 동일한 IAM 보안 주체의 권한도 확인합니다.

예를 들어, SSE-KMS를 사용하여 암호화되었을 때 Amazon S3에서 객체 암호 해독을 위해 AWS Key Management Service 직접 호출 시 FAS가 사용됩니다. SSE-KMS 암호화 객체를 다운로드할 때 data-reader라는 역할은 Amazon S3에 대해 객체에서 GetObject를 직접 호출하고, AWS KMS를 직접 호출하지는 않습니다. GetObject 요청을 수신하고 data-reader를 승인한 후 Amazon S3는 Amazon S3 객체의 암호를 해독하기 위해 AWS KMS에 FAS 요청을 보냅니다. KMS가 FAS 요청을 받으면 역할의 권한을 확인하고 data-reader에 KMS 키에 대한 올바른 권한을 가진 경우에만 암호 해독 요청을 승인합

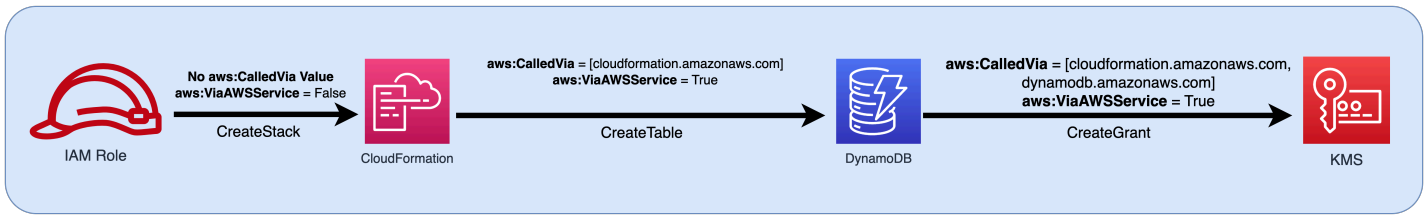
니다. Amazon S3와 AWS KMS 모두에 대한 요청은 역할의 권한을 사용하여 승인되고, data-reader가 Amazon S3 객체와 AWS KMS 키 모두에 대한 권한을 보유한 경우에만 성공합니다.



> **Note**
>
> FAS 요청을 받은 서비스에 의해 추가 FAS 요청이 이루어질 수 있습니다. 이 경우 요청 보안 주체에 FAS에서 호출한 모든 서비스에 대한 권한이 있어야 합니다.

FAS 요청 및 IAM 정책 조건

FAS 요청이 이루어졌을 때 aws:CalledVia, aws:CalledViaFirst 및 aws:CalledViaLast 조건 키는 FAS 직접 호출을 시작한 서비스의 서비스 보안 주체로 채워집니다. aws:ViaAWSService 조건 키 값은 FAS 요청이 이루어질 때마다 `true`로 설정됩니다. 다음 다이어그램에서 CloudFormation에 대한 요청에는 직접적으로 `aws:CalledVia` 또는 `aws:ViaAWSService` 조건 키가 설정되어 있지 않습니다. CloudFormation 및 DynamoDB에서 역할을 대신하여 다운스트림 FAS 요청을 보내면 이러한 조건 키의 값이 채워집니다.



조건 키 테스트 소스 IP 주소 또는 소스 VPC가 포함된 거부 정책 설명에 의해 FAS 요청이 거부될 때 FAS 요청을 허용하려면 조건 키를 사용하여 거부 정책에서 FAS 요청에 예외를 제공해야 합니다. 이는 `aws:ViaAWSService` 조건 키를 사용하여 모든 FAS 요청에 대해 가능합니다. 특정 AWS 서비스에서만 FAS 요청을 허용하려면 `aws:CalledVia`를 사용합니다.

> **Important**
>
> VPC 엔드포인트를 통해 최초 요청이 이루어진 후 FAS 요청이 이루어지면 최초 요청의 aws:SourceVpce, aws:SourceVpc, aws:VpcSourceIp에 대한 조건 키 값은 FAS 요청에서 사용되지 않습니다. aws:VPCSourceIP 또는 aws:SourceVPCE를 사용하여 조건부 액세스 권한을 부여하는 정책을 작성할 때 aws:ViaAWSService 또는 aws:CalledVia를 사용하여 FAS 요청도 허용해야 합니다. 퍼블릭 AWS 서비스 엔드포인트에서 최초 요청을 수신한 후 FAS 요청이 이루어지면 후속 FAS 요청은 동일한 aws:SourceIP 조건 키 값으로 이루어집니다.

예: VPC에서 또는 FAS를 사용하여 Amazon S3 액세스 허용

다음 IAM 정책 예시에서 Amazon S3 GetObject 및 Athena 요청은 *example_vpc*에 연결된 VPC 엔드포인트에서 시작되거나 Athena에서 이루어진 FAS 요청인 경우에만 허용됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "OnlyAllowMyIPs",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject*",
        "athena:StartQueryExecution",
        "athena:GetQueryResults",
        "athena:GetWorkGroup",
        "athena:StopQueryExecution",
        "athena:GetQueryExecution"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceVPC": [
            "example_vpc"
          ]
        }
      }
    },
    {
      "Sid": "OnlyAllowFAS",
```

```
      "Effect": "Allow",
      "Action": [
        "s3:GetObject*"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": "athena.amazonaws.com"
        }
      }
    }
  ]
}
```

조건 키를 사용하여 FAS 액세스를 허용하는 추가 예시는 데이터 경계 예시 정책 리포지토리를 참조하세요.

IAM 자격 증명 기반 정책의 예

정책은 자격 증명이나 리소스와 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. AWS는 IAM 보안 주체(사용자 또는 역할)가 요청을 보낼 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되는지 또는 거부되는지를 결정합니다. 대부분의 정책은 IAM 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결되는 JSON 문서로서 AWS에 저장됩니다. 자격 증명 기반 정책에는 AWS 관리형 정책, 고객 관리형 정책 및 인라인 정책이 포함됩니다. 이러한 예제 JSON 정책 문서를 사용하여 IAM 정책을 생성하는 방법에 대해 자세히 알아보려면 the section called “JSON 편집기를 사용하여 정책 생성” 섹션을 참조하세요.

기본적으로 모든 요청이 거부되기 때문에 ID가 액세스하려는 서비스, 작업 및 리소스에 대한 액세스 권한을 제공해야 합니다. 또한 IAM 콘솔에서 지정된 작업을 완료하기 위해 액세스를 허용하려면 추가 권한을 제공해야 합니다.

다음의 정책 라이브러리가 IAM ID에 대한 권한을 정의하는 데 도움이 될 수 있습니다. 필요로 하는 정책을 찾은 다음에 View this policy(이 정책 보기)를 선택하여 해당 정책의 JSON을 확인합니다. JSON 정책 문서를 자체 정책의 템플릿으로 활용할 수 있습니다.

> **Note**
>
> 이 참조 설명에 포함시킬 정책을 제출하고자 하는 경우 이 페이지의 하단에 있는 의견 버튼을 사용합니다.

AWS 정책의 예

- 특정 날짜 범위 동안 액세스를 허용합니다. (이 정책 보기.)
- AWS 리전 활성화 및 비활성화를 허용합니다. (이 정책 보기.)
- MFA 인증 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용합니다. (이 정책 보기.)
- 특정 날짜 범위 동안 MFA를 사용하는 경우 특정 액세스를 허용합니다. (이 정책 보기.)
- 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용합니다. (이 정책 보기.)
- 사용자가 보안 인증 페이지에서 자신의 MFA 디바이스를 관리할 수 있도록 허용합니다. (이 정책 보기.)
- 사용자가 보안 인증 페이지에서 자신의 암호를 관리할 수 있도록 허용합니다. (이 정책 보기.)
- 사용자가 보안 인증 페이지에서 자신의 암호, 액세스 키 및 SSH 퍼블릭 키를 관리할 수 있도록 허용합니다. (이 정책 보기.)
- 요청된 리전에 따라 AWS에 대한 액세스를 거부합니다. (이 정책 보기.)
- 소스 IP 주소를 기반으로 AWS에 대한 액세스를 거부합니다. (이 정책 보기.)

예제 정책: AWS Data Exchange

- AWS Data Exchange를 제외한 계정 외부의 Amazon S3 리소스에 대한 액세스를 거부합니다. (이 정책 보기.)

AWS Data Pipeline 정책의 예

- 사용자가 생성하지 않은 파이프라인에 대한 액세스를 거부합니다. (이 정책 보기.)

정책 예: Amazon DynamoDB

- 특정 Amazon DynamoDB 테이블에 대한 액세스를 허용합니다(이 정책 보기).
- 특정 Amazon DynamoDB 속성에 대한 액세스를 허용합니다(이 정책 보기.)
- Amazon Cognito ID를 기준으로 Amazon DynamoDB에 대한 항목 수준 액세스를 허용합니다(이 정책 보기).

정책의 예제: Amazon EC2

- 태그를 기준으로 Amazon EBS 볼륨을 Amazon EC2 인스턴스에 연결 또는 분리하도록 허용합니다 (이 정책 보기).
- 특정 서브넷에 있는 Amazon EC2 인스턴스를 프로그래밍 방식 및 콘솔에서 시작할 수 있도록 허용합니다(이 정책 보기).
- 특정 VPC와 관련된 Amazon EC2 보안 그룹을 프로그래밍 방식 및 콘솔에서 관리할 수 있도록 허용합니다(이 정책 보기).
- 사용자가 태그를 지정한 Amazon EC2 인스턴스를 프로그래밍 방식 및 콘솔에서 시작 또는 중지할 수 있도록 허용합니다(이 정책 보기).
- 리소스 및 보안 주체 태그 기반의 Amazon EC2 인스턴스를 프로그래밍 방식 및 콘솔에서 시작 또는 중지할 수 있도록 허용합니다(이 정책 보기).
- 리소스 및 주요 태그가 일치할 때 Amazon EC2 인스턴스를 시작 또는 중지할 수 있도록 허용합니다 (이 정책 보기).
- 특정 리전 내에서의 모든 Amazon EC2 액세스를 프로그래밍 방식으로 콘솔에서 허용 (이 정책 보기.)
- 프로그래밍 방식 및 콘솔에서 특정 Amazon EC2 인스턴스를 시작 또는 중지하고 특정 보안 그룹을 수정할 수 있도록 허용합니다(이 정책 보기).
- MFA 없이 특정 Amazon EC2 작업에 대한 액세스를 거부합니다(이 정책 보기).
- Amazon EC2 인스턴스 종료를 특정 IP 주소 범위로 제한합니다(이 정책 보기).

예제 정책 AWS Identity and Access Management(IAM)

- 정책 시뮬레이터 API에 대한 액세스를 허용합니다. (이 정책 보기.)
- 정책 시뮬레이터 콘솔에 대한 액세스를 허용합니다. (이 정책 보기.)
- 프로그래밍 방식 및 콘솔에서 특정 태그를 갖는 규칙을 수임하도록 허용합니다. (이 정책 보기.)
- 프로그래밍 방식 및 콘솔에서 여러 서비스에 대한 액세스를 허용 및 거부합니다. (이 정책 보기.)
- 프로그래밍 방식 및 콘솔에서 특정 태그가 있는 IAM 사용자에게 또 다른 특정 태그를 추가할 수 있도록 허용합니다(이 정책 보기).
- 프로그래밍 방식 및 콘솔에서 IAM 사용자 또는 역할에 대해 특정 태그를 추가할 수 있도록 허용합니다(이 정책 보기).
- 특정 태그가 있는 새 사용자만 만들 수 있도록 허용합니다. (이 정책 보기.)
- IAM 자격 증명 보고서 생성 및 검색을 허용합니다(이 정책 보기).

- 프로그래밍 방식 및 콘솔에서 그룹의 멤버십을 관리하도록 허용합니다. (이 정책 보기.)
- 특정 태그를 관리하도록 허용합니다. (이 정책 보기.)
- IAM 역할을 특정 서비스로 전달하도록 허용합니다(이 정책 보기).
- 보고 없이 IAM 콘솔에 대한 읽기 전용 액세스를 허용합니다(이 정책 보기).
- IAM 콘솔에 대한 읽기 전용 액세스를 허용합니다(이 정책 보기).
- 특정 사용자가 프로그래밍 방식 및 콘솔에서 그룹을 관리하도록 허용합니다. (이 정책 보기.)
- 프로그래밍 방식 및 콘솔에서 계정 암호 요구 사항을 설정하도록 허용합니다. (이 정책 보기.)
- 특정 경로를 지닌 사용자에 대해 정책 시뮬레이터 API의 사용을 허용합니다. (이 정책 보기.)
- 특정 경로를 지닌 사용자에 대해 정책 시뮬레이터 콘솔의 사용을 허용합니다. (이 정책 보기.)
- IAM 사용자가 MFA 디바이스를 스스로 관리하도록 허용 (이 정책 보기.)
- 프로그래밍 방식을 사용하거나 콘솔에서 IAM 사용자가 자신의 보안 인증을 설정하도록 허용합니다. (이 정책 보기.)
- IAM 콘솔에서 AWS Organizations 정책에 대해 마지막으로 액세스한 서비스 정보를 보도록 허용합니다. (이 정책 보기.)
- IAM 사용자, 그룹 또는 역할에 적용할 수 있는 관리형 정책을 제한합니다(이 정책 보기).
- 계정의 IAM 정책에 대한 액세스만 허용합니다(이 정책 보기).

AWS Lambda 정책의 예

- AWS Lambda 함수가 Amazon DynamoDB 테이블에 액세스하도록 허용합니다(이 정책 보기).

정책 예: Amazon RDS

- 특정 리전에 있는 Amazon RDS 데이터베이스에 대한 완전한 액세스를 허용합니다. (이 정책 보기.)
- 프로그래밍 방식 및 콘솔에서 Amazon RDS 데이터베이스를 복원하도록 허용합니다(이 정책 보기).
- 태그 소유자가 자신이 태그를 지정한 Amazon RDS 리소스에 대한 모든 액세스 권한을 가지도록 허용합니다(이 정책 보기).

정책 예: Amazon S3

- Amazon Cognito 사용자가 자신의 Amazon S3 버킷에 있는 객체에 액세스하도록 허용합니다(이 정책 보기).

- 페더레이션 사용자가 프로그래밍 방식 및 콘솔에서 Amazon S3에 있는 자신의 홈 디렉터리에 액세스하도록 허용합니다(이 정책 보기).
- 전체 S3 액세스를 허용하지만 관리자가 직전 30분 이내에 MFA를 사용하여 로그인하지 않은 경우 프로덕션 버킷에 대한 액세스를 명시적으로 거부합니다. (이 정책 보기.)
- IAM 사용자가 프로그래밍 방식 및 콘솔에서 Amazon S3에 있는 자신의 홈 디렉터리에 액세스하도록 허용합니다(이 정책 보기).
- 사용자가 하나의 Amazon S3 버킷을 관리하고 다른 모든 AWS 작업 및 리소스를 거부하도록 허용합니다(이 정책 보기).
- 특정 Amazon S3 버킷에 대한 `Read` 및 `Write` 액세스를 허용합니다(이 정책 보기).
- 프로그래밍 방식 및 콘솔에서 특정 Amazon S3 버킷에 대한 `Read` 및 `Write` 액세스를 허용합니다(이 정책 보기).

AWS: 날짜 및 시간에 따라 액세스 허용

이 예제는 날짜 및 시간을 기준으로 작업에 대한 액세스를 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 2020년 4월 1일부터 2020년 6월 30일(UTC) 사이에 발생하는 작업에 대한 액세스를 제한합니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

IAM 정책의 `Condition` 블록 내에서 복수 조건을 사용하는 방법에 관한 자세한 내용은 다수의 조건 값 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "service-prefix:action-name",
            "Resource": "*",
            "Condition": {
                "DateGreaterThan": {"aws:CurrentTime": "2020-04-01T00:00:00Z"},
                "DateLessThan": {"aws:CurrentTime": "2020-06-30T23:59:59Z"}
            }
        }
    ]
}
```

> **Note**
>
> 정책 변수를 Date 조건 연산자와 함께 사용할 수 없습니다. 자세한 내용은 조건 요소 섹션을 참조하세요.

AWS: AWS 리전 활성화 및 비활성화 허용

이 예제는 관리자가 아시아 태평양(홍콩) 리전(ap-east-1)을 활성화하고 비활성화할 수 있도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다. 이 설정은 AWS Management Console의 계정 설정 페이지에 표시됩니다. 이 페이지에는 계정 관리자만이 보고 관리해야 하는 민감한 계정 수준 정보가 포함되어 있습니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

> **Important**
>
> 기본적으로 활성화되는 리전은 활성화하거나 비활성화할 수 없습니다. 기본적으로 비활성화되는 리전만 추가할 수 있습니다. 자세한 내용은 AWS 일반 참조의 AWS 리전 관리 단원을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnableDisableHongKong",
            "Effect": "Allow",
            "Action": [
                "account:EnableRegion",
                "account:DisableRegion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"account:TargetRegion": "ap-east-1"}
            }
        },
        {
            "Sid": "ViewConsole",
            "Effect": "Allow",
```

```
            "Action": [
                "account:ListRegions"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS: MFA 인증 IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용

이 예시는 다중 인증(MFA)을 사용하여 인증된 IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용하는 자격 증명 기반 정책을 생성하는 방법을 보여줍니다. 이 AWS Management Console 페이지에는 계정 ID 및 정식 사용자 ID와 같은 계정 정보가 표시됩니다. 또한 사용자는 자신의 암호, 액세스 키, MFA 디바이스, X.509 인증서, SSH 키 및 Git 자격 증명을 보고 편집할 수 있습니다. 이 예제 정책에는 페이지에 있는 모든 정보를 보고 편집하는 데 필요한 권한이 포함되어 있습니다. 또한 사용자가 AWS에서 다른 작업을 수행하기 전에 MFA 사용을 설정하고 인증해야 합니다. 사용자가 MFA를 사용하지 않고 자신의 자격 증명을 관리하도록 허용하려면 AWS: IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용 섹션을 참조하세요.

사용자가 보안 인증 페이지에 액세스할 수 있는 방법을 알아보려면 IAM 사용자가 자신의 암호를 변경하는 방법(콘솔) 섹션을 참조하세요.

> **Note**
>
> - 이 정책 예제에서는 사용자가 처음으로 AWS Management Console에 로그인하는 동안 암호 재설정을 허용하지 않습니다. 새 사용자가 로그인할 때까지 새 사용자에게 권한을 부여하지 않는 것이 좋습니다. 자세한 내용은 IAM 사용자를 안전하게 생성하려면 어떻게 해야 하나요? 단원을 참조하세요. 이렇게 하면 암호가 만료된 사용자가 로그인 중 암호를 재설정할 수 없습니다. `iam:ChangePassword` 및 `iam:GetAccountPasswordPolicy`을 `DenyAllExceptListedIfNoMFA` 문에 추가하여 이를 허용할 수 있습니다. 그러나 사용자가 MFA 없이 암호를 변경하도록 허용하면 보안 위험이 발생할 수 있으므로 이는 권장되지 않습니다.
> - 프로그래밍 방식의 액세스에 이 정책을 사용하려면 `GetSessionToken`을 호출하여 MFA로 인증해야 합니다. 자세한 내용은 MFA를 통한 보안 API 액세스 단원을 참조하세요.

이 정책이 하는 일은 무엇입니까?

- `AllowViewAccountInfo` 문은 사용자가 계정 수중 정보를 볼 수 있도록 허용합니다. 이러한 권한은 리소스 ARN을 지원하지 않거나 리소스 ARN을 지정하는 데 필요하지 않기 때문에 자신의 문에 포함되어 있어야 합니다. 권한 대신 `"Resource" : "*"`를 지정합니다. 이 문에는 사용자가 특정 정보를 볼 수 있도록 허용하는 다음 작업이 포함되어 있습니다.
  - `GetAccountPasswordPolicy` - 자신의 IAM 사용자 암호를 변경하는 동안 계정 암호 요구 사항을 봅니다.
  - `ListVirtualMFADevices` - 사용자에 대해 활성화된 가상 MFA 디바이스에 대한 세부 정보를 봅니다.
- `AllowManageOwnPasswords` 문은 사용자가 자신의 암호를 변경할 수 있도록 허용합니다. 또한 이 문에는 My security credentials(내 보안 자격 증명) 페이지에 있는 대부분의 정보를 보는 데 필요한 `GetUser` 작업도 포함되어 있습니다.
- `AllowManageOwnAccessKeys` 문은 사용자가 자신의 액세스 키를 생성, 업데이트 및 삭제할 수 있도록 허용합니다. 사용자가 지정된 액세스 키가 마지막으로 사용된 시간에 대한 정보를 검색할 수도 있습니다.
- `AllowManageOwnSigningCertificates` 문은 사용자가 자신의 서명 인증서를 업로드, 업데이트 및 삭제할 수 있도록 허용합니다.
- `AllowManageOwnSSHPublicKeys` 문은 사용자가 CodeCommit에 대한 자신의 SSH 퍼블릭 키를 업로드, 업데이트 및 삭제할 수 있도록 허용합니다.
- `AllowManageOwnGitCredentials` 문은 사용자가 CodeCommit에 대한 자신의 Git 자격 증명을 생성, 업데이트 및 삭제할 수 있도록 허용합니다.
- `AllowManageOwnVirtualMFADevice` 문은 사용자가 자신의 가상 MFA 디바이스를 생성할 수 있도록 허용합니다. 이 문의 리소스 ARN은 사용자가 임의의 이름으로 MFA 디바이스를 생성할 수 있도록 허용하지만 정책의 다른 문은 사용자가 현재 로그인한 사용자에게만 디바이스를 연결할 수 있도록 허용합니다.
- `AllowManageOwnUserMFA` 문은 사용자가 자신의 사용자에 대해 가상, U2F 또는 하드웨어 MFA 디바이스를 보거나 관리할 수 있도록 허용합니다. 이 문의 리소스 ARN은 사용자 자신의 IAM 사용자에 대한 액세스만 허용합니다. 사용자는 다른 사용자의 MFA 디바이스를 보거나 관리할 수 없습니다.
- `DenyAllExceptListedIfNoMFA` 문은 사용자가 MFA를 사용하여 로그인하지 않은 경우에만 몇 가지 나열된 작업을 제외한 모든 AWS의 모든 작업에 대한 액세스를 거부합니다. 이 문은 `"Deny"` 및 `"NotAction"`의 조합을 사용하여 나열되지 않은 모든 작업에 대한 액세스를 명시적으로 거부합니다. 나열된 항목은 이 문에 따라 거부되거나 허용되지 않습니다. 하지만 정책의 다른 문에서 작업이 허용됩니다. 이 문의 로직에 대한 자세한 내용은 NotAction 및 Deny 섹션을 참조하세요. 사용자가 MFA를 사용하여 로그인한 경우 `Condition` 테스트가 실패하며 이 문은 어떠한 작업도 거부하지 않습니다. 이 경우 사용자에 대한 다른 정책 또는 문에 따라 사용자의 권한이 결정됩니다.

이 문을 사용하면 MFA를 사용하여 로그인하지 않은 사용자는 나열된 작업만 수행할 수 있습니다. 또한 사용자는 다른 문 또는 정책이 해당 작업에 대한 액세스를 허용하는 경우에만 나열된 작업을 수행할 수 있습니다. MFA 권한 부여가 없으면 `iam:ChangePassword` 작업이 허용되지 않기 때문에 사용자는 로그인 시 암호를 생성할 수 없습니다.

`...IfExists` 키를 분실했을 경우 `Bool` 연산자의 `aws:MultiFactorAuthPresent` 버전은 조건이 true로 반환됩니다. 즉, 액세스 키와 같은 장기 자격 증명으로 API를 액세스하는 사용자는 비 IAM API 작업에 대한 액세스가 거부됩니다.

이 정책은 사용자가 IAM 콘솔에서 사용자 페이지를 보거나 이 페이지를 사용하여 자신의 사용자 정보에 액세스할 수 있도록 허용합니다. 이 작업을 허용하려면 `iam:ListUsers` 작업을 `AllowViewAccountInfo` 문과 `DenyAllExceptListedIfNoMFA` 문에 추가합니다. 또한 이 문은 사용자가 자신의 사용자 페이지에서 암호를 변경하도록 허용하지 않습니다. 이 작업을 허용하려면 `iam:GetLoginProfile` 및 `iam:UpdateLoginProfile` 작업을 `AllowManageOwnPasswords` 문에 추가합니다. 또한 사용자가 MFA를 사용하여 로그인하지 않고 자신의 사용자 페이지에서 자신의 암호를 변경할 수 있도록 허용하려면 `iam:UpdateLoginProfile` 작업을 `DenyAllExceptListedIfNoMFA` 문에 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowViewAccountInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy",
                "iam:ListVirtualMFADevices"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswords",
            "Effect": "Allow",
            "Action": [
                "iam:ChangePassword",
                "iam:GetUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
```

```
            "Sid": "AllowManageOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:CreateAccessKey",
                "iam:DeleteAccessKey",
                "iam:ListAccessKeys",
                "iam:UpdateAccessKey",
                "iam:GetAccessKeyLastUsed"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnSigningCertificates",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteSigningCertificate",
                "iam:ListSigningCertificates",
                "iam:UpdateSigningCertificate",
                "iam:UploadSigningCertificate"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnSSHPublicKeys",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteSSHPublicKey",
                "iam:GetSSHPublicKey",
                "iam:ListSSHPublicKeys",
                "iam:UpdateSSHPublicKey",
                "iam:UploadSSHPublicKey"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnGitCredentials",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceSpecificCredential",
                "iam:DeleteServiceSpecificCredential",
                "iam:ListServiceSpecificCredentials",
                "iam:ResetServiceSpecificCredential",
                "iam:UpdateServiceSpecificCredential"
            ],
```

```
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnVirtualMFADevice",
            "Effect": "Allow",
            "Action": [
                "iam:CreateVirtualMFADevice"
            ],
            "Resource": "arn:aws:iam::*:mfa/*"
        },
        {
            "Sid": "AllowManageOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice",
                "iam:EnableMFADevice",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "DenyAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:CreateVirtualMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:GetMFADevice",
                "iam:ListMFADevices",
                "iam:ListVirtualMFADevices",
                "iam:ResyncMFADevice",
                "sts:GetSessionToken"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

AWS: 지정된 기간 동안 MFA를 사용한 특정 액세스 허용

이 예제는 논리 AND를 사용하여 평가되는 여러 조건을 사용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. SERVICE-NAME-1으로 명명된 서비스에 대해 모든 액세스를 허용하고 ACTION-NAME-A로 명명된 서비스에서 ACTION-NAME-B 및 SERVICE-NAME-2 작업에 대한 액세스를 허용합니다. 이들 작업은 사용자가 멀티 팩터 인증(MFA)을 통해 인증된 경우에만 허용됩니다. 액세스는 2017년 7월 1일과 2017년 12월 31일(UTC) 사이에 발생하는 작업으로 제한됩니다(7월 1일 및 12월 31일 포함). 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

IAM 정책의 Condition 블록 내에서 복수 조건을 사용하는 방법에 관한 자세한 내용은 다수의 조건 값 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "service-prefix-1:*",
            "service-prefix-2:action-name-a",
            "service-prefix-2:action-name-b"
        ],
        "Resource": "*",
        "Condition": {
            "Bool": {"aws:MultiFactorAuthPresent": true},
            "DateGreaterThan": {"aws:CurrentTime": "2017-07-01T00:00:00Z"},
            "DateLessThan": {"aws:CurrentTime": "2017-12-31T23:59:59Z"}
        }
    }
}
```

AWS: IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용

이 예시는 보안 인증 페이지에서 IAM 사용자가 모든 보안 인증을 관리할 수 있도록 허용하는 자격 증명 기반 정책을 생성하는 방법을 보여줍니다. 이 AWS Management Console 페이지에는 계정 ID 및 정식 사용자 ID와 같은 계정 정보가 표시됩니다. 또한 사용자는 자신의 암호, 액세스 키, X.509 인증서, SSH 키, Git 자격 증명을 보고 편집할 수 있습니다. 이 예제 정책에는 사용자의 MFA 디바이스를 제외하고 페이지에 있는 모든 정보를 보고 편집하는 데 필요한 권한이 포함되어 있습니다. 사용자가 MFA를 사용

하여 자신의 자격 증명을 관리하도록 허용하려면 AWS: MFA 인증 IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용 섹션을 참조하세요.

사용자가 보안 인증 페이지에 액세스할 수 있는 방법을 알아보려면 IAM 사용자가 자신의 암호를 변경하는 방법(콘솔) 섹션을 참조하세요.

이 정책이 하는 일은 무엇입니까?

- `AllowViewAccountInfo` 문은 사용자가 계정 수준 정보를 볼 수 있도록 허용합니다. 이러한 권한은 리소스 ARN을 지원하지 않거나 리소스 ARN을 지정하는 데 필요하지 않기 때문에 자신의 문에 포함되어 있어야 합니다. 권한 대신 `"Resource" : "*"`를 지정합니다. 이 문에는 사용자가 특정 정보를 볼 수 있도록 허용하는 다음 작업이 포함되어 있습니다.
  - `GetAccountPasswordPolicy` - 자신의 IAM 사용자 암호를 변경하는 동안 계정 암호 요구 사항을 봅니다.
  - `GetAccountSummary` - 계정 ID 및 계정 정식 사용자 ID를 봅니다.
- `AllowManageOwnPasswords` 문은 사용자가 자신의 암호를 변경할 수 있도록 허용합니다. 또한 이 문에는 My security credentials(내 보안 자격 증명) 페이지에 있는 대부분의 정보를 보는 데 필요한 `GetUser` 작업도 포함되어 있습니다.
- `AllowManageOwnAccessKeys` 문은 사용자가 자신의 액세스 키를 생성, 업데이트 및 삭제할 수 있도록 허용합니다. 사용자가 지정된 액세스 키가 마지막으로 사용된 시간에 대한 정보를 검색할 수도 있습니다.
- `AllowManageOwnSigningCertificates` 문은 사용자가 자신의 서명 인증서를 업로드, 업데이트 및 삭제할 수 있도록 허용합니다.
- `AllowManageOwnSSHPublicKeys` 문은 사용자가 CodeCommit에 대한 자신의 SSH 퍼블릭 키를 업로드, 업데이트 및 삭제할 수 있도록 허용합니다.
- `AllowManageOwnGitCredentials` 문은 사용자가 CodeCommit에 대한 자신의 Git 자격 증명을 생성, 업데이트 및 삭제할 수 있도록 허용합니다.

이 정책은 사용자가 자신의 MFA 디바이스를 보거나 관리하도록 허용하지 않습니다. 또한 사용자는 IAM 콘솔에서 사용자 페이지를 보거나 이 페이지를 사용하여 자신의 사용자 정보에 액세스할 수 없습니다. 이 작업을 허용하려면 `iam:ListUsers` 작업을 `AllowViewAccountInfo` 문에 추가합니다. 또한 이 문은 사용자가 자신의 사용자 페이지에서 암호를 변경하도록 허용하지 않습니다. 이 작업을 허용하려면 `iam:CreateLoginProfile`, `iam:DeleteLoginProfile`, `iam:GetLoginProfile` 및 `iam:UpdateLoginProfile` 작업을 `AllowManageOwnPasswords` 문에 추가합니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowViewAccountInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy",
                "iam:GetAccountSummary"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswords",
            "Effect": "Allow",
            "Action": [
                "iam:ChangePassword",
                "iam:GetUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:CreateAccessKey",
                "iam:DeleteAccessKey",
                "iam:ListAccessKeys",
                "iam:UpdateAccessKey",
                "iam:GetAccessKeyLastUsed"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnSigningCertificates",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteSigningCertificate",
                "iam:ListSigningCertificates",
                "iam:UpdateSigningCertificate",
                "iam:UploadSigningCertificate"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
```

```
            "Sid": "AllowManageOwnSSHPublicKeys",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteSSHPublicKey",
                "iam:GetSSHPublicKey",
                "iam:ListSSHPublicKeys",
                "iam:UpdateSSHPublicKey",
                "iam:UploadSSHPublicKey"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnGitCredentials",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceSpecificCredential",
                "iam:DeleteServiceSpecificCredential",
                "iam:ListServiceSpecificCredentials",
                "iam:ResetServiceSpecificCredential",
                "iam:UpdateServiceSpecificCredential"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

AWS: MFA 인증 IAM 사용자가 보안 인증 페이지에서 자신의 MFA 디바이스를 관리할 수 있도록 허용

이 예시는 다중 인증(MFA)을 사용하여 인증된 IAM 사용자가 보안 인증 페이지에서 자신의 MFA 디바이스를 관리할 수 있도록 허용하는 자격 증명 기반 정책을 생성하는 방법을 보여줍니다. 이 AWS Management Console 페이지에는 계정 및 사용자 정보가 표시되지만, 사용자는 자신의 MFA 디바이스만 보고 편집할 수 있습니다. 사용자가 MFA를 사용하여 자신의 모든 자격 증명을 관리하도록 허용하려면 AWS: MFA 인증 IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용 섹션을 참조하세요.

> **Note**
>
> 이 정책을 가진 IAM 사용자가 MFA 인증을 받지 않은 경우 이 정책은 MFA를 사용하여 인증하는 데 필요한 AWS 작업을 제외한 모든 해당 작업에 대한 액세스를 거부합니다. AWS CLI 및 AWS API를 사용하려면 IAM 사용자가 먼저 AWS STS GetSessionToken 작업을 사용하여

MFA 토큰을 검색한 다음 해당 토큰을 사용하여 원하는 작업을 인증해야 합니다. 리소스 기반 정책이나 기타 아이덴티티 기반 정책 등의 기타 정책은 다른 서비스의 작업을 허용할 수 있습니다. 이 정책은 IAM 사용자가 MFA 인증을 받지 않은 경우 해당 액세스를 거부합니다.

사용자가 보안 인증 페이지에 액세스할 수 있는 방법을 알아보려면 IAM 사용자가 자신의 암호를 변경하는 방법(콘솔) 섹션을 참조하세요.

이 정책이 하는 일은 무엇입니까?

- `AllowViewAccountInfo` 문은 사용자가 사용자에 대해 활성화된 가상 MFA 디바이스에 대한 세부 정보를 볼 수 있도록 허용합니다. 이 권한은 리소스 ARN 지정을 지원하지 않으므로 자신의 문에 들어 있어야 합니다. 그 대신 `"Resource" : "*"`를 지정해야 합니다.
- `AllowManageOwnVirtualMFADevice` 문은 사용자가 자신의 가상 MFA 디바이스를 생성할 수 있도록 허용합니다. 이 문의 리소스 ARN은 사용자가 임의의 이름으로 MFA 디바이스를 생성할 수 있도록 허용하지만 정책의 다른 문은 사용자가 현재 로그인한 사용자에게만 디바이스를 연결할 수 있도록 허용합니다.
- `AllowManageOwnUserMFA` 문은 사용자가 자신의 가상, U2F 또는 하드웨어 MFA 디바이스를 보거나 관리할 수 있도록 허용합니다. 이 문의 리소스 ARN은 사용자 자신의 IAM 사용자에 대한 액세스만 허용합니다. 사용자는 다른 사용자의 MFA 디바이스를 보거나 관리할 수 없습니다.
- `DenyAllExceptListedIfNoMFA` 문은 사용자가 MFA를 사용하여 로그인하지 않은 경우에만 몇 가지 나열된 작업을 제외한 모든 AWS의 모든 작업에 대한 액세스를 거부합니다. 이 문은 `"Deny"` 및 `"NotAction"`의 조합을 사용하여 나열되지 않은 모든 작업에 대한 액세스를 명시적으로 거부합니다. 나열된 항목은 이 문에 따라 거부되거나 허용되지 않습니다. 하지만 정책의 다른 문에서 작업이 허용됩니다. 이 문의 로직에 대한 자세한 내용은 NotAction 및 Deny 섹션을 참조하세요. 사용자가 MFA를 사용하여 로그인한 경우 `Condition` 테스트가 실패하며 이 문은 어떠한 작업도 거부하지 않습니다. 이 경우 사용자에 대한 다른 정책 또는 문에 따라 사용자의 권한이 결정됩니다.

  이 문을 사용하면 MFA를 사용하여 로그인하지 않은 사용자는 나열된 작업만 수행할 수 있습니다. 또한 사용자는 다른 문 또는 정책이 해당 작업에 대한 액세스를 허용하는 경우에만 나열된 작업을 수행할 수 있습니다.

  `...IfExists` 키를 분실했을 경우 `Bool` 연산자의 `aws:MultiFactorAuthPresent` 버전은 조건이 true로 반환됩니다. 따라서 액세스 키와 같은 장기 자격 증명을 사용하여 API 작업에 액세스하는 사용자는 비 IAM API 작업에 대한 액세스가 거부됩니다.

이 정책은 사용자가 IAM 콘솔에서 사용자 페이지를 보거나 이 페이지를 사용하여 자신의 사용자 정보에 액세스할 수 있도록 허용합니다. 이 작업을 허용하려면 `iam:ListUsers` 작업을 `AllowViewAccountInfo` 문과 `DenyAllExceptListedIfNoMFA` 문에 추가합니다.

> **Warning**
>
> MFA 인증 없이 MFA 디바이스를 삭제할 수 있는 권한을 추가하지 마세요. 이 정책을 보유한 사용자는 스스로를 가상 MFA 디바이스로 지정하려 하고 `iam:DeleteVirtualMFADevice` 수행에 필요한 권한이 부여되지 않았다는 오류가 표시될 수 있습니다. 이 경우 `DenyAllExceptListedIfNoMFA` 문에 해당 권한을 추가하지 마세요. MFA를 사용하여 인증되지 않은 사용자에게 MFA 디바이스 삭제를 허용해서는 안 됩니다. 이전에 사용자에게 가상 MFA 디바이스를 할당하기 시작하고 프로세스를 취소한 경우 이 오류가 표시될 수 있습니다. 이 문제를 해결하려면 사용자 또는 다른 관리자가 AWS CLI 또는 AWS API를 사용하여 사용자의 기존 가상 MFA 디바이스를 삭제해야 합니다. 자세한 내용은 iam:DeleteVirtualMFADevice를 수행할 권한이 없음 단원을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowViewAccountInfo",
            "Effect": "Allow",
            "Action": "iam:ListVirtualMFADevices",
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnVirtualMFADevice",
            "Effect": "Allow",
            "Action": [
                "iam:CreateVirtualMFADevice"
            ],
            "Resource": "arn:aws:iam::*:mfa/*"
        },
        {
            "Sid": "AllowManageOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
```

```
                "iam:GetMFADevice",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "DenyAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:CreateVirtualMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
                "iam:ListVirtualMFADevices",
                "iam:ResyncMFADevice",
                "sts:GetSessionToken"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {"aws:MultiFactorAuthPresent": "false"}
            }
        }
    ]
}
```

AWS: IAM 사용자가 보안 인증 페이지에서 자신의 콘솔 암호를 변경할 수 있도록 허용

이 예시는 보안 인증 페이지에서 IAM 사용자가 모든 AWS Management Console 암호를 변경할 수 있도록 허용하는 자격 증명 기반 정책을 생성하는 방법을 보여줍니다. 이 AWS Management Console 페이지에는 계정 및 사용자 정보가 표시되지만, 사용자는 자신의 암호에만 액세스할 수 있습니다. 사용자가 MFA를 사용하여 자신의 모든 자격 증명을 관리하도록 허용하려면 AWS: MFA 인증 IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용 섹션을 참조하세요. 사용자가 MFA를 사용하지 않고 자신의 자격 증명을 관리하도록 허용하려면 AWS: IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용 섹션을 참조하세요.

사용자가 보안 인증 페이지에 액세스할 수 있는 방법을 알아보려면 IAM 사용자가 자신의 암호를 변경하는 방법(콘솔) 섹션을 참조하세요.

이 정책이 하는 일은 무엇입니까?

- `ViewAccountPasswordRequirements` 문은 사용자가 자신의 IAM 사용자 암호를 변경하는 동안 계정 암호 요구 사항을 볼 수 있도록 허용합니다.
- `ChangeOwnPassword` 문은 사용자가 자신의 암호를 변경할 수 있도록 허용합니다. 또한 이 문에는 My security credentials(내 보안 자격 증명) 페이지에 있는 대부분의 정보를 보는 데 필요한 `GetUser` 작업도 포함되어 있습니다.

이 정책은 사용자가 IAM 콘솔에서 사용자 페이지를 보거나 이 페이지를 사용하여 자신의 사용자 정보에 액세스할 수 있도록 허용합니다. 이 작업을 허용하려면 `iam:ListUsers` 작업을 `ViewAccountPasswordRequirements` 문에 추가합니다. 또한 이 문은 사용자가 자신의 사용자 페이지에서 암호를 변경하도록 허용하지 않습니다. 이 작업을 허용하려면 `iam:GetLoginProfile` 및 `iam:UpdateLoginProfile` 작업을 `ChangeOwnPasswords` 문에 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewAccountPasswordRequirements",
            "Effect": "Allow",
            "Action": "iam:GetAccountPasswordPolicy",
            "Resource": "*"
        },
        {
            "Sid": "ChangeOwnPassword",
            "Effect": "Allow",
            "Action": [
                "iam:GetUser",
                "iam:ChangePassword"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

AWS: IAM 사용자가 보안 인증 페이지에서 자신의 암호, 액세스 키 및 SSH 퍼블릭 키를 관리할 수 있도록 허용

이 예시는 보안 인증 페이지에서 IAM 사용자가 암호, 액세스 키, X.509 인증서를 관리할 수 있도록 허용하는 자격 증명 기반 정책을 생성하는 방법을 보여줍니다. 이 AWS Management Console 페이지에는 계정 ID 및 정식 사용자 ID와 같은 계정 정보가 표시됩니다. 또한 사용자는 자신의 암호, 액세스

키, MFA 디바이스, X.509 인증서, SSH 키 및 Git 자격 증명을 보고 편집할 수 있습니다. 이 예제 정책에는 자신의 암호, 액세스 키 및 X.509 인증서만 보고 편집하는 데 필요한 권한이 포함되어 있습니다. 사용자가 MFA를 사용하여 자신의 모든 자격 증명을 관리하도록 허용하려면 AWS: MFA 인증 IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용 섹션을 참조하세요. 사용자가 MFA를 사용하지 않고 자신의 자격 증명을 관리하도록 허용하려면 AWS: IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용 섹션을 참조하세요.

사용자가 보안 인증 페이지에 액세스할 수 있는 방법을 알아보려면 IAM 사용자가 자신의 암호를 변경하는 방법(콘솔) 섹션을 참조하세요.

이 정책이 하는 일은 무엇입니까?

- `AllowViewAccountInfo` 문은 사용자가 계정 수준 정보를 볼 수 있도록 허용합니다. 이러한 권한은 리소스 ARN을 지원하지 않거나 리소스 ARN을 지정하는 데 필요하지 않기 때문에 자신의 문에 포함되어 있어야 합니다. 권한 대신 `"Resource" : "*"`를 지정합니다. 이 문에는 사용자가 특정 정보를 볼 수 있도록 허용하는 다음 작업이 포함되어 있습니다.
  - `GetAccountPasswordPolicy` - 자신의 IAM 사용자 암호를 변경하는 동안 계정 암호 요구 사항을 봅니다.
  - `GetAccountSummary` - 계정 ID 및 계정 정식 사용자 ID를 봅니다.
- `AllowManageOwnPasswords` 문은 사용자가 자신의 암호를 변경할 수 있도록 허용합니다. 또한 이 문에는 My security credentials(내 보안 자격 증명) 페이지에 있는 대부분의 정보를 보는 데 필요한 `GetUser` 작업도 포함되어 있습니다.
- `AllowManageOwnAccessKeys` 문은 사용자가 자신의 액세스 키를 생성, 업데이트 및 삭제할 수 있도록 허용합니다. 사용자가 지정된 액세스 키가 마지막으로 사용된 시간에 대한 정보를 검색할 수도 있습니다.
- `AllowManageOwnSSHPublicKeys` 문은 사용자가 CodeCommit에 대한 자신의 SSH 퍼블릭 키를 업로드, 업데이트 및 삭제할 수 있도록 허용합니다.

이 정책은 사용자가 자신의 MFA 디바이스를 보거나 관리하도록 허용하지 않습니다. 또한 사용자는 IAM 콘솔에서 사용자 페이지를 보거나 이 페이지를 사용하여 자신의 사용자 정보에 액세스할 수 없습니다. 이 작업을 허용하려면 `iam:ListUsers` 작업을 `AllowViewAccountInfo` 문에 추가합니다. 또한 이 문은 사용자가 자신의 사용자 페이지에서 암호를 변경하도록 허용하지 않습니다. 이 작업을 허용하려면 `iam:GetLoginProfile` 및 `iam:UpdateLoginProfile` 작업을 `AllowManageOwnPasswords` 문에 추가합니다.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AllowViewAccountInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy",
                "iam:GetAccountSummary"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswords",
            "Effect": "Allow",
            "Action": [
                "iam:ChangePassword",
                "iam:GetUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:CreateAccessKey",
                "iam:DeleteAccessKey",
                "iam:ListAccessKeys",
                "iam:UpdateAccessKey",
                "iam:GetAccessKeyLastUsed"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnSSHPublicKeys",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteSSHPublicKey",
                "iam:GetSSHPublicKey",
                "iam:ListSSHPublicKeys",
                "iam:UpdateSSHPublicKey",
                "iam:UploadSSHPublicKey"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
```

```
}
```

AWS: 요청된 리전에 따라 AWS에 대한 액세스를 거부

이 예에서는 NotAction을 사용하도록 지정된 서비스의 작업을 제외하고 aws:RequestedRegion 조건 키를 사용하여 지정된 리전 외부의 작업에 대한 액세스를 거부하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다. 이 정책을 사용하려면 정책 예제의 `#### ## ### ###`를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

이 정책은 NotAction 요소를 Deny 효과와 함께 사용하여 문에 나열되지 않은 모든 작업에 대한 액세스를 거부합니다. CloudFront, IAM, Route 53 및 AWS Support 서비스의 작업은 거부되어서는 안 되는데, 이는 이러한 서비스가 물리적으로 us-east-1 리전에 위치한 단일 엔드포인트가 포함된 유명한 AWS 전역 서비스이기 때문입니다. 이러한 서비스에 대한 모든 요청이 us-east-1 리전으로 전달되기 때문에 NotAction 요소 없이 요청은 거부됩니다. 이 요소를 편집하여 budgets, globalaccelerator, importexport, organizations 또는 waf 등과 같이 기타 AWS 전역 서비스에 대한 작업을 포함합니다. AWS Chatbot 및 AWS Device Farm과(와) 같은 일부 다른 전역 서비스는 물리적으로 us-west-2 리전에 위치한 엔드포인트를 포함한 전역 서비스입니다. 전역 엔드포인트를 보유한 모든 서비스에 대해 알아보려면 AWS 일반 참조의 AWS 리전 및 엔드포인트를 참조하세요. NotAction 효과와 Deny 요소의 사용에 대해 자세히 알아보려면 IAM JSON 정책 요소: NotAction 섹션을 참조하세요.

> **Important**
>
> 이 정책은 어떤 작업도 허용하지 않습니다. 이 정책을 특정 작업을 허용하는 다른 정책과 함께 사용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyAllOutsideRequestedRegions",
            "Effect": "Deny",
            "NotAction": [
                "cloudfront:*",
                "iam:*",
                "route53:*",
                "support:*"
```

```
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestedRegion": [
                        "eu-central-1",
                        "eu-west-1",
                        "eu-west-2",
                        "eu-west-3"
                    ]
                }
            }
        }
    ]
}
```

AWS: 소스 IP를 바탕으로 AWS에 대한 액세스 거부

이 예제는 지정된 IP 범위 외부의 보안 주체로부터 요청이 온 경우 계정의 모든 AWS 작업에 대한 액세스를 거부하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 회사의 IP 주소가 지정된 범위 내에 있는 경우에 유용합니다. 이 예제에서는 CIDR 범위 192.0.2.0/24 또는 203.0.113.0/24에서 시작되지 않는 한 요청이 거부됩니다. 정책은 원래 요청자의 IP 주소가 보존되므로 전달 액세스 세션를 사용하는 AWS 서비스에 의한 요청을 거부하지 않습니다.

`"Effect": "Deny"`와 동일한 정책 문에서 부정적인 조건을 사용해야 합니다. 이렇게 하면 정책 문에 지정된 작업이 지정된 조건을 제외한 모든 조건에서 명시적으로 거부됩니다.

> **Important**
>
> 이 정책은 어떤 작업도 허용하지 않습니다. 이 정책을 특정 작업을 허용하는 다른 정책과 함께 사용합니다.

다른 정책에서 작업을 허용하는 경우 보안 주체는 IP 주소 범위 내에서 요청을 할 수 있습니다. 또한 AWS 서비스는 보안 주체의 자격 증명을 사용하여 요청할 수도 있습니다. 보안 주체가 IP 범위 밖에서 요청을 하면 요청이 거부됩니다.

정책에서 `aws:SourceIp`가 작동하지 않는 경우를 포함해 `aws:SourceIp` 조건 키에 대한 자세한 내용은 AWS 글로벌 조건 컨텍스트 키 섹션을 참조하세요.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
            "NotIpAddress": {
                "aws:SourceIp": [
                    "192.0.2.0/24",
                    "203.0.113.0/24"
                ]
            }
        }
    }
}
```

AWS: AWS Data Exchange를 제외한 계정 외부의 Amazon S3 리소스에 대한 액세스 거부

이 예제는 AWS Data Exchange이(가) 일반 작업에 필요한 리소스를 제외하고 계정에 속하지 않은 AWS의 모든 리소스에 대한 액세스를 거부하는 ID 기반 정책을 생성하는 방법을 보여줍니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

조건 키 `aws:ResourceOrgPaths` 및 `aws:ResourceOrgID`을(를) 사용하여 AWS Data Exchange 소유 리소스를 고려하면서 조직 또는 조직 단위 내 리소스에 대한 액세스를 제한하는 유사한 정책을 만들 수 있습니다.

환경에서 AWS Data Exchange를 사용하는 경우 서비스는 서비스 계정이 소유한 Amazon S3 버킷과 같은 리소스를 생성하고 상호 작용합니다. 예를 들어, AWS Data Exchange는 AWS Data Exchange API를 호출하는 IAM 보안 주체(사용자 또는 역할)를 대신하여 AWS Data Exchange 서비스가 소유한 Amazon S3 버킷에 요청을 전송합니다. 이 경우 AWS Data Exchange 소유 리소스를 고려하지 않고 정책에서 `aws:ResourceAccount`, `aws:ResourceOrgPaths` 또는`aws:ResourceOrgID`을(를) 사용하면 서비스 계정이 소유한 버킷에 대한 액세스가 거부됩니다.

- `DenyAllAwsResourcesOutsideAccountExceptS3` 문은 문에 나열되지 않고 나열된 계정에도 속하지 않는 모든 작업에 대한 액세스를 명시적으로 거부하는 Deny 효과와 함께 `NotAction` 요소를 사용합니다. `NotAction` 요소는 이 문의 예외를 나타냅니다. 이러한 작업은 이 문의 예외입니다. AWS Data Exchange에서 생성한 리소스에서 작업을 수행하는 경우 정책이 이를 거부하기 때문입니다.

- DenyAllS3ResoucesOutsideAccountExceptDataExchange 문은 ResourceAccount 및 CalledVia 조건을 조합해서 사용하여 이전 문에서 제외된 세 가지 Amazon S3 작업에 대한 액세스를 거부합니다. 이 문은 리소스가 나열된 계정에 속하지 않는 경우 그리고 호출 서비스가 AWS Data Exchange이 아닌 경우 작업을 거부합니다. 이 문은 리소스가 나열된 계정에 속하거나 나열된 서비스 주체인 dataexchange.amazonaws.com이 작업을 수행하는 경우 작업을 거부하지 않습니다.

> **Important**
>
> 이 정책은 어떤 작업도 허용하지 않습니다. 이는 나열된 계정에 속하지 않는 문에 나열된 모든 리소스에 대한 액세스를 명시적으로 거부하는 Deny 효과를 사용합니다. 이 정책을 특정 리소스에 대한 액세스를 허용하는 다른 정책과 함께 사용합니다.

다음 예제는 필요한 Amazon S3 버킷에 대한 액세스를 허용하도록 정책을 구성하는 방법을 보여줍니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAllAwsReourcesOutsideAccountExceptAmazonS3",
      "Effect": "Deny",
      "NotAction": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "111122223333"
          ]
        }
      }
    },
    {
      "Sid": "DenyAllS3ResourcesOutsideAccountExceptDataExchange",
      "Effect": "Deny",
      "Action": [
```

```
        "s3:GetObject",
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "111122223333"
          ]
        },
        "ForAllValues:StringNotEquals": {
          "aws:CalledVia": [
            "dataexchange.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

AWS Data Pipeline: 사용자가 생성하지 않은 DataPipeline 파이프라인에 대한 액세스 거부

이 예제는 사용자가 생성하지 않은 파이프라인에 대한 액세스를 거부하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. `PipelineCreator` 필드의 값이 IAM 사용자 이름과 일치하는 경우 지정된 작업이 거부되지 않습니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다.

> **Important**
>
> 이 정책은 어떤 작업도 허용하지 않습니다. 이 정책을 특정 작업을 허용하는 다른 정책과 함께 사용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExplicitDenyIfNotTheOwner",
            "Effect": "Deny",
```

```
            "Action": [
                "datapipeline:ActivatePipeline",
                "datapipeline:AddTags",
                "datapipeline:DeactivatePipeline",
                "datapipeline:DeletePipeline",
                "datapipeline:DescribeObjects",
                "datapipeline:EvaluateExpression",
                "datapipeline:GetPipelineDefinition",
                "datapipeline:PollForTask",
                "datapipeline:PutPipelineDefinition",
                "datapipeline:QueryObjects",
                "datapipeline:RemoveTags",
                "datapipeline:ReportTaskProgress",
                "datapipeline:ReportTaskRunnerHeartbeat",
                "datapipeline:SetStatus",
                "datapipeline:SetTaskStatus",
                "datapipeline:ValidatePipelineDefinition"
            ],
            "Resource": ["*"],
            "Condition": {
                "StringNotEquals": {"datapipeline:PipelineCreator": "${aws:userid}"}
            }
        }
    ]
}
```

Amazon DynamoDB: 특정 테이블에 대한 액세스 허용

이 예제는 `MyTable` DynamoDB 테이블에 대한 모든 액세스를 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 `#### ## ### ###`를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

> **Important**
>
> 이 정책은 DynamoDB 테이블에서 수행 가능한 모든 작업을 허용합니다. 이러한 작업을 검토하려면 Amazon DynamoDB 개발자 안내서의 DynamoDB API 권한: 작업, 리소스 및 조건 참조를 참조하세요. 개별 작업 각각을 등록하여 동일한 권한을 제공할 수 있습니다. 그러나 `"dynamodb:List*"`와 같이 `Action` 요소에서 와일드카드(*)를 사용하는 경우, DynamoDB에서 새 목록 작업을 추가한다면 정책을 업데이트할 필요가 없습니다.

이 정책은 지정된 이름을 지닌 DynamoDB 테이블에 대해서만 작업을 허용합니다. DynamoDB에 있는 모든 것에 대한 Read 액세스 권한을 사용자에게 허용하려면 AmazonDynamoDBReadOnlyAccess AWS 관리형 정책을 연결할 수도 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListAndDescribe",
            "Effect": "Allow",
            "Action": [
                "dynamodb:List*",
                "dynamodb:DescribeReservedCapacity*",
                "dynamodb:DescribeLimits",
                "dynamodb:DescribeTimeToLive"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SpecificTable",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGet*",
                "dynamodb:DescribeStream",
                "dynamodb:DescribeTable",
                "dynamodb:Get*",
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:BatchWrite*",
                "dynamodb:CreateTable",
                "dynamodb:Delete*",
                "dynamodb:Update*",
                "dynamodb:PutItem"
            ],
            "Resource": "arn:aws:dynamodb:*:*:table/MyTable"
        }
    ]
}
```

Amazon DynamoDB: 특정 속성에 대한 액세스 허용

이 예제는 특정 DynamoDB 속성에 대한 액세스를 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데

필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

`dynamodb:Select` 요건을 설정하면 API 작업이 인덱스 프로젝션 등의 방법으로 허용되지 않는 속성을 반환할 수 없게 됩니다. DynamoDB 조건 키에 대한 자세한 정보는 Amazon DynamoDB 개발자 안내서의 조건 지정: 조건 키 사용 섹션을 참조하세요. IAM 정책의 `Condition` 블록 내에서 복수 조건 또는 복수 조건 키를 사용하는 방법에 관한 자세한 내용은 다수의 조건 값 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:BatchGetItem",
                "dynamodb:Query",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem",
                "dynamodb:BatchWriteItem"
            ],
            "Resource": ["arn:aws:dynamodb:*:*:table/table-name"],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:Attributes": [
                        "column-name-1",
                        "column-name-2",
                        "column-name-3"
                    ]
                },
                "StringEqualsIfExists": {"dynamodb:Select": "SPECIFIC_ATTRIBUTES"}
            }
        }
    ]
}
```

Amazon DynamoDB: Amazon Cognito ID를 기준으로 DynamoDB에 대한 항목 수준 액세스 허용

이 예제는 Amazon Cognito 자격 증명 풀 사용자 ID를 기준으로 `MyTable` DynamoDB 테이블에 대한 항목 수준 액세스를 허용하는 자격 증명 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS

API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

이 정책을 사용하려면 Amazon Cognito 자격 증명 풀 사용자 ID가 파티션 키가 되도록 DynamoDB 테이블을 구성해야 합니다. 자세한 내용은 Amazon DynamoDB 개발자 안내서의 테이블 생성을 참조하세요.

DynamoDB 조건 키에 대한 자세한 정보는 Amazon DynamoDB 개발자 안내서의 조건 지정: 조건 키 사용 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
            "Resource": ["arn:aws:dynamodb:*:*:table/MyTable"],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:LeadingKeys": ["${cognito-identity.amazonaws.com:sub}"]
                }
            }
        }
    ]
}
```

Amazon EC2: 태그를 기준으로 Amazon EBS 볼륨을 EC2 인스턴스에 연결 또는 분리

이 예제는 EBS 볼륨 소유자가 태그 `VolumeUser`를 사용하여 정의한 자신의 EBS 볼륨을 개발 인스턴스(`Department=Development`)로 태그가 지정된 EC2 인스턴스에 연결하거나 분리할 수 있도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

Amazon EC2 리소스에 대한 액세스를 제어하는 IAM 정책을 생성하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 리소스에 대한 액세스 제어를 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:DetachVolume"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Department": "Development"}
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:DetachVolume"
            ],
            "Resource": "arn:aws:ec2:*:*:volume/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/VolumeUser": "${aws:username}"}
            }
        }
    ]
}
```

Amazon EC2: 특정 서브넷에 있는 EC2 인스턴스를 프로그래밍 방식으로 콘솔에서 시작할 수 있도록 허용

이 예제는 모든 EC2 객체에 대한 정보를 나열하고 특정 서브넷에서 EC2 인스턴스를 시작하는 것을 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

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

```
        {
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "ec2:GetConsole*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:*:*:subnet/subnet-subnet-id",
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*::image/ami-*",
                "arn:aws:ec2:*:*:key-pair/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        }
    ]
}
```

Amazon EC2: 특정 태그 키 값 쌍이 있는 EC2 보안 그룹을 콘솔에서 프로그래밍 방식으로 관리할 수 있도록 허용

이 예제는 동일한 태그가 있는 보안 그룹에 대해 특정 작업을 수행할 수 있는 권한을 사용자에게 부여하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 Amazon EC2 콘솔에서 보안 그룹을 조회하고, 인바운드 및 아웃바운드 규칙을 추가 및 제거하고, `Department=Test` 태그가 있는 기존 보안 그룹에 대한 규칙 설명을 나열하고 수정할 권한을 부여합니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다. 이 정책을 사용하려면 정책 예제의 `#### ## ### ###`를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
         "ec2:DescribeSecurityGroups",
         "ec2:DescribeSecurityGroupRules",
         "ec2:DescribeTags"
```

```
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
         "ec2:AuthorizeSecurityGroupIngress",
         "ec2:RevokeSecurityGroupIngress",
         "ec2:AuthorizeSecurityGroupEgress",
         "ec2:RevokeSecurityGroupEgress",
         "ec2:ModifySecurityGroupRules",
         "ec2:UpdateSecurityGroupRuleDescriptionsIngress",
         "ec2:UpdateSecurityGroupRuleDescriptionsEgress"
      ],
      "Resource": [
         "arn:aws:ec2:region:111122223333:security-group/*"
      ],
      "Condition": {
         "StringEquals": {
            "aws:ResourceTag/Department": "Test"
         }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
         "ec2:ModifySecurityGroupRules"
      ],
      "Resource": [
         "arn:aws:ec2:region:111122223333:security-group-rule/*"
      ]
    }
  ]
}
```

Amazon EC2: 프로그래밍 방식으로 콘솔에서 사용자가 태그를 지정한 EC2 인스턴스를 시작 또는 중지할 수 있도록 허용

이 예제는 IAM 사용자가 EC2 인스턴스를 시작 또는 중지할 수 있도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 단, 인스턴스 태그 `Owner`에는 해당 사용자의 사용자 이름 값이 있어야 합니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Owner": "${aws:username}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeInstances",
            "Resource": "*"
        }
    ]
}
```

EC2: 태그를 기반으로 인스턴스 시작 또는 중지

이 예제는 태그 키 값 페어 `Project = DataAnalytics`로 인스턴스를 시작 또는 중지할 수 있도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 단, 태그 키 값 페어 `Department = Data`가 있는 보안 주체만 가능합니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 *#### ### ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

정책의 조건은 조건의 두 부분이 모두 true인 경우 true를 반환합니다. 인스턴스에 `Project=DataAnalytics` 태그가 있어야 합니다. 또한, 요청을 보내는 IAM 보안 주체(사용자나 역할)에 `Department=Data` 태그가 있어야 합니다.

> **Note**
>
> 가장 좋은 방법은 `aws:PrincipalTag` 조건 키가 있는 정책을 IAM 그룹에 연결하는 것입니다. 이 경우 일부 사용자는 지정된 태그가 있고 일부 사용자는 그렇지 않을 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "StartStopIfTags",
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:region:account-id:instance/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Project": "DataAnalytics",
                    "aws:PrincipalTag/Department": "Data"
                }
            }
        }
    ]
}
```

EC2: 일치하는 보안 주체 및 리소스 태그를 기반으로 인스턴스 시작 또는 중지

이 예제는 인스턴스의 리소스 태그와 보안 주체 태그가 CostCenter 태그 키와 동일한 값을 가질 때 Amazon EC2 인스턴스를 시작하거나 중지할 수 있는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 `#### ## ### ###`를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

> **Note**
>
> 가장 좋은 방법은 aws:PrincipalTag 조건 키가 있는 정책을 IAM 그룹에 연결하는 것입니다. 이 경우 일부 사용자는 지정된 태그가 있고 일부 사용자는 그렇지 않을 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "ec2:startInstances",
```

```
            "ec2:stopInstances"
        ],
        "Resource": "*",
        "Condition": {"StringEquals":
            {"aws:ResourceTag/CostCenter": "${aws:PrincipalTag/CostCenter}"}}
    }
}
```

Amazon EC2: 특정 리전 내에서의 모든 EC2 액세스를 프로그래밍 방식으로 콘솔에서 허용

이 예제는 특정 리전 내에서 모든 EC2 액세스를 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다. 리전 코드 목록은 Amazon EC2 사용 설명서의 사용 가능한 리전을 참조하세요.

또는 모든 Amazon EC2 API 작업에서 지원하는 글로벌 조건 키 `aws:RequestedRegion`을 사용할 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 예제: 특정 리전으로 액세스 제한을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "ec2:*",
            "Resource": "*",
            "Effect": "Allow",
            "Condition": {
                "StringEquals": {
                    "ec2:Region": "us-east-2"
                }
            }
        }
    ]
}
```

Amazon EC2: 프로그래밍 방식으로 콘솔에서 EC2 인스턴스를 시작 또는 중지하고 보안 그룹을 수정할 수 있도록 허용

이 예제는 특정 EC2 인스턴스를 시작 또는 중지하고 특정 보안 그룹을 수정할 수 있도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한

권한을 정의합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSecurityGroupReferences",
        "ec2:DescribeStaleSecurityGroups"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:instance/i-instance-id",
        "arn:aws:ec2:*:*:security-group/sg-security-group-id"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Amazon EC2: 특정 EC2 작업에 대해 MFA(GetSessionToken)를 요구

이 예제는 Amazon EC2 내에서 모든 AWS API 작업에 대한 액세스를 허용하는 아이덴티티 기반 정책을 작성하는 방법을 보여줍니다. 하지만 사용자가 멀티 팩터 인증(MFA)을 사용하여 인증되지 않은 경우 `StopInstances` 및 `TerminateInstances` API 작업에 대해 액세스는 명시적으로 거부합니다. 이를 프로그래밍 방식으로 수행하려면 사용자가 `GetSessionToken` 작업을 호출하는 동안 선택 사항인 `SerialNumber` 및 `TokenCode` 값을 포함해야 합니다. 이 작업은 MFA를 사용하여 인증된 임시 자

격 증명을 반환합니다. GetSessionToken에 대해 자세히 알아보려면 GetSessionToken - 신뢰할 수 없는 환경에 있는 사용자를 위한 임시 자격 증명 섹션을 참조하세요.

이 정책이 하는 일은 무엇입니까?

- AllowAllActionsForEC2 문은 모든 Amazon EC2 작업을 허용합니다.
- DenyStopAndTerminateWhenMFAIsNotPresent 문은 MFA 컨텍스트가 누락된 경우 StopInstances 및 TerminateInstances 작업을 거부합니다. 따라서 Multi-Factor Authentication(MFA) 컨텍스트가 누락된 경우(MFA가 사용되지 않은 경우) 작업이 거부됩니다. 거부는 허용을 무시합니다.

> **Note**
>
> MFA를 사용하지 않을 때는 키가 없어 키를 평가할 수 없기 때문에 Deny 문의 MultiFactorAuthPresent에 대한 조건 확인이 {"Bool": {"aws:MultiFactorAuthPresent":false}}이면 안 됩니다. 따라서 값을 확인하기 전에 BoolIfExists를 사용하여 키가 있는지 확인해야 합니다. 자세한 내용은 …IfExists 조건 연산자 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllActionsForEC2",
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        },
        {
            "Sid": "DenyStopAndTerminateWhenMFAIsNotPresent",
            "Effect": "Deny",
            "Action": [
                "ec2:StopInstances",
                "ec2:TerminateInstances"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {"aws:MultiFactorAuthPresent": false}
            }
```

```
        }
    ]
}
```

Amazon EC2: EC2 인스턴스 종료를 IP 주소 범위로 제한

이 예제는 작업을 허용하지만 요청이 지정된 IP 범위를 벗어나는 곳에서 오는 경우 액세스를 명시적으로 거부함으로써 EC2 인스턴스를 제한하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 회사의 IP 주소가 지정된 범위 내에 있는 경우에 유용합니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

이 정책을 `ec2:TerminateInstances` 작업을 허용하는 다른 정책(예: AmazonEC2FullAccess AWS 관리형 정책)과 조합하여 사용하는 경우 액세스가 거부됩니다. 이는 명시적 거부문이 허용문보다 우선 적용되기 때문입니다. 자세한 내용은 the section called “계정 내에서 요청 허용 여부 결정” 섹션을 참조하세요.

> **Important**
>
> `aws:SourceIp` 조건 키는 여러분을 대신하여 호출하는 AWS CloudFormation과 같은AWS 서비스에 대한 액세스를 거부합니다. `aws:SourceIp` 조건 키 사용에 관한 자세한 내용은 AWS 글로벌 조건 컨텍스트 키 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ec2:TerminateInstances"],
            "Resource": ["*"]
        },
        {
            "Effect": "Deny",
            "Action": ["ec2:TerminateInstances"],
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
```

```
                    "203.0.113.0/24"
                ]
            }
        },
        "Resource": ["*"]
    }
  ]
}
```

IAM: 정책 시뮬레이터 API에 액세스

이 예제는 현재 AWS 계정에서 사용자, 그룹 또는 역할에 연결된 정책에 대해 정책 시뮬레이터 API의 사용을 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 또한 이 정책은 API에 문자열로 전달되는 덜 민감한 정책을 시뮬레이션할 수 있도록 액세스를 허용합니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetContextKeysForCustomPolicy",
                "iam:GetContextKeysForPrincipalPolicy",
                "iam:SimulateCustomPolicy",
                "iam:SimulatePrincipalPolicy"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 사용자가 정책 시뮬레이터 콘솔에 액세스하여 현재 AWS 계정의 사용자, 그룹 또는 역할에 연결된 정책을 시뮬레이션하도록 허용하는 방법은 IAM: 정책 시뮬레이터 콘솔 액세스 섹션을 참조하세요.

IAM: 정책 시뮬레이터 콘솔 액세스

이 예제는 현재 AWS 계정에서 사용자, 그룹 또는 역할에 연결된 정책에 대해 정책 시뮬레이터 콘솔의 사용을 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다.

https://policysim.aws.amazon.com/에서 IAM 정책 시뮬레이터 콘솔에 액세스할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetGroup",
                "iam:GetGroupPolicy",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:GetUser",
                "iam:GetUserPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListAttachedRolePolicies",
                "iam:ListAttachedUserPolicies",
                "iam:ListGroups",
                "iam:ListGroupPolicies",
                "iam:ListGroupsForUser",
                "iam:ListRolePolicies",
                "iam:ListRoles",
                "iam:ListUserPolicies",
                "iam:ListUsers"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

IAM: 특정 태그가 있는 역할 수임

이 예제는 IAM 사용자가 태그 키 값 페어가 `Project = ExampleCorpABC`인 역할을 수임할 수 있도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI

에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 `#### ## ### ###`를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

이 태그가 포함된 역할이 사용자와 동일한 계정에 존재하는 경우 사용자는 해당 역할을 수임할 수 있습니다. 이 태그가 포함된 역할이 사용자가 아닌 다른 계정에 존재하는 경우 추가 권한이 필요합니다. 교차 계정 역할의 신뢰 정책에서 사용자 또는 사용자 계정의 모든 멤버가 역할을 수임하도록 허용해야 합니다. 교차 계정 액세스에 대한 역할 사용에 대한 자세한 정보는 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AssumeTaggedRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {"iam:ResourceTag/Project": "ExampleCorpABC"}
            }
        }
    ]
}
```

IAM: 프로그래밍 방식에서, 그리고 콘솔에서 여러 서비스에 대한 액세스를 허용 및 거부

이 예제는 IAM에서 여러 서비스에 대한 전체 액세스 및 제한적 자체 관리 액세스를 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. Amazon S3 `logs` 버킷 또는 Amazon EC2 `i-1234567890abcdef0` 인스턴스로의 액세스도 거부합니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다. 이 정책을 사용하려면 정책 예제의 `#### ## ### ###`를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

> **Warning**
>
> 이 정책은 여러 서비스의 모든 작업 및 리소스에 대한 전체 액세스를 허용합니다. 이 정책은 신뢰할 수 있는 관리자에게만 적용되어야 합니다.

이 정책을 권한 경계로 사용하여 자격 증명 기반 정책이 IAM 사용자에게 부여할 수 있는 최대 권한을 정의할 수 있습니다. 자세한 내용은 권한 경계를 사용하여 다른 사용자에게 책임 위임 섹션을 참조하세요. 정책이 사용자에 대한 권한 경계로 사용되는 경우 문에서 다음 경계를 정의합니다.

- `AllowServices` 문은 지정된 AWS 서비스에 대한 완전한 액세스를 허용합니다. 이런 서비스의 사용자 작업이 사용자에 연결된 권한 정책에 따라서만 제한된다는 의미입니다.
- `AllowIAMConsoleForCredentials` 문에서 모든 IAM 사용자를 나열할 수 있는 액세스를 허용합니다. 이 액세스는 AWS Management Console의 사용자 페이지를 탐색하는 데 필요합니다. 또한 계정의 암호 요구 사항을 확인하도록 허용합니다. 이 액세스는 사용자가 자신의 고유 암호를 변경할 때 필요합니다.
- `AllowManageOwnPasswordAndAccessKeys` 문은 사용자가 자신의 고유 콘솔 암호와 프로그래밍 방식의 액세스 키만 관리하도록 허용합니다. 이런 점은 중요합니다. 또 다른 정책에서 사용자에게 전체 IAM 액세스가 되는 권한 정책을 부여한다면 사용자 자신 또는 다른 사용자 권한을 변경할 수 있기 때문입니다. 이 설명문은 이런 상황을 방지할 수 있습니다.
- `DenyS3Logs` 설명문은 `logs` 버킷 액세스를 명시적으로 거부합니다. 이 정책은 사용자의 회사 제한을 적용합니다.
- `DenyEC2Production` 설명문은 `i-1234567890abcdef0` 인스턴스 액세스를 명시적으로 거부합니다.

이 정책은 다른 서비스 또는 작업에 대한 액세스를 허용하지 않습니다. 정책이 사용자에 대한 권한 경계로 사용되는 경우 사용자와 연결된 다른 정책에서 이러한 작업을 허용하더라도 AWS에서는 요청을 거부합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowServices",
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "cloudwatch:*",
                "ec2:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowIAMConsoleForCredentials",
```

```
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers",
                "iam:GetAccountPasswordPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswordAndAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:*AccessKey*",
                "iam:ChangePassword",
                "iam:GetUser",
                "iam:*LoginProfile*"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "DenyS3Logs",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::logs",
                "arn:aws:s3:::logs/*"
            ]
        },
        {
            "Sid": "DenyEC2Production",
            "Effect": "Deny",
            "Action": "ec2:*",
            "Resource": "arn:aws:ec2:*:*:instance/i-1234567890abcdef0"
        }
    ]
}
```

IAM: 특정 태그가 있는 사용자에게 특정 태그 추가

이 예제는 태그 키 Department를 태그 값 Marketing, Development 또는 QualityAssurance와 함께 IAM 사용자에게 추가할 수 있는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 사용자가 이미 태그 키 값 페어 JobFunction = manager를 포함해야 합니다. 이 정책을 사용하여 관리자가 세 부서 중 하나에 속하도록 요구할 수 있습니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권

한을 정의합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

ListTagsForAllUsers 문을 사용하면 계정의 모든 사용자에 대한 태그를 볼 수 있습니다.

TagManagerWithSpecificDepartment 문의 첫 번째 조건에는 StringEquals 조건 연산자가 사용됩니다. 이 조건은 조건의 두 부분이 모두 true인 경우 true를 반환합니다. 태그 지정될 사용자에게 이미 JobFunction=Manager 태그가 있어야 합니다. 나열된 태그 값 중 하나가 지정된 Department 태그 키가 요청에 포함되어야 합니다.

두 번째 조건에는 ForAllValues:StringEquals 조건 연산자가 사용됩니다. 이 조건은 요청의 모든 태그 키가 정책의 키와 일치하는 경우 true를 반환합니다. 즉 Department가 요청의 유일한 태그 키여야 합니다. ForAllValues 사용에 관한 자세한 내용은 다중 값 컨텍스트 키 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListTagsForAllUsers",
            "Effect": "Allow",
            "Action": [
                "iam:ListUserTags",
                "iam:ListUsers"
            ],
            "Resource": "*"
        },
        {
            "Sid": "TagManagerWithSpecificDepartment",
            "Effect": "Allow",
            "Action": "iam:TagUser",
            "Resource": "*",
            "Condition": {"StringEquals": {
                "iam:ResourceTag/JobFunction": "Manager",
                "aws:RequestTag/Department": [
                    "Marketing",
                    "Development",
                    "QualityAssurance"
                    ]
                },
                "ForAllValues:StringEquals": {"aws:TagKeys": "Department"}
            }
        }
    ]
```

```
}
```

IAM: 특정 값이 있는 특정 태그 추가

이 예제는 태그 키 CostCenter 및 태그 값 A-123 또는 태그 값 B-456을 가진 경우에만 IAM 사용자 또는 역할에 추가하도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책을 사용하여 특정 태그 키 및 태그 값 세트로 태그 지정을 제한할 수 있습니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다. 이 정책을 사용하려면 정책 예제의 `#### ## ### ###`를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

ConsoleDisplay 문을 사용하면 계정의 모든 사용자 및 역할에 대한 태그를 볼 수 있습니다.

AddTag 문의 첫 번째 조건에는 StringEquals 조건 연산자가 사용됩니다. 이 조건은 나열된 태그 값 중 하나가 지정된 CostCenter 태그 키가 요청에 포함된 경우 true를 반환합니다.

두 번째 조건에는 ForAllValues:StringEquals 조건 연산자가 사용됩니다. 이 조건은 요청의 모든 태그 키가 정책의 키와 일치하는 경우 true를 반환합니다. 즉 CostCenter가 요청의 유일한 태그 키여야 합니다. ForAllValues 사용에 관한 자세한 내용은 다중 값 컨텍스트 키 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ConsoleDisplay",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:GetUser",
                "iam:ListRoles",
                "iam:ListRoleTags",
                "iam:ListUsers",
                "iam:ListUserTags"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AddTag",
            "Effect": "Allow",
            "Action": [
                "iam:TagUser",
                "iam:TagRole"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CostCenter": [
                        "A-123",
                        "B-456"
                    ]
                },
                "ForAllValues:StringEquals": {"aws:TagKeys": "CostCenter"}
            }
        }
    ]
}
```

IAM: 특정 태그가 있는 새 사용자만 생성

이 예제는 IAM 사용자를 생성할 수 있도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 단, `Department` 및 `JobFunction` 태그 키를 하나 이상 사용해야 합니다. `Department` 태그 키에는 `Development` 또는 `QualityAssurance` 태그 값이 지정되어야 합니다. `JobFunction` 태그 키에는 `Employee` 태그 값이 지정되어야 합니다. 새 사용자가 특정 업무 기능 및 부서를 갖도록 하려면 이 정책을 사용하세요. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

이 설명문의 첫 번째 조건에는 `StringEqualsIfExists` 조건 연산자가 사용됩니다. 요청에 키가 `Department` 또는 `JobFunction`인 태그가 있는 경우 태그에 지정된 값이 있어야 합니다. 두 키가 모두 없으면 이 조건은 true로 평가됩니다. 조건이 false로 평가되는 유일한 경우는 지정된 조건 키 중 하나가 요청에 있지만 허용된 값이 아닌 다른 값이 지정된 경우뿐입니다. `IfExists` 사용에 관한 자세한 내용은 …IfExists 조건 연산자 섹션을 참조하세요.

두 번째 조건에는 `ForAllValues:StringEquals` 조건 연산자가 사용됩니다. 이 조건은 요청에서 지정한 각 태그 키 하나하나와 정책의 값 중 적어도 하나 이상이 일치하는 경우에만 true를 반환합니다. 즉 요청의 모든 태그가 이 목록에 있어야 합니다. 하지만 요청은 목록에 있는 태그 중 하나만 포함할 수 있습니다. 예를 들면 `Department=QualityAssurance` 태그만 지정된 IAM 사용자를 생성할 수 있습니다. 하지만 `JobFunction=employee` 태그와 `Project=core` 태그가 지정된 IAM 사용자는 생성할 수 없습니다. `ForAllValues` 사용에 관한 자세한 내용은 다중 값 컨텍스트 키 섹션을 참조하세요.

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

```
        {
            "Sid": "TagUsersWithOnlyTheseTags",
            "Effect": "Allow",
            "Action": [
                "iam:CreateUser",
                "iam:TagUser"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "aws:RequestTag/Department": [
                        "Development",
                        "QualityAssurance"
                    ],
                    "aws:RequestTag/JobFunction": "Employee"
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "Department",
                        "JobFunction"
                    ]
                }
            }
        }
    ]
}
```

IAM: IAM 자격 증명 보고서 생성 및 검색

이 예제는 사용자가 자신의 AWS 계정에 있는 모든 IAM 사용자를 나열한 보고서를 생성 및 다운로드하도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 보고서에는 암호, 액세스 키, MFA 디바이스, 서명 인증서를 포함한 사용자 자격 증명의 상태가 포함됩니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다.

자격 증명 보고서에 대한 자세한 정보는 섹션을 참조하세요AWS 계정의 자격 증명 보고서 생성

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:GenerateCredentialReport",
            "iam:GetCredentialReport"
```

```
        ],
        "Resource": "*"
    }
}
```

IAM: 프로그래밍 방식으로, 그리고 콘솔에서 그룹의 멤버십을 관리하도록 허용

이 예제는 MarketingTeam이라는 그룹의 멤버십을 업데이트할 수 있는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

이 정책이 하는 일은 무엇입니까?

- ViewGroups 문은 사용자에게 AWS Management Console의 모든 사용자와 그룹을 목록으로 나열하도록 허용합니다. 또한 사용자가 계정 내 사용자의 기본 정보를 볼 수 있도록 허용합니다. 이러한 권한은 리소스 ARN을 지원하지 않거나 리소스 ARN을 지정하는 데 필요하지 않기 때문에 자신의 문에 포함되어 있어야 합니다. 권한 대신 "Resource" : "*"를 지정합니다.
- ViewEditThisGroup 문은 사용자가 MarketingTeam 그룹에 대한 정보를 보고 해당 그룹에서 사용자를 제거할 수 있도록 허용합니다.

이 정책은 사용자가 사용자 또는 MarketingTeam 그룹의 권한을 보거나 편집하도록 허용하지 않습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewGroups",
            "Effect": "Allow",
            "Action": [
                "iam:ListGroups",
                "iam:ListUsers",
                "iam:GetUser",
                "iam:ListGroupsForUser"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ViewEditThisGroup",
```

```
            "Effect": "Allow",
            "Action": [
                "iam:AddUserToGroup",
                "iam:RemoveUserFromGroup",
                "iam:GetGroup"
            ],
            "Resource": "arn:aws:iam::*:group/MarketingTeam"
        }
    ]
}
```

IAM: 특정 태그 관리

이 예제는 IAM 엔터티(사용자 및 역할)의 태그 키 Department를 포함한 태그를 추가 및 제거할 수 있는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 Department 태그의 값을 제한하지 않습니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 #### ## ### ###를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:TagUser",
            "iam:TagRole",
            "iam:UntagUser",
            "iam:UntagRole"

        ],
        "Resource": "*",
        "Condition": {"ForAllValues:StringEquals": {"aws:TagKeys": "Department"}}
    }
}
```

IAM: IAM 역할을 특정 AWS 서비스로 전달

이 예제는 IAM 서비스 역할을 Amazon CloudWatch 서비스로 전달하도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 #### ## ## # ###를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

서비스 역할은 AWS 서비스를 역할을 수임할 수 있는 보안 주체로 지정하는 IAM 역할입니다. 이를 사용하면 서비스는 역할을 수임하고 사용자를 대신해 다른 서비스의 리소스에 액세스할 수 있습니다. Amazon CloudWatch에서 전달한 역할을 수임하도록 허용하려면 `cloudwatch.amazonaws.com` 서비스 보안 주체를 역할의 신뢰 정책 내 보안 주체로 지정해야 합니다. 서비스 보안 주체는 서비스가 정의합니다. 서비스의 보안 주체를 확인하려면 해당 서비스의 설명서를 참조하세요. 일부 서비스에서는 AWS IAM으로 작업하는 서비스를 참조하고 서비스 연결 역할 열에 예라고 표시된 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다. `amazonaws.com` 검색을 통해 서비스 보안 주체를 확인합니다.

서비스 역할을 서비스로 전달하는 것에 대해 자세히 알아보려면 사용자에게 AWS 서비스에 역할을 전달할 권한 부여 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {"iam:PassedToService": "cloudwatch.amazonaws.com"}
            }
        }
    ]
}
```

IAM: 보고 없이 IAM 콘솔에 대한 읽기 전용 액세스 허용

이 예제는 IAM 사용자가 `Get` 또는 `List` 문자열로 시작하는 IAM 작업을 수행하도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 사용자가 콘솔에서 작업할 경우 콘솔은 그룹, 사용자, 역할 및 정책 나열과 이러한 리소스에 대한 보고서 생성을 IAM에 요청합니다.

별표(*)는 와일드카드 역할을 합니다. 정책에서 `iam:Get*`을 사용할 때 결과 권한에는 으로 시작하는 모든 IAM 작업(예:`Get` `GetUser` 및 `GetRole`)이 포함됩니다. 와일드카드는 향후 새로운 유형의 엔터티가 IAM에 추가되는 경우 유용합니다. 이러한 경우 정책에서 부여한 권한은 자동으로 사용자가 새로운 엔터티에 대한 세부 정보를 나열하고 가져오도록 허용합니다.

이 정책은 보고서 또는 서비스에 마지막으로 액세스한 세부 정보를 생성하는 데 사용할 수 없습니다. 이를 허용하는 다른 정책에 대해서는 IAM: 콘솔에 대한 읽기 전용 액세스 허용 섹션을 참조하세요.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:Get*",
            "iam:List*"
        ],
        "Resource": "*"
    }
}
```

IAM: 콘솔에 대한 읽기 전용 액세스 허용

이 예제는 IAM 사용자가 `Get`, `List` 또는 `Generate` 문자열로 시작하는 IAM 작업을 수행하도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 사용자가 IAM 콘솔에서 작업할 경우 콘솔은 그룹, 사용자, 역할 및 정책 나열과 이러한 리소스에 대한 보고서 생성을 요청합니다.

별표(*)는 와일드카드 역할을 합니다. 정책에서 `iam:Get*`을 사용할 때 결과 권한에는 으로 시작하는 모든 IAM 작업(예:`Get` `GetUser` 및 `GetRole`)이 포함됩니다. 향후 새로운 유형의 엔터티가 IAM에 추가되는 경우 와일드카드를 사용하는 것이 이롭습니다. 이러한 경우 정책에서 부여한 권한은 자동으로 사용자가 새로운 엔터티에 대한 세부 정보를 나열하고 가져오도록 허용합니다.

보고서 또는 서비스에 마지막으로 액세스한 세부 정보를 생성할 수 있는 권한이 포함된 콘솔 액세스에 이 정책을 사용합니다. 작업 생성을 허용하지 않는 다른 정책에 대해서는 IAM: 보고 없이 IAM 콘솔에 대한 읽기 전용 액세스 허용 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:Get*",
            "iam:List*",
            "iam:Generate*"
        ],
        "Resource": "*"
    }
}
```

IAM: 특정 IAM 사용자가 프로그래밍 방식으로, 그리고 콘솔에서 그룹을 관리하도록 허용

이 예제는 특정 IAM 사용자가 `AllUsers` 그룹을 관리하도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

이 정책이 하는 일은 무엇입니까?

- `AllowAllUsersToListAllGroups` 문은 모든 그룹의 표시를 허용합니다. 이는 콘솔 액세스에 대해 필수입니다. 이 권한은 리소스 ARN을 지원하지 않으므로 자신의 문에 들어 있어야 합니다. 권한 대신 `"Resource" : "*"`를 지정합니다.
- `AllowAllUsersToViewAndManageThisGroup` 문은 그룹 리소스 유형에서 수행할 수 있는 모든 그룹 작업을 허용합니다. 사용자 리소스 유형에서는 수행할 수 있지만 그룹 리소스 유형에서 수행할 수 없는 `ListGroupsForUser` 작업은 허용하지 않습니다. IAM 작업에 지정할 수 있는 리소스 유형에 대한 자세한 내용은AWS Identity and Access Management 에서 사용되는 작업, 리소스 및 조건 키를 참조하세요.
- `LimitGroupManagementAccessToSpecificUsers` 문은 이름이 지정된 사용자의 쓰기 액세스와 그룹 작업 관리 권한을 거부합니다. 정책에 지정된 사용자가 그룹을 변경하려는 경우 문에서 이 요청을 거부합니다. 해당 요청은 `AllowAllUsersToViewAndManageThisGroup` 문에서 허용합니다. 다른 사용자가 이 작업을 수행하려는 경우 요청은 거부됩니다. IAM 콘솔에서 이 정책을 생성하는 동안 쓰기 또는 권한 관리로 정의된 IAM 작업을 볼 수 있습니다. 이를 수행하려면 JSON 탭에서 시각적 편집기 탭으로 전환합니다. 액세스 수준에 대한 자세한 내용은 AWS Identity and Access Management에서 사용되는 작업, 리소스 및 조건 키를 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllUsersToListAllGroups",
            "Effect": "Allow",
            "Action": "iam:ListGroups",
            "Resource": "*"
        },
        {
            "Sid": "AllowAllUsersToViewAndManageThisGroup",
            "Effect": "Allow",
```

```
            "Action": "iam:*Group*",
            "Resource": "arn:aws:iam::*:group/AllUsers"
        },
        {
            "Sid": "LimitGroupManagementAccessToSpecificUsers",
            "Effect": "Deny",
            "Action": [
                "iam:AddUserToGroup",
                "iam:CreateGroup",
                "iam:RemoveUserFromGroup",
                "iam:DeleteGroup",
                "iam:AttachGroupPolicy",
                "iam:UpdateGroup",
                "iam:DetachGroupPolicy",
                "iam:DeleteGroupPolicy",
                "iam:PutGroupPolicy"
            ],
            "Resource": "arn:aws:iam::*:group/AllUsers",
            "Condition": {
                "StringNotEquals": {
                    "aws:username": [
                        "srodriguez",
                        "mjackson",
                        "adesai"
                    ]
                }
            }
        }
    ]
}
```

IAM: 프로그래밍 방식으로, 그리고 콘솔에서 계정 암호 요구 사항을 설정하도록 허용

이 예제는 사용자가 계정의 암호 요구 사항을 보고 업데이트하도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 암호 요구 사항은 계정 멤버의 암호에 대한 복잡성 요구 사항 및 필수 교체 기산을 지정합니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다.

계정에 대해 암호 요구 사항 정책을 설정하는 방법을 알아보려면 IAM 사용자의 계정 암호 정책 설정 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": {
```

```
        "Effect": "Allow",
        "Action": [
            "iam:GetAccountPasswordPolicy",
            "iam:UpdateAccountPasswordPolicy"
        ],
        "Resource": "*"
    }
}
```

IAM: 사용자 경로를 바탕으로 정책 시뮬레이터 API 액세스

이 예제는 경로 Department/Development를 가진 사용자에 대해서만 정책 시뮬레이터 콘솔의 사용을 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetContextKeysForPrincipalPolicy",
                "iam:SimulatePrincipalPolicy"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:user/Department/Development/*"
        }
    ]
}
```

> **Note**
>
> 경로 Department/Development을 지닌 사용자에 대해 정책 시뮬레이터 콘솔의 사용을 허용하는 정책을 생성하는 방법은 IAM: 사용자 경로를 바탕으로 정책 시뮬레이터 콘솔 액세스 섹션을 참조하세요.

IAM: 사용자 경로를 바탕으로 정책 시뮬레이터 콘솔 액세스

이 예제는 경로 Department/Development를 가진 사용자에 대해서만 정책 시뮬레이터 콘솔의 사용을 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 #### ## ### ###를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

https://policysim.aws.amazon.com/에서 IAM 정책 시뮬레이터에 액세스할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetPolicy",
                "iam:GetUserPolicy"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "iam:GetUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListGroupsForUser",
                "iam:ListUserPolicies",
                "iam:ListUsers"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:user/Department/Development/*"
        }
    ]
}
```

IAM: IAM 사용자가 MFA 디바이스를 스스로 관리하도록 허용

이 예제는 IAM 사용자가 자신의 다중 인증(MFA) 디바이스를 자체 관리할 수 있도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다.

> **Note**
>
> 이 정책을 가진 IAM 사용자가 MFA 인증을 받지 않은 경우 이 정책은 MFA를 사용하여 인증하는 데 필요한 AWS 작업을 제외한 모든 해당 작업에 대한 액세스를 거부합니다. AWS에 로그인되어 있는 사용자에게 이들 권한을 추가하는 경우 사용자가 로그아웃한 다음 변경을 확인해야 할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowListActions",
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers",
                "iam:ListVirtualMFADevices"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUserToCreateVirtualMFADevice",
            "Effect": "Allow",
            "Action": [
                "iam:CreateVirtualMFADevice"
            ],
            "Resource": "arn:aws:iam::*:mfa/*"
        },
        {
            "Sid": "AllowUserToManageTheirOwnMFA",
            "Effect": "Allow",
            "Action": [
                "iam:EnableMFADevice",
                "iam:GetMFADevice",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowUserToDeactivateTheirOwnMFAOnlyWhenUsingMFA",
            "Effect": "Allow",
```

```
            "Action": [
                "iam:DeactivateMFADevice"
            ],
            "Resource": [
                "arn:aws:iam::*:user/${aws:username}"
            ],
            "Condition": {
                "Bool": {
                    "aws:MultiFactorAuthPresent": "true"
                }
            }
        },
        {
            "Sid": "BlockMostAccessUnlessSignedInWithMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:CreateVirtualMFADevice",
                "iam:EnableMFADevice",
                "iam:ListMFADevices",
                "iam:ListUsers",
                "iam:ListVirtualMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

IAM: 프로그래밍 방식을 사용하거나 콘솔에서 IAM 사용자가 자신의 보안 인증을 업데이트하도록 허용

이 예제는 IAM 사용자가 자신의 액세스 키, 서명 인증서, 서비스별 보안 인증 및 암호를 업데이트하도록 허용하는 자격 증명 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다.

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

```
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers",
                "iam:GetAccountPasswordPolicy"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:*AccessKey*",
                "iam:ChangePassword",
                "iam:GetUser",
                "iam:*ServiceSpecificCredential*",
                "iam:*SigningCertificate*"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        }
    ]
}
```

사용자가 콘솔에서 자신의 암호를 변경하는 방법에 대해서는 the section called “IAM 사용자가 자신의 암호를 변경하는 방법” 섹션을 참조하세요.

IAM: Organizations 정책에 대해 마지막으로 액세스한 서비스 정보 보기

이 예제는 특정 Organizations 정책에 대해 마지막으로 액세스한 서비스 정보를 볼 수 있도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 `p-policy123` ID가 포함된 서비스 제어 정책(SCP)에 대한 데이터 검색을 허용합니다. 보고서를 생성하고 보는 사람은 AWS Organizations 관리 계정 자격 증명을 사용하여 인증되어야 합니다. 이 정책은 요청자가 조직에 있는 조직 엔터티에 대한 데이터를 검색하도록 허용합니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

필요 권한, 문제 해결, 지원되는 리전을 포함하여 마지막으로 액세스한 정보에 대한 중요 정보는 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 섹션을 참조하세요.

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

```
            "Sid": "AllowOrgsReadOnlyAndIamGetReport",
            "Effect": "Allow",
            "Action": [
                "iam:GetOrganizationsAccessReport",
                "organizations:Describe*",
                "organizations:List*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowGenerateReportOnlyForThePolicy",
            "Effect": "Allow",
            "Action": "iam:GenerateOrganizationsAccessReport",
            "Resource": "*",
            "Condition": {
                "StringEquals": {"iam:OrganizationsPolicyId": "p-policy123"}
            }
        }
    ]
}
```

IAM: IAM 사용자, 그룹 또는 역할에 적용 가능한 관리형 정책을 제한

이 예제는 IAM 사용자, 그룹 또는 역할에 적용할 수 있는 고객 관리형 및 AWS 관리형 정책을 제한하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 `#### ## ### ###`를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:AttachUserPolicy",
            "iam:DetachUserPolicy"
        ],
        "Resource": "*",
        "Condition": {
            "ArnEquals": {
                "iam:PolicyARN": [
                    "arn:aws:iam::*:policy/policy-name-1",
                    "arn:aws:iam::*:policy/policy-name-2"
```

```
                ]
            }
        }
    }
}
```

AWS: AWS 관리형 IAM 정책을 제외한 계정 외부의 리소스에 대한 액세스 거부

아이덴티티 기반 정책에서 `aws:ResourceAccount`를 사용하면 서비스가 소유한 계정의 리소스와 상호 작용해야 하는 일부 서비스를 활용하는 사용자 또는 역할의 기능에 영향이 있을 수 있습니다.

AWS 관리형 IAM 정책을 허용하는 예외가 있는 정책을 생성할 수 있습니다. AWS Organizations 외부의 서비스 관리형 계정은 관리형 IAM 정책을 소유합니다.
AWS 관리형 정책을 나열하고 검색하는 네 가지 IAM 작업이 있습니다. 정책에서 `AllowAccessToS3ResourcesInSpecificAccountsAndSpecificService1` 문의 NotAction 요소에 이러한 작업을 사용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToResourcesInSpecificAccountsAndSpecificService1",
      "Effect": "Deny",
      "NotAction": [
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:ListEntitiesForPolicy",
        "iam:ListPolicies"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "111122223333"
          ]
        }
      }
    }
  ]
}
```

AWS Lambda: Lambda 함수가 Amazon DynamoDB 테이블에 액세스할 수 있도록 허용합니다.

이 예제는 특정 Amazon DynamoDB 테이블에 대한 읽기 및 쓰기 액세스를 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 또한 CloudWatch Logs에 대한 로그 파일 쓰기를 허용합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

이 정책을 사용하려면 정책을 Lambda 서비스 역할에 연결합니다. 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하도록 계정에 생성한 역할입니다. 이 서비스 역할에는 AWS Lambda가 신뢰 정책의 보안 주체로 포함되어야 합니다. 이 정책을 사용하는 방법에 대한 자세한 내용은 AWS 보안 블로그의 Amazon DynamoDB 테이블에 대한 AWS Lambda 액세스 권한을 부여하는 AWS IAM 정책을 생성하는 방법을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteTable",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGetItem",
                "dynamodb:GetItem",
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:BatchWriteItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem"
            ],
            "Resource": "arn:aws:dynamodb:*:*:table/SampleTable"
        },
        {
            "Sid": "GetStreamRecords",
            "Effect": "Allow",
            "Action": "dynamodb:GetRecords",
            "Resource": "arn:aws:dynamodb:*:*:table/SampleTable/stream/* "
        },
        {
            "Sid": "WriteLogStreamsAndGroups",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
```

```
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CreateLogGroup",
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "*"
        }
    ]
}
```

Amazon RDS: 특정 리전에 있는 RDS 데이터베이스에 대한 완전한 액세스 허용

이 예제는 특정 리전 내에서 모든 RDS 데이터베이스 액세스를 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds:*",
            "Resource": ["arn:aws:rds:region:*:*"]
        },
        {
            "Effect": "Allow",
            "Action": ["rds:Describe*"],
            "Resource": ["*"]
        }
    ]
}
```

Amazon RDS: 프로그램 방식으로 콘솔에서 RDS 데이터베이스를 복원하도록 허용

이 예제는 RDS 데이터베이스 복원을 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSnapshot",
                "rds:DeleteDBSnapshot",
                "rds:Describe*",
                "rds:DownloadDBLogFilePortion",
                "rds:List*",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyOptionGroup",
                "rds:RebootDBInstance",
                "rds:RestoreDBInstanceFromDBSnapshot",
                "rds:RestoreDBInstanceToPointInTime"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon RDS: 태그 소유자가 자신이 태그를 지정한 RDS 리소스에 대한 모든 액세스 권한을 가지도록 허용

이 예제는 태그가 지정된 RDS 리소스에 대한 모든 액세스를 태그 소유자에게 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "rds:Describe*",
                "rds:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
```

```
            "Action": [
                "rds:DeleteDBInstance",
                "rds:RebootDBInstance",
                "rds:ModifyDBInstance"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:db-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:ModifyOptionGroup",
                "rds:DeleteOptionGroup"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:og-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:ModifyDBParameterGroup",
                "rds:ResetDBParameterGroup"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:pg-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:AuthorizeDBSecurityGroupIngress",
                "rds:RevokeDBSecurityGroupIngress",
                "rds:DeleteDBSecurityGroup"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:secgrp-tag/Owner": "${aws:username}"}
            }
```

```
        },
        {
            "Action": [
                "rds:DeleteDBSnapshot",
                "rds:RestoreDBInstanceFromDBSnapshot"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:snapshot-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:ModifyDBSubnetGroup",
                "rds:DeleteDBSubnetGroup"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:subgrp-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:ModifyEventSubscription",
                "rds:AddSourceIdentifierToSubscription",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:DeleteEventSubscription"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:es-tag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

Amazon S3: Amazon Cognito 사용자가 버킷에 있는 객체에 액세스하도록 허용

이 예제는 Amazon Cognito 사용자가 특정 S3 버킷에 있는 객체에 액세스하도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 ${cognito-identity.amazonaws.com:sub} 변수로

표현되는 cognito, 애플리케이션 이름 및 페더레이션 사용자의 ID를 포함하는 이름을 통해 객체에 대한 액세스만을 허용합니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

> **Note**
>
> 객체 키에 사용된 '하위' 값은 사용자 풀의 사용자 하위 값이 아니라 자격 증명 풀의 사용자와 연결된 ID입니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListYourObjects",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": [
        "arn:aws:s3:::bucket-name"
      ],
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "cognito/application-name/${cognito-identity.amazonaws.com:sub}/*"
          ]
        }
      }
    },
    {
      "Sid": "ReadWriteDeleteYourObjects",
      "Effect": "Allow",
      "Action": [
        "s3:DeleteObject",
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name/cognito/application-name/${cognito-
identity.amazonaws.com:sub}/*"
      ]
    }
```

```
  ]
}
```

Amazon Cognito는 웹 및 모바일 앱에 대한 인증, 권한 부여 및 사용자 관리를 제공합니다. 사용자는 사용자 이름과 암호를 사용하여 직접 로그인하거나 Facebook, Amazon, 또는 Google 같은 타사를 통해 로그인할 수 있습니다.

Amazon Cognito의 두 가지 주요 구성 요소는 사용자 풀과 자격 증명 풀입니다. 사용자 풀은 앱 사용자의 가입 및 로그인 옵션을 제공하는 사용자 디렉터리입니다. 자격 증명 풀을 통해 사용자에게 기타 AWS 서비스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자격 증명 풀과 사용자 풀을 별도로 또는 함께 사용할 수 있습니다.

Amazon Cognito에 대한 자세한 내용은 Amazon Cognito 사용자 안내서를 참조하세요.

Amazon S3: 페더레이션 사용자가 프로그램 방식으로 콘솔에서 자신의 S3 홈 디렉터리에 액세스하도록 허용

이 예제는 페더레이션 사용자가 S3에 있는 자신의 홈 디렉터리 버킷 객체에 액세스하도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 홈 디렉터리는 개별 페더레이션 사용자의 home 폴더를 포함하는 버킷입니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

이 정책의 `${aws:userid}` 변수가 `role-id:specified-name`로 변환됩니다. 페더레이션 사용자 ID의 `role-id` 부분은 생성 중에 연합된 사용자의 역할에 할당된 고유한 식별자입니다. 자세한 내용은 고유 식별자 섹션을 참조하세요. `specified-name`는 페더레이션 사용자가 역할을 수임했을 때 `AssumeRoleWithWebIdentity` 요청으로 전달된 RoleSessionName 파라미터입니다.

AWS CLI 명령 `aws iam get-role --role-name` *`specified-name`*.을 사용하여 역할 ID를 볼 수 있습니다. 예를 들어, 기억하기 쉬운 이름 `John`을 지정하고 CLI가 역할 ID `AROAXXT2NJT7D3SIQN7Z6`를 반환한다고 가정해 봅시다. 이 경우 페더레이션 사용자 ID는 `AROAXXT2NJT7D3SIQN7Z6:John`입니다. 그러면 이 정책에서 페더레이션 사용자 John이 접두사 `AROAXXT2NJT7D3SIQN7Z6:John`로 시작하는 Amazon S3 버킷에 액세스할 수 있도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ConsoleAccess",
            "Effect": "Allow",
```

```
            "Action": [
                "s3:GetAccountPublicAccessBlock",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::bucket-name",
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "",
                        "home/",
                        "home/${aws:userid}/*"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::bucket-name/home/${aws:userid}",
                "arn:aws:s3:::bucket-name/home/${aws:userid}/*"
            ]
        }
    ]
}
```

Amazon S3: S3 버킷 액세스, 하지만 프로덕션 버킷은 최근 MFA 없이 거부

이 예제는 Amazon S3 관리자가 객체 업데이트, 추가 및 삭제를 포함하여 모든 버킷에 액세스하도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 하지만 사용자가 지난 30분 내에 멀티 팩터 인증(MFA)을 사용하여 로그인하지 않은 경우 `Production` 버킷에 대한 액세스를 명시적으로 거부합니다. 이 정책은 콘솔에서 또는 프로그래밍 방식으로 AWS CLI 또는 AWS API를 사용하여 이 작업

을 수행하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 `#### ## ### ###`를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

이 정책은 장기 사용자 액세스 키를 사용한 `Production` 버킷에 대한 프로그래밍 방식 액세스를 허용하지 않습니다. 이 작업은 `NumericGreaterThanIfExists` 조건 연산자와 함께 `aws:MultiFactorAuthAge` 조건 키를 사용해 수행됩니다. 이 정책 조건은 MFA가 없거나 MFA 사용 기간이 30분 이상인 경우 `true`를 반환합니다. 이러한 상황에서는 액세스가 거부됩니다. 프로그래밍 방식으로 `Production` 버킷에 액세스하려면 S3 관리자는 GetSessionToken API 작업을 사용하여 지난 30분 동안 생성된 임시 자격 증명을 사용해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListAllS3Buckets",
            "Effect": "Allow",
            "Action": ["s3:ListAllMyBuckets"],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowBucketLevelActions",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowBucketObjectActions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:s3:::*/*"
        },
        {
            "Sid": "RequireMFAForProductionBucket",
```

```
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::Production/*",
                "arn:aws:s3:::Production"
            ],
            "Condition": {
                "NumericGreaterThanIfExists": {"aws:MultiFactorAuthAge": "1800"}
            }
        }
    ]
}
```

Amazon S3: IAM 사용자가 프로그램 방식으로 콘솔에서 자신의 S3 홈 디렉터리에 액세스하도록 허용

이 예제는 IAM 사용자가 S3에 있는 자신의 홈 디렉터리 버킷 객체에 액세스하도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 홈 디렉터리는 개별 사용자의 home 폴더를 포함하는 버킷입니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다. 이 정책을 사용하려면 정책 예제의 `#### ## ### ###`를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

IAM 역할을 사용하는 경우 `aws:username` 변수를 사용할 수 없으므로 IAM 역할을 사용할 때는 이 정책이 작동하지 않습니다. 보안 주체 키 값에 대한 자세한 내용은 보안 주체 키 값 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ConsoleAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetAccountPublicAccessBlock",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
```

```
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::bucket-name",
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "",
                        "home/",
                        "home/${aws:username}/*"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::bucket-name/home/${aws:username}",
                "arn:aws:s3:::bucket-name/home/${aws:username}/*"
            ]
        }
    ]
}
```

Amazon S3: 특정 S3 버킷으로 관리 제한

이 예제는 Amazon S3 버킷의 관리를 해당 특정 버킷으로 제한하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 모든 Amazon S3 작업을 수행할 수 있는 권한을 부여하지만 Amazon S3를 제외한 모든 AWS 서비스에 대한 액세스는 거부합니다. 다음 예를 참조하세요. 이 정책에 따르면 S3 버킷 또는 S3 객체 리소스에 대해 수행할 수 있는 Amazon S3 작업에만 액세스할 수 있습니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

이 정책을 이 정책에서 거부하는 작업을 허용하는 다른 정책(예: AmazonS3FullAccess 또는 AmazonEC2FullAccess AWS 관리형 정책)과 조합하여 사용하는 경우 액세스가 거부됩니다. 이는 명시적 거부문이 허용문보다 우선 적용되기 때문입니다. 자세한 내용은 the section called “계정 내에서 요청 허용 여부 결정” 섹션을 참조하세요.

> **Warning**
>
> NotAction 및 NotResource는 신중히 사용해야 하는 고급 정책 요소입니다. 이 정책에서는 Amazon S3를 제외한 모든 AWS 서비스에 대한 액세스를 거부합니다. 이 정책을 사용자에게 연결할 경우 다른 서비스에 대한 권한을 부여하는 다른 정책은 무시되거나 액세스가 거부됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        },
        {
            "Effect": "Deny",
            "NotAction": "s3:*",
            "NotResource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

Amazon S3 버킷 객체에 대한 읽기/쓰기 액세스 권한 부여

이 예제는 Read 및 Write에서 특정 Amazon S3 버킷에 있는 객체에 대한 액세스를 허용하는 ID 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 AWS API 또는 AWS CLI에서 이러한 작업을 프로그래밍 방식으로 완료하는 데 필요한 권한을 부여합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

s3:*Object 작업에서는 와일드카드를 작업 이름의 일부로 사용합니다. AllObjectActions 문은 'Object' 단어로 끝나는 GetObject, DeleteObject, PutObject 및 기타 Amazon S3 작업을 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": ["s3:ListBucket"],
            "Resource": ["arn:aws:s3:::bucket-name"]
        },
        {
            "Sid": "AllObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object",
            "Resource": ["arn:aws:s3:::bucket-name/*"]
        }
    ]
}
```

> **Note**
>
> Amazon S3 버킷에 있는 객체에 대한 `Read` 및 `Write` 액세스 권한을 허용하고 콘솔 액세스에 대한 추가 권한을 포함하려면 Amazon S3: S3 버킷에 있는 객체에 대한 읽기 및 쓰기 액세스 권한을 프로그래밍 방식으로 콘솔에서 허용 섹션을 참조하세요.

Amazon S3: S3 버킷에 있는 객체에 대한 읽기 및 쓰기 액세스 권한을 프로그래밍 방식으로 콘솔에서 허용

이 예제는 `Read` 및 `Write`에서 특정 S3 버킷에 있는 객체에 대한 액세스를 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체하세요. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

`s3:*Object` 작업에서는 와일드카드를 작업 이름의 일부로 사용합니다. `AllObjectActions` 문은 'Object' 단어로 끝나는 `GetObject`, `DeleteObject`, `PutObject` 및 기타 Amazon S3 작업을 허용합니다.

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

```
        {
            "Sid": "S3ConsoleAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetAccountPublicAccessBlock",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": ["arn:aws:s3:::bucket-name"]
        },
        {
            "Sid": "AllObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object",
            "Resource": ["arn:aws:s3:::bucket-name/*"]
        }
    ]
}
```

IAM 정책 관리

IAM은 모든 유형의 IAM 정책(관리형 정책 및 인라인 정책)을 생성하고 관리하는 도구를 제공합니다. IAM 자격 증명(IAM 사용자, 그룹 또는 역할)에 권한을 추가하려면 정책을 생성하고 검증한 다음 정책을 자격 증명에 연결합니다. 다수의 정책을 자격 증명 하나에 연결하거나, 정책마다 다수의 권한이 포함될 수 있습니다.

자세한 내용은 다음 리소스를 참조하세요.

- 다양한 유형의 IAM 정책에 대한 자세한 정보는 IAM의 정책 및 권한 섹션을 참조하세요.
- IAM 내에서 정책 사용에 대한 일반적인 내용은 AWS리소스에 대한 액세스 관리 섹션을 참조하세요.

- 다수의 정책이 임의의 IAM 자격 증명 하나에 적용되는 경우 권한 평가 방법에 대한 자세한 내용은 정책 평가 로직 섹션을 참조하세요.
- AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 단원을 참조하십시오.

주제

- 고객 관리형 정책으로 사용자 지정 IAM 권한 정의
- IAM 정책 검증
- 액세스 활동을 기반으로 정책 생성
- IAM 정책 시뮬레이터로 IAM 정책 테스트
- IAM 자격 증명 권한 추가 및 제거
- IAM 정책 버전 관리
- IAM 정책 편집
- IAM 정책 삭제
- 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의

고객 관리형 정책으로 사용자 지정 IAM 권한 정의

정책은 AWS에서 ID 또는 리소스에 대한 권한을 정의합니다. AWS Management Console, AWS CLI 또는 AWS API를 사용하여 IAM에서 고객 관리형 정책을 생성할 수 있습니다. 고객 관리형 정책은 자체 AWS 계정에서 관리하는 독립형 정책입니다. 그런 다음 정책을 AWS 계정의 자격 증명(사용자, 그룹, 역할)에 연결합니다.

ID 기반 정책은 IAM의 ID에 연결되는 정책입니다. ID 기반 정책에는 AWS 관리형 정책, 고객 관리형 정책 및 인라인 정책이 포함될 수 있습니다. AWS 관리형 정책은 AWS에 의해 생성 및 관리되고, 사용자가 이를 사용할 수는 있지만 관리할 수는 없습니다. 인라인 정책은 사용자가 생성한 정책으로 IAM 사용자 그룹, 사용자 또는 역할에 직접 삽입할 수 있습니다. 인라인 정책은 다른 ID에서 재사용하거나 해당 ID 외부에서 관리할 수 없습니다. 자세한 내용은 IAM 자격 증명 권한 추가 및 제거 단원을 참조하십시오.

인라인 정책이나 AWS 관리형 정책 대신 고객 관리형 정책을 사용하는 것이 좋습니다. AWS 관리형 정책은 일반적으로 광범위한 관리 또는 읽기 전용 권한을 제공합니다. 보안을 극대화하려면 최소 권한을 부여하세요. 즉, 특정 작업을 수행하는 데 필요한 권한만 부여해야 합니다.

IAM 정책을 생성하거나 편집할 때 AWS는 정책 검증을 자동으로 수행하여 최소 권한을 염두에 두고 효과적인 정책을 생성할 수 있습니다. AWS Management Console에서 IAM은 JSON 구문 오류를 식별하는 반면, IAM Access Analyzer는 정책을 더욱 구체화하는 데 도움이 되는 권장 사항과 함께 추가 정책 검사를 제공합니다. 정책 검증에 대한 자세한 내용은 IAM 정책 검증 섹션을 참조하세요. IAM Access Analyzer 정책 확인 및 실행 가능한 권장 사항에 대한 자세한 내용은 IAM Access Analyzer 정책 검증을 참조하세요.

AWS Management Console, AWS CLI 또는 AWS API를 사용하여 IAM에서 고객 관리형 정책을 생성할 수 있습니다. AWS CloudFormation 템플릿을 사용하여 정책을 추가하거나 업데이트하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS Identity and Access Management resource type reference(IAM 리소스 유형 참조)를 참조하세요.

주제

- IAM 정책 생성(콘솔)
- IAM 정책 생성(AWS CLI)
- IAM 정책 생성(AWS API)

IAM 정책 생성(콘솔)

정책은 자격 증명 또는 리소스에 연결될 때 해당 권한을 정의하는 개체입니다. AWS Management Console을 사용하여 IAM에서 고객 관리형 정책을 생성할 수 있습니다. 고객 관리형 정책은 자체 AWS 계정에서 관리하는 독립형 정책입니다. 그런 다음 정책을 AWS 계정의 자격 증명(사용자, 그룹 또는 역할)에 연결합니다.

주제

- IAM 정책 생성
- JSON 편집기를 사용하여 정책 생성
- 시각적 편집기를 사용하여 정책 생성
- 기존 관리형 정책 가져오기

IAM 정책 생성

다음 방법 중 하나를 사용하여 AWS Management Console에서 고객 관리형 정책을 생성할 수 있습니다.

- JSON - 게시된 예시 자격 증명 기반 정책을 붙여넣고 사용자 지정합니다.

- 시각적 편집기 - 시각적 편집기에서 정책을 새로 생성합니다. 시각적 편집기를 사용할 경우 JSON 구문을 이해할 필요가 없습니다.
- 가져오기 - 계정 내에서 관리형 정책을 가져오고 사용자 지정합니다. 이전에 생성한 AWS 관리형 정책 또는 고객 관리형 정책을 가져올 수 있습니다.

AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 섹션을 참조하세요.

JSON 편집기를 사용하여 정책 생성

JSON 옵션을 선택하여 JSON에 정책을 입력하거나 붙여 넣을 수 있습니다. 이 방법은 계정에서 사용하기 위해 예시 정책을 복사할 경우 유용합니다. 또는 JSON 편집기에 고유한 JSON 정책 문서를 입력할 수 있습니다. JSON 옵션을 통해 시각적 편집기와 JSON 간에 전환하여 보기를 비교할 수도 있습니다.

JSON 편집기에서 정책을 생성하거나 편집할 때 IAM은 효과적인 정책을 생성하는 데 도움이 되는 정책 검증을 수행합니다. IAM은 JSON 구문 오류를 식별하는 반면, IAM Access Analyzer는 정책을 더욱 구체화하는 데 도움이 되는 실행 가능한 권장 사항과 함께 추가 정책 검사를 제공합니다.

JSON 정책 문서는 하나 이상의 문으로 구성되어 있습니다. 각 문에는 동일한 효과(`Allow` 또는 `Deny`)를 공유하며 동일한 리소스와 조건을 지원하는 모든 작업이 포함되어야 합니다. 한 작업에서 모든 리소스를 지정(`"*"`)하도록 요구하고 다른 작업에서 특정 리소스의 Amazon 리소스 이름(ARN)을 지원하는 경우 이들은 두 개의 별개 JSON 문에 있어야 합니다. ARN 형식에 대한 자세한 내용은 AWS 일반 참조 안내서의 Amazon 리소스 이름(ARN)을 참조하세요. IAM 정책에 대한 일반적인 내용은 IAM의 정책 및 권한 섹션을 참조하세요. IAM 정책 언어에 대한 정보는 IAM JSON 정책 참조 섹션을 참조하세요.

JSON 정책 편집기를 사용하여 정책을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 정책(Policies)을 선택합니다.
3. [정책 생성(Create policy)]을 선택합니다.
4. 정책 편집기 섹션에서 JSON 옵션을 선택합니다.
5. JSON 정책 문서를 입력하거나 붙여 넣습니다. IAM 정책 언어에 대한 자세한 정보는 IAM JSON 정책 참조 섹션을 참조하세요.
6. 정책 검증 동안 생성된 모든 보안 경고, 오류 또는 일반 경고를 해결하고 다음을 선택합니다.

> **Note**
>
> 언제든지 시각적 편집기 옵션과 JSON 편집기 옵션을 서로 전환할 수 있습니다. 그러나 변경을 적용하거나 시각적 편집기에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화되도록 정책을 재구성할 수 있습니다. 자세한 내용은 정책 재구성 섹션을 참조하세요.

7. (선택 사항) AWS Management Console에서 정책을 만들거나 편집할 때 AWS CloudFormation 템플릿에서 사용할 수 있는 JSON 또는 YAML 정책 템플릿을 생성할 수 있습니다.

   이렇게 하려면 정책 편집기에서 작업을 선택한 다음, CloudFormation 템플릿 생성을 선택합니다. AWS CloudFormation에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS Identity and Access Management리소스 유형 참조를 참조하세요.
8. 정책에 권한 추가를 완료했으면 다음을 선택합니다.
9. 검토 및 생성 페이지에서 생성하는 정책에 대한 정책 이름과 설명(선택 사항)을 입력합니다. 이 정책에 정의된 권한을 검토하여 정책이 부여한 권한을 확인합니다.
10. (선택 사항) 태그를 키 값 페어로 연결하여 메타데이터를 정책에 추가합니다. IAM에서의 태그 사용에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요.
11. 정책 생성(Create policy)을 선택하고 새로운 정책을 저장합니다.

정책을 생성한 후 그룹, 사용자 또는 역할에 연결할 수 있습니다. 자세한 내용은 IAM 자격 증명 권한 추가 및 제거 섹션을 참조하세요.

시각적 편집기를 사용하여 정책 생성

IAM 콘솔의 시각적 편집기는 JSON 구문을 작성하지 않고 정책을 생성하는 방법을 안내합니다. 시각적 편집기를 사용하여 정책을 생성하는 예시를 보려면 the section called “자격 증명에 대한 액세스 제어” 섹션을 참조하세요.

시각적 편집기를 사용하여 정책을 생성하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 정책(Policies)을 선택합니다.
3. [정책 생성(Create policy)]을 선택합니다.

4. 정책 편집기 섹션에서 서비스 선택 섹션을 찾은 다음, AWS 서비스를 선택합니다. 상단의 검색 상자를 사용하여 서비스 목록 결과를 제한할 수 있습니다. 시각적 편집기 권한 블록 내에서 하나의 서비스만 선택할 수 있습니다. 둘 이상의 서비스에 액세스 권한을 부여하려면 더 많은 권한 추가를 선택하여 여러 개의 권한 블록을 추가합니다.
5. 허용된 작업에서 정책에 추가할 작업을 선택합니다. 작업을 선택하는 방법은 다음과 같습니다.

   - 모든 작업에 대한 확인란을 선택합니다.
   - Add actions(작업 추가)를 선택하여 특정 작업의 이름을 입력합니다. 와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다.
   - 액세스 수준 그룹 중 하나를 선택하여 액세스 수준에 대한 모든 작업(예: 읽기, 쓰기 또는 목록)을 선택합니다.
   - 각 액세스 레벨 그룹을 확장하여 개별 작업을 선택합니다.

   기본적으로 생성되는 정책은 사용자가 선택하는 작업을 허용합니다. 대신 선택한 작업을 거부하려면 Switch to deny permissions(권한 거부로 전환)을 선택합니다. 기본적으로 IAM은 거부하기 때문에, 보안 모범 사례로 사용자에게 필요한 작업과 리소스에만 권한을 허용하는 것이 좋습니다. 다른 문 또는 정책에서 별도로 허용되는 권한을 재정의하려는 경우에만 권한을 거부하려면 JSON 문을 생성해야 합니다. 권한 거부의 수가 늘어나면 권한 문제를 해결하기가 더 어려워질 수 있기 때문에 그 수를 최소한으로 제한하는 것이 좋습니다.
6. 리소스에서 이전 단계에서 선택한 서비스 및 작업이 특정 리소스 선택을 지원하지 않는 경우에는 모든 리소스가 허용되며 이 섹션을 편집할 수 없습니다.

   리소스 수준 권한을 지원하는 작업을 하나 이상 선택하면 시각적 편집기에 해당 리소스가 나열됩니다. 그러면 리소스를 확장하여 정책에 대한 리소스를 지정할 수 있습니다.

   다음과 같은 방법으로 리소스를 지정할 수 있습니다.

   - ARN 추가를 선택하여 Amazon 리소스 이름(ARN)별로 리소스를 지정합니다. 시각적 ARN 편집기를 사용하거나 ARN을 수동으로 나열할 수 있습니다. ARN 구문에 대한 자세한 내용은 AWS 일반 참조 안내서의 Amazon 리소스 이름(ARN)을 참조하세요. 정책의 `Resource` 요소에 ARN을 사용하는 방법에 대한 자세한 내용은 IAM JSON 정책 요소: Resource 섹션을 참조하세요.
   - 리소스 옆의 이 계정에서 모두를 선택하여 해당 유형의 모든 리소스에 권한을 부여합니다.
   - 모두를 선택하여 해당 서비스에 대한 모든 리소스를 선택합니다.
7. (선택 사항) 요청 조건 - 선택 사항을 선택하여 생성하는 정책에 조건을 추가합니다. 조건은 JSON 정책 문의 효과를 제한합니다. 예를 들어 특정 시간 범위 내에 사용자의 요청이 발생하는 경우에만 사용자가 리소스에 대한 작업을 수행할 수 있도록 지정할 수 있습니다. 또한 일반적으로 사용되는

조건을 사용하여 사용자가 멀티 팩터 인증(MFA) 디바이스를 사용하여 인증받아야 하는지를 제한할 수 있습니다. 또는 요청이 특정 IP 주소 범위에서 발생하도록 요구할 수 있습니다. 정책 조건을 사용할 수 있는 컨텍스트 키의 전체 목록은 서비스 승인 참조의 AWS 서비스에 대한 작업, 리소스 및 조건 키를 참조하세요.

조건을 선택하는 방법은 다음과 같습니다.

- 확인란을 사용하여 일반적으로 사용되는 조건을 선택합니다.
- 다른 조건 추가를 선택하여 다른 조건을 지정합니다. 조건의 조건 키, 한정어, 연산자를 선택한 후 값을 입력합니다. 값을 두 개 이상 추가하려면 추가를 선택합니다. 해당 값이 논리적 "OR" 연산자로 연결되는 것으로 생각할 수 있습니다. 마치면 조건 추가를 선택합니다.

조건을 두 개 이상 추가하려면 다시 다른 조건 추가를 선택합니다. 필요에 따라 반복합니다. 각 조건은 이 시각적 편집기 권한 블록 하나에만 적용됩니다. 권한 블록이 일치하는 것으로 간주되려면 모든 조건이 true여야 합니다. 즉, 이들 조건이 논리적 "AND" 연산자로 연결되는 것으로 간주됩니다.

조건 요소에 대한 자세한 정보는 IAM JSON 정책 요소: Condition에서 IAM JSON 정책 참조 섹션을 참조하세요.

8. 더 많은 권한 블록을 추가하려면 더 많은 권한 추가를 선택합니다. 각 블록에 대해 2~5단계를 반복합니다.

> **Note**
>
> 언제든지 시각적 편집기 옵션과 JSON 편집기 옵션을 서로 전환할 수 있습니다. 그러나 변경을 적용하거나 시각적 편집기에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화되도록 정책을 재구성할 수 있습니다. 자세한 내용은 정책 재구성 섹션을 참조하세요.

9. (선택 사항) AWS Management Console에서 정책을 만들거나 편집할 때 AWS CloudFormation 템플릿에서 사용할 수 있는 JSON 또는 YAML 정책 템플릿을 생성할 수 있습니다.

   이렇게 하려면 정책 편집기에서 작업을 선택한 다음, CloudFormation 템플릿 생성을 선택합니다. AWS CloudFormation에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS Identity and Access Management리소스 유형 참조를 참조하세요.

10. 정책에 권한 추가를 완료했으면 다음을 선택합니다.
11. 검토 및 생성 페이지에서 생성하는 정책에 대한 정책 이름과 설명(선택 사항)을 입력합니다. 이 정책에 정의된 권한을 검토하여 의도한 권한을 부여했는지 확인합니다.

12. (선택 사항) 태그를 키 값 페어로 연결하여 메타데이터를 정책에 추가합니다. IAM에서의 태그 사용에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요.
13. 정책 생성(Create policy)을 선택하고 새로운 정책을 저장합니다.

정책을 생성한 후 그룹, 사용자 또는 역할에 연결할 수 있습니다. 자세한 내용은 IAM 자격 증명 권한 추가 및 제거 섹션을 참조하세요.

기존 관리형 정책 가져오기

새 정책을 생성하는 쉬운 방법은 최소한으로 필요한 권한 중 일부가 이미 존재하는 계정으로 기존 관리형 정책을 가져오는 것입니다. 그런 다음, 새로운 요구 사항에 일치하도록 정책을 사용자 지정할 수 있습니다.

인라인 정책은 가져올 수 없습니다. 관리형 정책과 인라인 정책의 차이에 대해 자세히 알아보려면 관리형 정책과 인라인 정책 섹션을 참조하세요.

시각적 편집기에서 기존 관리형 정책을 가져오려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 정책(Policies)을 선택합니다.
3. [정책 생성(Create policy)]을 선택합니다.
4. 정책 편집기에서 시각적 편집기를 선택한 다음, 페이지 오른쪽에서 작업을 선택하고 정책 가져오기를 선택합니다.
5. 정책 가져오기 창에서 새 정책에 포함할 정책과 가장 근접한 관리형 정책을 선택합니다. 상단의 검색 상자를 사용하여 정책 목록 결과를 제한할 수 있습니다.
6. 정책 가져오기를 선택합니다.

   가져온 정책은 정책 하단의 새 권한 블록에 추가됩니다.
7. Visual editor(시각적 편집기)를 사용하거나 JSON을 선택하여 정책을 사용자 지정합니다. 이후 다음을 선택합니다.

> **Note**
>
> 언제든지 시각적 편집기 옵션과 JSON 편집기 옵션을 서로 전환할 수 있습니다. 그러나 변경을 적용하거나 시각적 편집기에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화되도록 정책을 재구성할 수 있습니다. 자세한 내용은 정책 재구성 섹션을 참조하세요.

8. 검토 및 생성 페이지에서 생성하는 정책에 대한 정책 이름과 설명(선택 사항)을 입력합니다. 이러한 설정은 나중에 편집할 수 없습니다. 이 정책에 정의된 권한을 검토한 다음, 정책 생성을 선택하여 작업을 저장합니다.

JSON 편집기에서 기존 관리형 정책을 가져오려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 정책(Policies)을 선택합니다.
3. [정책 생성(Create policy)]을 선택합니다.
4. 정책 편집기 섹션에서 JSON 옵션을 선택한 다음, 페이지 오른쪽에서 작업을 선택하고 정책 가져오기를 선택합니다.
5. 정책 가져오기 창에서 새 정책에 포함할 정책과 가장 근접한 관리형 정책을 선택합니다. 상단의 검색 상자를 사용하여 정책 목록 결과를 제한할 수 있습니다.
6. 정책 가져오기를 선택합니다.

   가져온 정책의 문은 JSON 정책 하단에 추가됩니다.
7. JSON으로 정책을 사용자 지정합니다. 정책 검증 동안 생성된 모든 보안 경고, 오류 또는 일반 경고를 해결하고 다음을 선택합니다. 또는 시각적 편집기(Visual editor)에서 정책을 사용자 지정합니다. 이후 다음을 선택합니다.

> **Note**
>
> 언제든지 시각적 편집기 옵션과 JSON 편집기 옵션을 서로 전환할 수 있습니다. 그러나 변경을 적용하거나 시각적 편집기에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화되도록 정책을 재구성할 수 있습니다. 자세한 내용은 정책 재구성 섹션을 참조하세요.

8. 검토 및 생성 페이지에서 생성하는 정책에 대한 정책 이름과 설명(선택 사항)을 입력합니다. 이러한 필드는 나중에 편집할 수 없습니다. 이 정책에 정의된 권한 정책을 검토한 다음, 정책 생성을 선택하여 작업을 저장합니다.

정책을 생성한 후 그룹, 사용자 또는 역할에 연결할 수 있습니다. 자세한 내용은 IAM 자격 증명 권한 추가 및 제거 섹션을 참조하세요.

IAM 정책 생성(AWS CLI)

정책은 자격 증명 또는 리소스에 연결될 때 해당 권한을 정의하는 개체입니다. AWS CLI을 사용하여 IAM에서 고객 관리형 정책을 생성할 수 있습니다. 고객 관리형 정책은 자체 AWS 계정에서 관리하는 독립형 정책입니다. 가장 좋은 방법은 IAM Access Analyzer를 사용하여 IAM 정책을 검증하여 안전하고 기능적인 권한을 보장하는 것입니다. 정책을 검증하면 정책을 AWS 계정의 자격 증명(사용자, 그룹, 역할)에 연결하기 전에 오류나 권장 사항을 해결할 수 있습니다.

AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 섹션을 참조하세요.

IAM 정책 생성(AWS CLI)

AWS Command Line Interface(AWS CLI)를 사용하여 IAM 고객 관리형 정책 또는 인라인 정책을 생성할 수 있습니다.

고객 관리형 정책을 만들려면(AWS CLI)

다음 명령을 사용합니다.

- create-policy

IAM 자격 증명(그룹, 사용자 또는 역할)에 대한 인라인 정책을 만들려면(AWS CLI)

다음 명령 중 하나를 사용합니다.

- put-group-policy
- put-role-policy
- put-user-policy

> **Note**
>
> IAM을 사용하여 service-linked role에 인라인 정책을 포함시킬 수 없습니다.

고객 관리형 정책을 검증하려면(AWS CLI)

다음 IAM Access Analyzer 명령을 사용합니다.

- validate-policy

IAM 정책 생성(AWS API)

정책은 자격 증명 또는 리소스에 연결될 때 해당 권한을 정의하는 개체입니다. AWS API를 사용하여 IAM에서 고객 관리형 정책을 생성할 수 있습니다. 고객 관리형 정책은 자체 AWS 계정에서 관리하는 독립형 정책입니다. 가장 좋은 방법은 IAM Access Analyzer를 사용하여 IAM 정책을 검증하여 안전하고 기능적인 권한을 보장하는 것입니다. 정책을 검증하면 정책을 AWS 계정의 자격 증명(사용자, 그룹, 역할)에 연결하기 전에 오류나 권장 사항을 해결할 수 있습니다.

AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 섹션을 참조하세요.

IAM 정책 생성(AWS API)

AWS API를 사용하여 IAM 고객 관리형 정책 또는 인라인 정책을 생성할 수 있습니다.

고객 관리형 정책을 만들려면(AWS API)

다음 작업을 호출합니다.

- CreatePolicy

IAM 자격 증명(그룹, 사용자 또는 역할)에 대한 인라인 정책을 만들려면(AWS API)

다음 작업 중 하나를 호출합니다.

- PutGroupPolicy
- PutRolePolicy
- PutUserPolicy

> **Note**
>
> IAM을 사용하여 service-linked role에 인라인 정책을 포함시킬 수 없습니다.

고객 관리형 정책을 검증하려면(AWS API)

다음 IAM Access Analyzer 작업을 호출합니다.

- ValidatePolicy

IAM 정책 검증

여기에서 정책이란 IAM 정책 문법을 사용한 JSON 문서를 말합니다. 정책을 IAM 엔터티(예: 사용자, 그룹 또는 역할)에 연결하면 해당 엔터티에 권한을 부여합니다.

AWS Management Console을 사용해 IAM 액세스 제어 정책을 생성하거나 편집하는 경우 AWS는 IAM 정책 문법을 준수하는지 확인하기 위해 해당 정책을 자동으로 검사합니다. AWS에서 정책 문법을 준수하지 않은 정책을 발견하면 해당 정책을 수정하라는 메시지가 표시됩니다.

IAM Access Analyzer는 정책을 더욱 구체화하는 데 도움이 되는 권장 사항과 함께 추가 정책 검사를 제공합니다. IAM Access Analyzer 정책 검사기 및 실행 가능한 권장 사항에 대한 자세한 내용은 IAM Access Analyzer 정책 검증을 참조하세요. IAM Access Analyzer에서 반환된 경고, 오류 및 제안 사항 목록을 보려면 IAM Access Analyzer 정책 확인 참조를 참조하세요.

검증 대상

AWS에서는 JSON 정책 구문 및 문법을 검사합니다. 또한 ARN 형식이 올바른지와 작업 이름 및 조건 키가 올바른지도 확인합니다.

정책 검증 액세스

JSON 정책을 생성하거나 AWS Management Console에서 기존 정책을 편집하면 정책이 자동으로 검증됩니다. 정책 구문이 유효하지 않으면 알림이 표시되고 계속 진행하기 전에 문제를 해결해야 합니다. IAM Access Analyzer 정책 검증 결과는 `access-analyzer:ValidatePolicy`에 대한 권한이 있는 경우 자동으로 AWS Management Console에 반환됩니다. 또한 AWS API 또는 AWS CLI를 사용하여 정책을 검증할 수도 있습니다.

기존 정책

기존 정책이 정책 엔진에 대한 최신 업데이트 전에 생성되었거나 마지막으로 저장되었기 때문에 유효하지 않은 정책이 있을 수 있습니다. 가장 좋은 방법은 IAM Access Analyzer를 사용하여 IAM 정책을 검증하여 안전하고 기능적인 권한을 보장하는 것입니다. 기존 정책을 열고 생성된 정책 검증 결과를 검토하는 것이 좋습니다. 정책 구문 오류를 수정하지 않으면 기존 정책을 편집 및 저장할 수 없습니다.

액세스 활동을 기반으로 정책 생성

관리자나 개발자는 IAM 엔터티(사용자 또는 역할)에 필요한 것 이상의 권한을 부여할 수 있습니다. IAM에서는 부여하는 권한을 구체화하는 데 도움이 되는 몇 가지 옵션을 제공합니다. 그중 하나는 엔터티에 대한 액세스 활동을 기반으로 하는 IAM 정책을 생성하는 것입니다. IAM Access Analyzer는 사용자의 AWS CloudTrail 로그를 검토하고 지정된 날짜 범위에 엔터티에서 사용한 권한을 포함하는 정책 템플릿을 생성합니다. 템플릿을 사용하여 특정 사용 사례를 지원하는 데 필요한 권한만 부여하는 세분화된 권한을 가진 정책을 만들 수 있습니다.

예를 들어, 귀하가 개발자이고 엔지니어링 팀이 새 애플리케이션을 만들기 위해 프로젝트를 진행하고 있다고 가정해 보겠습니다. 실험을 장려하고 팀이 빠르게 진행할 수 있도록 하기 위해, 애플리케이션 개발 과정 중에 광범위한 권한을 가진 역할을 구성했습니다. 이제 애플리케이션을 프로덕션할 준비가 되었습니다. 프로덕션 계정에서 애플리케이션을 시작하기 전에, 애플리케이션이 작동하기 위해 역할에 필요한 권한만 식별하려고 합니다. 그러면 최소 권한 부여 모범 사례를 더욱 효과적으로 준수할 수 있습니다. 개발 계정의 애플리케이션에 대해 사용 중인 역할의 액세스 활동을 기반으로 정책을 생성할 수 있습니다. 생성된 정책을 추가로 구체화한 다음 프로덕션 계정의 엔터티에 정책을 연결할 수 있습니다.

IAM Access Analyzer 정책 생성에 대한 자세한 내용은 IAM Access Analyzer 정책 생성을 참조하세요.

IAM 정책 시뮬레이터로 IAM 정책 테스트

IAM 정책의 사용 방식과 이유에 대한 자세한 정보는 IAM의 정책 및 권한 섹션을 참조하세요.

https://policysim.aws.amazon.com/에서 IAM 정책 시뮬레이터 콘솔에 액세스할 수 있습니다.

> **Important**
>
> 정책 시뮬레이터 결과는 실제 AWS 환경과 다를 수 있습니다. 정책 시뮬레이터를 사용하여 테스트 후 정책을 실제 AWS 환경과 비교하여 원하는 결과가 얻었는지 확인하는 것이 좋습니다. 자세한 내용은 IAM 정책 시뮬레이터의 원리 섹션을 참조하세요.

IAM 정책 시뮬레이터 시작하기

IAM 정책 시뮬레이터를 사용하여 ID 기반 정책 및 IAM 권한 경계를 테스트하고 문제를 해결할 수 있습니다. 다음은 정책 시뮬레이터로 수행할 수 있는 몇 가지 일반적인 사항입니다.

- AWS 계정의 IAM 사용자, 사용자 그룹 또는 역할에 연결된 ID 기반 정책을 테스트합니다. 사용자, 사용자 그룹 또는 역할에 추가된 정책이 다수일 때는 모든 정책을 테스트하거나, 테스트할 정책만 따로

선택할 수 있습니다. 특정 리소스에 대해 선택한 정책에서 어떤 작업을 허용하거나 거부하는지 테스트할 수 있습니다.

- IAM 엔터티 권한 경계의 테스트 및 문제 해결 효과 한 번에 하나의 권한 경계만 시뮬레이션할 수 있습니다.
- Amazon S3 버킷, Amazon SQS 대기열, Amazon SNS 주제 또는 Amazon S3 Glacier 볼트와 같은 AWS 리소스에 연결된 IAM 사용자에 대한 리소스 기반 정책의 영향을 테스트합니다. IAM 사용자를 위해 정책 시뮬레이터에서 리소스 기반 정책을 사용하려면 시뮬레이션에 리소스를 포함해야 합니다. 또한 해당 리소스의 정책을 시뮬레이션에 포함하려면 확인란을 선택해야 합니다.

> **Note**
>
> IAM 역할에는 리소스 기반 정책의 시뮬레이션이 지원되지 않습니다.

- AWS 계정이 AWS Organizations의 조직에 속한 경우, 서비스 제어 정책(SCP)이 IAM 기반 정책에 미치는 영향을 테스트할 수 있습니다.

> **Note**
>
> 정책 시뮬레이터는 조건이 있는 SCP를 평가하지 않습니다.

- 사용자, 사용자 그룹 또는 역할에 아직 추가되지 않은 새로운 ID 기반 정책을 정책 시뮬레이터에 입력 또는 복사하여 테스트합니다. 이는 시뮬레이션에서만 사용되며 저장되지 않습니다. 리소스 기반 정책을 시뮬레이터에 입력하거나 복사하지 마세요.
- 선택한 서비스, 작업 및 리소스로 ID 기반 정책을 테스트합니다. 예를 들어 정책이 특정 버킷의 Amazon S3 서비스에서 주체가 `ListAllMyBuckets`, `CreateBucket` 및 `DeleteBucket` 작업을 수행할 수 있도록 허용하는지 확인하기 위해 테스트할 수 있습니다.
- 테스트할 정책에서 키가 지정되어 있는 경우에는 테스트할 정책의 `Condition` 요소에 포함된 IP 주소나 날짜 같은 콘텍스트 키를 제공하여 실제 시나리오를 시뮬레이션합니다.

> **Note**
>
> 시뮬레이션의 ID 기반 정책에 태그를 명시적으로 확인하는 `Condition` 요소가 없는 경우 정책 시뮬레이터는 입력으로 제공된 태그를 시뮬레이션하지 않습니다.

- ID 기반 정책에서 특정 리소스 또는 작업에 대한 액세스를 허용하거나 거부하는 특정 문을 식별합니다.

주제

IAM 정책 시뮬레이터의 원리

정책 시뮬레이터는 ID 기반 정책의 문과 시뮬레이션 중 사용자가 제공하는 입력을 평가합니다. 정책 시뮬레이터 결과는 실제 AWS 환경과 다를 수 있습니다. 정책 시뮬레이터를 사용하여 테스트 후 정책을 실제 AWS 환경과 비교하여 원하는 결과가 얻었는지 확인하는 것이 좋습니다.

정책 시뮬레이터는 라이브 AWS 환경과 다음과 같은 점에서 다릅니다.

- 정책 시뮬레이터는 실제로 AWS 서비스를 요청하는 것은 아니기 때문에 실행 중인 AWS 환경을 변경하지 않고 요청을 안전하게 테스트할 수 있습니다. 정책 시뮬레이터는 프로덕션에서 실제 컨텍스트 키 값을 고려하지 않습니다.
- 정책 시뮬레이터는 선택한 작업 중 실행 중인 작업은 시뮬레이션하지 않기 때문에 시뮬레이션된 요청에 대한 응답을 보고하지 않습니다. 요청된 작업이 허용되는지 아니면 거부되는지 여부만 결과로 반환됩니다.
- 정책 시뮬레이터에서 정책을 편집할 경우 이러한 변경은 정책 시뮬레이터에만 영향을 줍니다. AWS 계정의 해당 정책은 변함없이 그대로 유지됩니다.
- 조건이 있는 서비스 제어 정책(SCP)은 테스트할 수 없습니다.
- 정책 시뮬레이터는 크로스 계정 액세스를 위해 IAM 역할 및 사용자에 대한 시뮬레이션을 지원하지 않습니다.

> **Note**
>
> IAM 정책 시뮬레이터는 권한 부여를 위해 글로벌 조건 키를 지원하는 서비스를 결정하지 않습니다. 예를 들어, 정책 시뮬레이터는 서비스가 `aws:TagKeys`를 지원하지 않는다는 것을 식별하지 못합니다.

IAM 정책 시뮬레이터를 사용하는 데 필요한 권한

정책 시뮬레이터 콘솔 또는 정책 시뮬레이터 API를 사용하여 정책을 테스트할 수 있습니다. 기본적으로 콘솔 사용자는 사용자, 사용자 그룹 또는 역할에 아직 연결되지 않은 정책을 정책 시뮬레이터에 입력하거나 복사하여 테스트할 수 있습니다. 이러한 정책은 시뮬레이션에만 사용되며 민감한 정보를 공개하지 않습니다. API 사용자가 연결되지 않은 정책을 테스트하려면 권한이 있어야 합니다. 콘솔 또는 API 사용자가 AWS 계정의 IAM 사용자, 사용자 그룹 또는 역할에 연결된 정책을 테스트하도록 허용할 수 있습니다. 이렇게 하려면 해당 정책을 검색할 수 있는 권한을 제공해야 합니다. 리소스 기반 정책을 테스트하려면 사용자가 리소스의 정책을 검색할 수 있는 권한을 보유해야 합니다.

사용자가 시뮬레이션할 수 있는 콘솔 및 API 정책의 예시는 the section called “예제 정책 AWS Identity and Access Management(IAM)” 섹션을 참조하세요.

정책 시뮬레이터 콘솔을 사용하는 데 필요한 권한

사용자가 AWS 계정의 IAM 사용자, 사용자 그룹 또는 역할에 연결된 정책을 테스트하도록 허용할 수 있습니다. 이렇게 하려면 사용자에게 해당 정책을 검색할 수 있는 권한을 제공해야 합니다. 리소스 기반 정책을 테스트하려면 사용자가 리소스의 정책을 검색할 수 있는 권한을 보유해야 합니다.

사용자, 사용자 그룹 또는 역할에 연결된 정책에 대해 정책 시뮬레이터 콘솔의 사용을 허용하는 정책 예는 IAM: 정책 시뮬레이터 콘솔 액세스 섹션을 참조하세요.

특정 경로를 지닌 사용자에 대해서만 정책 시뮬레이터 콘솔의 사용을 허용하는 정책의 예시는 IAM: 사용자 경로를 바탕으로 정책 시뮬레이터 콘솔 액세스 섹션을 참조하세요.

한 가지 유형의 엔터티에 대해서만 정책 시뮬레이터 콘솔의 사용을 허용하는 정책을 만들려면 다음의 절차에 따릅니다.

콘솔 사용자가 사용자를 위한 정책을 시뮬레이션하도록 허용하는 방법

정책에 다음의 작업을 포함시킵니다.

- `iam:GetGroupPolicy`
- `iam:GetPolicy`
- `iam:GetPolicyVersion`
- `iam:GetUser`
- `iam:GetUserPolicy`
- `iam:ListAttachedUserPolicies`
- `iam:ListGroupsForUser`

- iam:ListGroupPolicies
- iam:ListUserPolicies
- iam:ListUsers

콘솔 사용자가 사용자 그룹을 위한 정책을 시뮬레이션하도록 허용하는 방법

정책에 다음의 작업을 포함시킵니다.

- iam:GetGroup
- iam:GetGroupPolicy
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:ListAttachedGroupPolicies
- iam:ListGroupPolicies
- iam:ListGroups

콘솔 사용자가 역할에 대한 정책을 시뮬레이션하도록 허용하는 방법

정책에 다음의 작업을 포함시킵니다.

- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetRole
- iam:GetRolePolicy
- iam:ListAttachedRolePolicies
- iam:ListRolePolicies
- iam:ListRoles

리소스 기반 정책을 테스트하려면 사용자가 리소스의 정책을 검색할 수 있는 권한을 보유해야 합니다.

콘솔 사용자가 Amazon S3 버킷에서 리소스 기반 정책을 테스트하도록 허용하는 방법

정책에 다음의 작업을 포함시킵니다.

- s3:GetBucketPolicy

예를 들어, 다음 정책에서 이 작업을 사용하여 특정 Amazon S3 버킷에서 콘솔 사용자가 리소스 기반 정책을 시뮬레이션하도록 허용합니다.

```
{
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Action": "s3:GetBucketPolicy",
            "Resource":"arn:aws:s3:::bucket-name/*"
          }
        ]
      }
```

API 정책 시뮬레이터를 사용하는 데 필요한 권한

정책 시뮬레이터 API 작업 GetContextKeyForCustomPolicy 및 SimulateCustomPolicy는 아직 사용자, 사용자 그룹 또는 역할에 연결되지 않은 정책을 테스트하도록 허용합니다. 이러한 정책을 테스트하려면 정책을 문자열로 API에 전달합니다. 이러한 정책은 시뮬레이션에만 사용되며 민감한 정보를 공개하지 않습니다. API를 사용하여 AWS 계정의 IAM 사용자, 사용자 그룹 또는 역할에 연결된 정책을 테스트할 수도 있습니다. 이렇게 하려면 사용자에게 GetContextKeyForPrincipalPolicy 및 SimulatePrincipalPolicy를 호출할 수 있는 권한을 제공해야 합니다.

현재 AWS 계정에 연결된 정책 및 연결되지 않은 정책에 대해 정책 시뮬레이터 API를 사용하도록 허용하는 예제 정책을 보려면 IAM: 정책 시뮬레이터 API에 액세스 섹션을 참조하세요.

한 가지 유형의 정책에 대해서만 정책 시뮬레이터 API의 사용을 허용하는 정책을 만들려면 다음의 절차에 따릅니다.

API 사용자가 API에 문자열로 직접 전달되는 정책을 시뮬레이션하도록 허용하는 방법

정책에 다음의 작업을 포함시킵니다.

- `iam:GetContextKeysForCustomPolicy`
- `iam:SimulateCustomPolicy`

API 사용자로 하여금 IAM 사용자, 사용자 그룹 또는 역할에 연결된 정책을 시뮬레이션하도록 하는 방법

정책에 다음의 작업을 포함시킵니다.

- `iam:GetContextKeysForPrincipalPolicy`
- `iam:SimulatePrincipalPolicy`

예를 들어, Alice라는 사용자에게 할당된 정책을 시뮬레이션할 수 있는 권한을 Bob이라는 사용자에게 부여하려면, Bob에게 `arn:aws:iam::777788889999:user/alice`라는 리소스에 액세스할 수 있는 권한을 부여해야 합니다.

특정 경로를 지닌 사용자에 대해서만 정책 시뮬레이터 API의 사용을 허용하는 정책의 예시는 IAM: 사용자 경로를 바탕으로 정책 시뮬레이터 API 액세스 섹션을 참조하세요.

IAM 정책 시뮬레이터 사용(콘솔)

기본적으로 사용자는 사용자, 사용자 그룹 또는 역할에 아직 연결되지 않은 정책을 정책 시뮬레이터 콘솔에 입력하거나 복사하여 테스트할 수 있습니다. 이러한 정책은 시뮬레이션에만 사용되며 민감한 정보를 공개하지 않습니다.

사용자, 사용자 그룹 또는 역할에 연결되지 않은 정책을 테스트하려면(콘솔)

1. https://policysim.aws.amazon.com/에서 IAM 정책 시뮬레이터 콘솔을 엽니다.
2. 페이지의 상단에 있는 Mode:(모드:) 메뉴에서 New Policy(새 정책)을 선택합니다.
3. Policy Sandbox(정책 샌드박스)에서 새 정책 생성을 선택합니다.
4. 정책 시뮬레이터에 입력하거나 복사하여 붙여 넣고, 다음 단계의 설명에 따라 정책 시뮬레이터를 사용합니다.

IAM 정책 시뮬레이터 콘솔을 사용할 권한이 있으면 정책 시뮬레이터를 사용하여 IAM 사용자, 사용자 그룹, 역할 또는 리소스 정책을 테스트할 수 있습니다.

사용자, 사용자 그룹 또는 역할에 연결된 정책을 테스트하려면(콘솔)

1. https://policysim.aws.amazon.com/에서 IAM 정책 시뮬레이터 콘솔을 엽니다.

> **Note**
>
> IAM 사용자로 정책 시뮬레이터에 로그인하려면 고유의 로그인 URL을 사용하여 AWS Management Console에 로그인합니다. 그런 다음 https://policysim.aws.amazon.com/으로 이동합니다. IAM 사용자 권한의 로그인에 대한 자세한 정보는 IAM 사용자가 AWS에 로그인하는 방법 섹션을 참조하세요.

정책 시뮬레이터가 Existing Policies(기존 정책) 모드로 열리며 Users, Groups, and Roles(사용자, 그룹 및 역할) 아래 계정에 속한 IAM 사용자가 표시됩니다.

2. 작업에 적합한 옵션을 선택합니다.

테스트 대상	수행할 작업:
사용자에게 연결된 정책	사용자, 그룹 및 역할(Users, Groups, and Roles) 목록에서 사용자(Users)를 선택합니다. 그런 다음 사용자를 선택합니다.
사용자 그룹에 연결된 정책	사용자, 그룹 및 역할(Users, Groups, and Roles) 목록에서 그룹(Groups)을 선택합니다. 그런 다음 사용자 그룹을 선택합니다.
역할에 연결된 정책	사용자, 그룹 및 역할(Users, Groups, and Roles) 목록에서 역할(Roles)을 선택합니다. 그런 다음 역할을 선택합니다.
리소스에 연결된 정책	Step 9 섹션을 참조하세요.
사용자, 사용자 그룹 또는 역할에 대한 사용자 지정 정책	새 정책 생성을 선택합니다. 새 정책 창에 정책을 입력하거나 붙여넣은 후 적용을 선택합니다.

> **도움말**
>
> 사용자 그룹에 연결된 정책을 테스트하려면 IAM 정책 시뮬레이터를 IAM 콘솔에서 직접 실행한 후 탐색 창에서 사용자 그룹을 선택합니다. 정책을 테스트하려는 그룹 이름을 선택한 후 권한 탭을 선택합니다. 시뮬레이션을 선택합니다.
> 사용자에게 연결된 고객 관리형 정책을 테스트하려면 탐색 창에서 사용자를 선택합니다. 정책을 테스트하고자 하는 사용자의 이름을 선택합니다. 그런 다음 권한 탭을 선택하고 테스트할 정책을 확장합니다. 오른쪽 맨 끝에서 Simulate Policy(정책 시뮬레이션)를 선택합니다. IAM 정책 시뮬레이터가 새 창으로 열리면서 선택한 정책을 정책 창에 표시합니다.

3. (선택 사항) 계정이 AWS Organizations의 조직 멤버인 경우 AWS Organizations SCP 옆에 있는 확인란을 선택하여 시뮬레이션 평가에 SCP를 포함시킬 수 있습니다. SCP는 조직 또는 조직 단위(OU)에 최대 권한을 지정하는 JSON 정책입니다. SCP는 멤버 계정의 엔터티에 대한 권한을 제한합니다. SCP가 서비스 또는 작업을 차단하는 경우 해당 계정에 있는 어떤 엔터티도 해당 서비스에

액세스하거나 해당 작업을 수행할 수 없습니다. 이는 관리자가 IAM 또는 리소스 정책을 통해 해당 서비스 또는 작업에 명시적으로 권한을 부여하는 경우에도 해당합니다.

계정이 조직의 멤버가 아닌 경우 확인란이 나타나지 않습니다.

4. (선택 사항) IAM 엔터티(사용자 또는 역할. 사용자 그룹은 제외)의 권한 경계로 설정된 정책을 테스트할 수 있습니다. 현재 엔터티에 대해 권한 경계 정책이 설정되어 있는 경우 정책 창에 표시됩니다. 한 엔터티에 대해 권한 경계 하나만 설정할 수 있습니다. 다른 권한 경계를 테스트하려면 사용자 지정 권한 경계를 만들면 됩니다. 이렇게 하려면 새 정책 생성을 선택합니다. 새 정책 창이 열립니다. 메뉴에서 사용자 지정 IAM 권한 경계 정책을 선택합니다. 새 정책의 이름을 입력하고 아래 공백에 정책을 입력하거나 복사합니다. 적용을 선택하여 정책을 저장합니다. 그런 다음 뒤로를 선택하여 원래 정책 창으로 돌아갑니다. 시뮬레이션에 사용할 권한 경계 옆의 확인란을 선택합니다.

5. (선택 사항) 사용자, 사용자 그룹 또는 역할에 연결된 정책의 하위 집합만 테스트할 수 있습니다. 이렇게 하려면 정책 창에서 제외할 각 정책 옆에 있는 확인란의 선택을 취소합니다.

6. Policy Simulator(정책 시뮬레이터)에서 Select service(서비스 선택)를 선택한 후 테스트할 서비스를 선택합니다. 그런 다음 Select actions(작업 선택)을 선택하고 테스트할 작업을 한 개 이상 선택합니다. 메뉴에는 한 번에 한 서비스에 대해 가능한 선택만 표시되지만 선택한 모든 서비스와 작업이 Action Settings and Results(작업 설정 및 결과)에 나타납니다.

7. (선택 사항) Step 2 및 Step 5에서 선택하는 정책 중 하나라도 AWS전역 조건 키를 지닌 조건을 포함하는 경우, 해당 키에 대한 값을 제공합니다. 글로벌 설정 섹션을 확장하고 표시된 키 이름의 값을 입력하여 키에 대한 값을 제공할 수 있습니다.

> **Warning**
>
> 조건 키의 값을 비워 놓으면 해당 키가 시뮬레이션 중에 무시됩니다. 이로 인해 오류가 발생하고 시뮬레이션이 실행되지 않는 경우가 있습니다. 또한 시뮬레이션은 실행되지만 결과를 신뢰할 수 없는 경우도 있습니다. 이 경우 조건 키에 대한 값이나 변수를 포함하는 실제 조건과 시뮬레이션이 일치하지 않습니다.

8. (선택 사항) 선택한 각 작업은 시뮬레이션을 실제로 실행할 때까지 권한 열에서 시뮬레이트되지 않음과 함께 동작 설정 및 결과에 나타납니다. 시뮬레이션을 실행하기 전에 리소스를 포함하는 각 작업을 구성할 수 있습니다. 특정 시나리오에 맞게 개별 작업을 구성하려면 화살표를 선택하여 작업 행을 확장합니다. 작업이 리소스 수준 권한을 지원할 경우 액세스를 테스트하려는 특정 리소스의 Amazon 리소스 이름(ARN)을 입력할 수 있습니다. 기본적으로 각 리소스는 와일드카드(*)로 설정됩니다. 또한 임의의 조건 콘텍스트 키에 대한 값을 지정할 수 있습니다. 앞에서도 설명했듯이 값이 비어 있는 키는 무시되며, 이로 인해 시뮬레이션이 실패하거나 신뢰할 수 없는 결과가 반환될 수 있습니다.

a. 작업 이름 옆에 있는 화살표를 선택하여 각 행을 확장하고 해당 시나리오에 맞게 작업을 정확하게 시뮬레이션하는 데 필요한 추가 정보를 구성합니다. 작업에 리소스 수준 권한이 필요할 경우 액세스를 시뮬레이션하려는 특정 리소스의 Amazon 리소스 이름(ARN)을 입력할 수 있습니다. 기본적으로 각 리소스는 와일드카드(*)로 설정됩니다.

b. 작업이 리소스 수준 권한을 지원하지만 그러한 권한이 필요하지 않을 경우 리소스 추가를 선택하여 시뮬레이션에 추가하려는 리소스 유형을 선택합니다.

c. 선택한 정책이 해당 작업의 서비스에 대한 콘텍스트 키를 참조하는 `Condition` 요소를 포함할 경우 해당 키 이름이 작업 아래에 표시됩니다. 지정한 리소스에 대한 해당 작업의 시뮬레이션 중에 사용할 값을 지정할 수 있습니다.

여러 리소스 유형 그룹이 필요한 작업

일부 작업은 서로 다른 환경에서 여러 리소스 유형이 필요합니다. 리소스 유형의 각 그룹은 시나리오와 관련이 있습니다. 이 중 하나가 시뮬레이션에 적용될 경우 리소스를 선택하면 정책 시뮬레이터가 해당 시나리오에 적합한 리소스 유형을 필요로 합니다. 다음 목록에는 지원되는 각 시나리오 옵션과 시뮬레이션을 실행하기 위해 정의해야 하는 리소스가 나와 있습니다.

다음의 Amazon EC2 시나리오 각각에 대해 `instance`, `image`, `security-group` 리소스를 지정해야 합니다. 시나리오에 EBS 볼륨이 포함될 경우에는 해당 `volume`을 리소스로 지정해야 합니다. Amazon EC2 시나리오에 Virtual Private Cloud(VPC)가 포함될 경우에는 `network-interface` 리소스를 제공해야 합니다. IP 서브넷이 포함될 경우에는 `subnet` 리소스를 지정해야 합니다. Amazon EC2 시나리오 옵션에 대한 자세한 내용은 Amazon EC2 사용 설명서의 지원되는 플랫폼을 참조하세요.

- EC2-VPC-InstanceStore

  인스턴스, 이미지, 보안 그룹, 네트워크 인터페이스

- EC2-VPC-InstanceStore-Subnet

  인스턴스, 이미지, 보안 그룹, 네트워크 인터페이스, 서브넷

- EC2-VPC-EBS

  인스턴스, 이미지, 보안 그룹, 네트워크 인터페이스, 볼륨

- EC2-VPC-EBS-Subnet

  인스턴스, 이미지, 보안 그룹, 네트워크 인터페이스, 서브넷, 볼륨

9. (선택 사항) 시뮬레이션에 리소스 기반 정책을 포함하려면 먼저 해당 리소스에 대해 시뮬레이션하려는 작업을 Step 6에서 선택해야 합니다. 선택한 작업의 행을 확장하고 시뮬레이션하려는 정책을 포함하는 리소스의 ARN을 입력합니다. 그런 다음 ARN 텍스트 상자 옆의 리소스 정책 포함(Include Resource Policy)을 선택합니다. IAM 정책 시뮬레이터는 현재, Amazon S3(리소스 기반 정책만 해당. ACL은 현재 지원되지 않음), Amazon SQS, Amazon SNS 및 잠겨 있지 않은 S3 Glacier 볼트(잠겨 있는 볼트는 현재 지원되지 않음) 서비스의 리소스 기반 정책만 지원합니다.
10. 상단 오른쪽 모서리 부분에서 Run Simulation(시뮬레이션 실행)을 선택합니다.

    동작 설정 및 결과 행의 각 권한 열에 지정된 리소스에 대한 해당 작업의 시뮬레이션 결과를 표시합니다.
11. 정책에서 작업을 명시적으로 허용하거나 거부한 문을 보려면 권한 열의 ***N***개 일치하는 문 링크를 선택하여 행을 확장합니다. Show statement(문 표시) 링크를 선택합니다. 정책 창에 해당 정책이 표시되고 시뮬레이션 결과에 영향을 준 문이 강조 표시됩니다.

> **Note**
>
> 작업이 암묵적으로 거부됨(즉, 명시적으로 허용되지 않기 때문에 작업이 거부된 경우)인 경우 나열(List) 및 문 표시(Show statement) 옵션은 표시되지 않습니다.

IAM 정책 시뮬레이터 콘솔 메시지 문제 해결

다음 표에는 IAM 정책 시뮬레이터 사용 시 나타날 수 있는 정보 메시지와 경고 메시지가 나와 있습니다. 그 밖에 문제 해결에 필요한 단계도 나와 있습니다.

메시지	문제 해결 단계
This policy has been edited. Changes will not be saved to your account.	작업이 필요하지 않음 이것은 정보 메시지입니다. IAM 정책 시뮬레이터에서 기존 정책을 편집하더라도 AWS 계정에서는 변경 사항이 적용되지 않습니다. 정책 시뮬레이터에서는 테스트 목적으로만 정책을 변경할 수 있습니다.
Cannot get the resource policy. 사유: ## ## ## ##	요청된 리소스 기반 정책에 정책 시뮬레이터가 액세스할 수 없습니다. 지정된 리소스 ARN이 정확하며, 시뮬레이션을 실행하는 사용자가 리소

메시지	문제 해결 단계
	스의 정책을 읽을 수 있는 권한이 있는지 확인하세요.
One or more policies require values in the simulation settings. The simulation might fail without these values.	이 메시지는 테스트하려는 정책에 포함되어 있는 조건 키 또는 변수 값을 Simulation Settings(시뮬레이션 설정)에 입력하지 않은 경우 나타납니다. 이 메시지를 닫으려면 Simulation Settings(시뮬레이션 설정)를 선택한 다음 각 조건 키 또는 변수에 대한 값을 입력합니다.
You have changed policies. These results are no longer valid.	이 메시지는 결과가 Results 창에 표시되는 중에 선택한 정책을 변경하였을 때 나타납니다. Results 창에 표시되는 결과는 동적으로 업데이트되지 않습니다. 이 메시지를 닫으려면 시뮬레이션 실행을 다시 선택하여 정책 창에서 변경한 내용을 기반으로 새로운 시뮬레이션 결과를 표시합니다.
The resource you typed for this simulation does not match this service.	이 메시지는 현재 시뮬레이션에서 선택한 서비스와 일치하지 않는 Amazon 리소스 이름(ARN)을 Simulation Settings(시뮬레이션 설정) 창에 입력했을 때 나타납니다. 예를 들어, Amazon DynamoDB 리소스의 ARN을 지정하고 시뮬레이션할 서비스로 Amazon Redshift를 선택하면 이 메시지가 나타납니다. 이 메시지를 닫으려면 다음 중 한 가지를 실행합니다. • Simulation Settings(시뮬레이션 설정) 창의 상자에서 ARN을 삭제합니다. • Simulation Settings(시뮬레이션 설정)에서 지정한 ARN과 일치하는 서비스를 선택합니다.

메시지	문제 해결 단계
이 작업은 Amazon S3 ACL 또는 S3 Glacier 볼트 잠금 정책 같은 리소스 기반 정책 외에 특수 액세스 제어 방식을 지원하는 서비스에 속합니다. The policy simulator does not support these mechanisms, so the results can differ from your production environment.	작업이 필요하지 않음 이것은 정보 메시지입니다. 현재 버전에서 정책 시뮬레이터는 사용자와 사용자 그룹에 연결된 정책을 평가하며,Amazon S3, Amazon SQS, Amazon SNS, S3 Glacier에 대한 리소스 기반 정책을 평가할 수 있습니다. 정책 시뮬레이터가 다른 AWS 서비스에서 지원하는 액세스 제어 방식을 모두 지원하는 것은 아닙니다.
DynamoDB FGAC is currently not supported.	작업이 필요하지 않음 이 정보 메시지는 세분화된 액세스 제어를 가리킵니다. 세분화된 액세스 제어는 IAM 정책 조건을 사용하여 DynamoDB 테이블 및 인덱스의 개별 데이터 항목과 속성에 액세스할 수 있는 사용자를 결정하는 기능입니다. 또한 이러한 테이블 및 인덱스에서 수행할 수 있는 작업을 나타내기도 합니다. 현재 버전의 IAM 정책 시뮬레이터는 이 유형의 정책 조건을 지원하지 않습니다. DynamoDB 세분화된 액세스 제어에 대한 자세한 정보는 DynamoDB에 대한 세분화된 액세스 제어를 참조하세요.
You have policies that do not comply with the policy syntax. 정책 검증을 사용해 정책에 대한 권장 사항 업데이트를 검토하고 수락할 수 있습니다.	이 메시지는 IAM 정책 문법을 위반하는 정책이 있는 경우 정책 목록 상단에 나타납니다. 이러한 정책을 시뮬레이션하려면 IAM 정책 검증의 정책 검증 옵션을 검토하여 해당 정책을 식별하고 수정합니다.
This policy must be updated to comply with the latest policy syntax rules.	이 메시지는 IAM 정책 문법을 위반하는 정책이 있는 경우에 표시됩니다. 이러한 정책을 시뮬레이션하려면 IAM 정책 검증의 정책 검증 옵션을 검토하여 해당 정책을 식별하고 수정합니다.

IAM 정책 시뮬레이터의 사용(AWS CLI 및 AWS API)

정책 시뮬레이터 명령어는 다음의 2가지 작업을 수행하는 데 일반적으로 API 작업 호출이 필요합니다.

1. 정책을 평가하고 정책이 참조하는 콘텍스트 키 목록을 반환합니다. 어떤 콘텍스트 키가 참조되는지 알아야 다음 단계에서 콘텍스트 키에 값을 제공할 수 있습니다.
2. 시뮬레이션 중에 사용되는 작업, 리소스, 콘텍스트 키의 목록을 제공하여 정책을 시뮬레이션합니다.

보안 상의 이유로 API 작업은 2개의 그룹으로 나뉘어 있습니다.

- API에 직접 문자열로 전달되는 정책만을 시뮬레이션하는 API 작업. 이 세트에는 GetContextKeysForCustomPolicy 및 SimulateCustomPolicy가 포함됩니다.
- 지정된 IAM 사용자, 사용자 그룹, 역할 또는 리소스에 연결된 정책을 시뮬레이션하는 API 작업. 이러한 API 작업은 다른 IAM 주체에 할당된 권한의 세부 정보를 알려주기 때문에 이 API 작업에 대한 액세스 제한을 고려해 보아야 합니다. 이 세트에는 GetContextKeysForPrincipalPolicy 및 SimulatePrincipalPolicy가 포함됩니다. API 작업 액세스 제한에 대한 자세한 정보는 예제 정책 AWS Identity and Access Management(IAM) 섹션을 참조하세요.

두 경우 모두 API 작업은 1개 이상의 정책들이 작업 및 리소스 목록에 미치는 영향을 시뮬레이션합니다. 각 작업은 각 리소스와 짝을 이루고, 시뮬레이션은 정책이 리소스에 대한 작업을 허용 또는 거부하는지 여부를 결정합니다. 또한, 정책이 참조하는 모든 콘텍스트 키에 대한 값을 제공할 수 있습니다. 정책이 참조하는 콘텍스트 키 목록은 `GetContextKeysForCustomPolicy` 또는 `GetContextKeysForPrincipalPolicy`를 호출하여 확인할 수 있습니다. 콘텍스트 키에 대한 값을 제공하지 않는다 해도 시뮬레이션은 여전히 실행되고 있지만, 정책 시뮬레이터가 평가 시에 컨텍스트 키를 포함할 수 없기 때문에 그 결과를 신뢰하지 못할 수 있습니다.

조건 키 목록을 확인하려면(AWS CLI, AWS API)

다음을 사용하여 정책 목록을 평가하고, 정책에 사용된 콘텍스트 키 목록을 반환합니다.

- AWS CLI: `aws iam get-context-keys-for-custom-policy` 및 `aws iam get-context-keys-for-principal-policy`
- AWS API: `GetContextKeysForCustomPolicy` 및 `GetContextKeysForPrincipalPolicy`

IAM 정책을 시뮬레이션하려면(AWS CLI, AWS API)

다음을 통해 IAM 정책을 시뮬레이션하여 사용자의 유효 권한을 확인합니다.

- AWS CLI: `aws iam simulate-custom-policy` 및 `aws iam simulate-principal-policy`
- AWS API: `SimulateCustomPolicy` 및 `SimulatePrincipalPolicy`

IAM 자격 증명 권한 추가 및 제거

정책을 사용하여 자격 증명(사용자, 사용자 그룹 또는 역할)에 대한 권한을 정의합니다. AWS Command Line Interface, AWS CLI(AWS Management Console) 또는 AWS API를 사용하여 자격 증명에 대한 IAM 정책을 첨부 및 분리하여 사용 권한을 추가 및 제거할 수 있습니다. 정책을 사용하여 동일한 방법을 사용하는 엔터티(사용자 또는 역할)에 대해서만 권한 경계를 설정할 수 있습니다. 권한 경계는 엔터티가 가질 수 있는 최대 권한을 제어하는 고급 AWS 기능입니다.

주제

- 용어
- 자격 증명 작업 보기
- IAM 자격 증명 권한 추가(콘솔)
- IAM 자격 증명 권한 제거(콘솔)
- IAM 정책 추가(AWS CLI)
- IAM 정책 제거(AWS CLI)
- IAM 정책 추가(AWS API)
- IAM 정책 제거(AWS API)

용어

권한 정책을 자격 증명(사용자, 사용자 그룹, 역할)과 연결할 때, 관리형 정책을 사용하는지 아니면 인라인 정책을 사용하는지에 따라 용어와 절차가 달라집니다.

- 연결 - 관리형 정책에 사용됩니다. 자격 증명(사용자, 사용자 그룹 또는 역할)에 관리형 정책을 연결합니다. 정책을 연결하면 정책의 해당 권한이 자격 증명에 적용됩니다.
- 분리 - 관리형 정책에 사용됩니다. IAM 자격 증명(사용자, 사용자 그룹, 역할)에서 관리형 정책을 분리합니다. 정책을 분리하면 자격 증명에서 해당 권한이 제거됩니다.
- 포함 - 인라인 정책에 사용됩니다. 자격 증명(사용자, 사용자 그룹 또는 역할)에 인라인 정책을 포함시킵니다. 정책을 포함하면 정책의 해당 권한이 자격 증명에 적용됩니다. 인라인 정책은 자격 증명에 저장되므로 결과는 비슷하지만 연결되지 않고 포함됩니다.

> **Note**
>
> 역할에 따라 달라지는 서비스에만 서비스 연결 역할에 대한 인라인 정책을 포함할 수 있습니다. 서비스가 이 기능을 지원하는지 여부를 확인하려면 서비스에 대한 AWS 설명서를 참조하세요.

- 삭제 - 인라인 정책에 사용됩니다. IAM 자격 증명(사용자, 사용자 그룹, 역할)에서 인라인 정책을 삭제합니다. 정책을 삭제하면 자격 증명에서 해당 권한이 제거됩니다.

> **Note**
>
> 역할에 따른 서비스에서만 서비스 연결 역할의 인라인 정책을 삭제할 수 있습니다. 서비스가 이 기능을 지원하는지 여부를 확인하려면 서비스에 대한 AWS 설명서를 참조하세요.

콘솔, AWS CLI 또는 AWS API를 사용하여 다음과 같은 작업을 수행할 수 있습니다.

추가 정보

- 관리형 정책과 인라인 정책의 차이점에 대한 자세한 정보는 관리형 정책과 인라인 정책 섹션을 참조하세요.
- 권한 경계에 대한 자세한 정보는 IAM 엔터티의 권한 범위 섹션을 참조하세요.
- IAM 정책에 대한 일반적인 내용은 IAM의 정책 및 권한 섹션을 참조하세요.
- IAM 검증 정책에 대한 자세한 내용은 IAM 정책 검증 섹션을 참조하세요.
- AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 섹션을 참조하세요.

자격 증명 작업 보기

자격 증명(사용자, 사용자 그룹 또는 역할)에 대한 사용 권한을 변경하기 전에 최근 서비스 수준 활동을 검토해야 합니다. 이 기능은 사용 중인 보안 주체(사람 또는 애플리케이션)의 액세스 권한을 제거하지 않으려는 경우 중요합니다. 마지막으로 액세스한 정보 보기에 대한 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 섹션을 참조하세요.

IAM 자격 증명 권한 추가(콘솔)

AWS Management Console을 사용하여 자격 증명(사용자, 사용자 그룹 또는 역할)에 권한을 추가할 수 있습니다. 이렇게 하려면 권한을 제어하는 관리형 정책을 연결하거나 권한 경계 역할을 하는 정책을 지정하세요. 인라인 정책을 포함할 수도 있습니다.

자격 증명에 대한 권한 정책으로서 관리형 정책을 사용하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Policies(정책)을 선택합니다.
3. 정책 목록에서 연결할 정책 이름 옆의 라디오 버튼을 선택합니다. 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 작업을 선택한 후 연결을 선택합니다.
5. 정책을 연결할 자격 증명을 하나 이상 선택합니다. 검색 상자를 사용하면 보안 주체 엔터티 목록을 필터링할 수 있습니다. 자격 증명을 선택한 후 정책 연결을 선택합니다.

보안 경계(콘솔)를 설정하기 위해서 관리형 정책을 사용하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Policies(정책)을 선택합니다.
3. 정책 목록에서 설정할 정책 이름을 선택합니다. 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 정책 세부 정보 페이지에서 연결된 엔터티 탭을 선택하고 필요하다면 권한 경계로 연결 섹션을 열고 이 정책을 권한 경계로 설정을 선택합니다.
5. 권한 경계에 대한 정책이 사용될 하나 이상의 사용자 또는 역할을 선택하세요. 검색 상자를 사용하면 보안 주체 엔터티 목록을 필터링할 수 있습니다. 보안 주체를 선택한 후 권한 경계 설정을 선택합니다.

사용자 또는 역할의 인라인 정책을 포함하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

2. 탐색 창에서 사용자 또는 역할을 선택합니다.
3. 목록에서 정책을 삽입할 그룹, 사용자 또는 역할 이름을 선택합니다.
4. 권한(Permissions) 탭을 선택합니다.
5. 권한 추가를 선택하고 인라인 정책 생성을 선택합니다.

> **Note**
>
> IAM의 서비스 연결 역할에는 인라인 정책을 포함할 수 없습니다. 링크된 서비스가 역할 권한을 수정할 수 있는지 여부를 결정하기 때문에 서비스 콘솔이나 API 또는 AWS CLI에서 정책을 추가할 수 있습니다. 서비스에 대한 서비스 연결 역할 설명서를 보려면 AWS IAM으로 작업하는 서비스을 참조하고 사용하는 서비스의 서비스 연결 역할(Service-Linked Role) 열에서 예(Yes)를 선택합니다.

6. 다음 방법 중에서 선택하여 정책을 생성하는 데 필요한 단계를 볼 수 있습니다.

   - 기존 관리형 정책 가져오기 - 계정으로 관리형 정책을 가져온 다음 정책을 편집하여 특정 요구 사항에 맞게 사용자 지정할 수 있습니다. 관리형 정책은 사용자가 이전에 생성한 고객 관리형 정책이거나 AWS 관리형 정책일 수 있습니다.
   - 시각적 편집기를 사용하여 정책 생성 - 시각적 편집기에서 정책을 새로 생성할 수 있습니다. 시각적 편집기를 사용할 경우 JSON 구문을 이해할 필요가 없습니다.
   - JSON 편집기를 사용하여 정책 생성 - JSON 편집기 옵션에서 JSON 구문을 사용하여 정책을 생성할 수 있습니다. 새 JSON 정책 문서를 입력하거나 예시 정책을 붙여 넣을 수 있습니다.

7. 인라인 정책을 생성하고 나면 이 정책이 사용자나 역할에 자동으로 포함됩니다.

사용자 그룹의 인라인 정책을 포함하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자 그룹(User groups)을 선택합니다.
3. 목록에서 정책을 삽입할 사용자 그룹 이름을 선택합니다.
4. 권한(Permissions) 탭을 선택하고 권한 추가(Add permissions), 인라인 정책 생성(Create inline policy)을 차례로 선택합니다.
5. 다음 중 하나를 수행하세요.

- 시각적 편집기 옵션에서 정책을 생성하도록 선택합니다. 자세한 내용은 시각적 편집기를 사용하여 정책 생성 섹션을 참조하세요.
- JSON 옵션에서 정책을 생성하도록 선택합니다. 자세한 내용은 JSON 편집기를 사용하여 정책 생성 섹션을 참조하세요.

6. 정책이 마음에 들면 Create policy(정책 생성)를 선택합니다.

하나 이상의 엔터티에 대한 권한 경계 설정을 변경하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Policies(정책)을 선택합니다.
3. 정책 목록에서 설정할 정책 이름을 선택합니다. 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 정책 세부 정보 페이지에서 연결된 엔터티 탭을 선택하고 필요하다면 권한 경계로 연결 섹션을 엽니다. 변경할 경계의 사용자 또는 역할 옆에 있는 확인란을 선택한 후 변경을 선택합니다.
5. 새로운 정책을 선택하여 권한 경계를 사용하세요. 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다. 정책을 선택한 후 권한 경계 설정을 선택합니다.

IAM 자격 증명 권한 제거(콘솔)

AWS Management Console을 사용하여 자격 증명(사용자, 사용자 그룹 또는 역할)에서 권한을 제거할 수 있습니다. 이렇게 하려면 권한을 제어하는 관리형 정책을 분리하거나 권한 경계 역할을 하는 정책을 제거하세요. 인라인 정책을 삭제할 수도 있습니다.

권한 정책(콘솔)으로서 사용된 관리형 정책을 분리하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Policies(정책)을 선택합니다.
3. 정책 목록에서 분리할 정책 이름 옆의 라디오 버튼을 선택합니다. 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 작업(Actions)을 선택한 후 분리(Detach)를 선택합니다.
5. 정책을 분리할 자격 증명을 선택합니다. 검색 상자를 사용하여 자격 증명 목록을 필터링할 수 있습니다. 자격 증명을 선택한 후 Detach policy(정책 분리)를 선택합니다.

권한 경계(콘솔)를 제거하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Policies(정책)을 선택합니다.
3. 정책 목록에서 설정할 정책 이름을 선택합니다. 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 정책 세부 정보 페이지에서 연결된 엔터티 탭을 선택하고 필요하다면 권한 경계로 연결 섹션을 열고 권한 경계에서 제거할 엔터티를 선택합니다. 그런 다음 경계 제거를 선택합니다.
5. 경계 제거를 선택하여 경계를 제거합니다.

인라인 정책을 삭제하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자 그룹, 사용자 또는 역할을 선택합니다.
3. 목록에서 제거할 정책이 있는 사용자 그룹, 사용자 또는 역할 이름을 선택합니다.
4. 권한(Permissions) 탭을 선택합니다.
5. 정책 옆에 있는 확인란을 선택하고 제거를 선택합니다.
6. 확인 상자가 나타나면 제거을 선택합니다.

IAM 정책 추가(AWS CLI)

AWS CLI을 사용하여 자격 증명(사용자, 사용자 그룹 또는 역할)에 권한을 추가할 수 있습니다. 이렇게 하려면 권한을 제어하는 관리형 정책을 연결하거나 권한 경계 역할을 하는 정책을 지정하세요. 인라인 정책을 포함할 수도 있습니다.

엔터티에 대한 권한 정책으로서 관리형 정책을 사용하려면(AWS CLI)

1. (선택 사항) 관리형 정책에 대한 정보를 보려면 다음 명령을 실행합니다.
   - 관리형 정책의 목록 보기: aws iam list-policies
   - 관리형 정책에 대한 세부 정보 가져오기: get-policy

2. 관리형 정책을 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결하려면 다음 명령 중 하나를 사용합니다.

   - aws iam attach-user-policy
   - aws iam attach-group-policy
   - aws iam attach-role-policy

보안 경계(AWS CLI)를 설정하기 위해서 관리형 정책을 사용하려면

1. (선택 사항) 관리형 정책에 대한 정보를 보려면 다음 명령을 실행합니다.

   - 관리형 정책의 목록 보기: aws iam list-policies
   - 관리형 정책에 대한 세부 정보 가져오기: aws iam get-policy
2. 관리형 정책을 사용하여 엔터티(사용자 또는 역할)에 대한 권한 경계를 설정하려면 다음 명령 중 하나를 사용합니다.

   - aws iam put-user-permissions-boundary
   - aws iam put-role-permissions-boundary

인라인 정책을 포함시키려면(AWS CLI)

인라인 정책을 자격 증명(사용자, 사용자 그룹 또는 서비스 연결 역할이 아닌 역할)에 포함시키려면 다음 명령 중 하나를 사용합니다.

- aws iam put-user-policy
- aws iam put-group-policy
- aws iam put-role-policy

IAM 정책 제거(AWS CLI)

AWS CLI를 사용하여 권한을 제어하는 관리형 정책을 분리하거나 권한 경계 역할을 하는 정책을 제거할 수 있습니다. 인라인 정책을 삭제할 수도 있습니다.

권한 정책(AWS CLI)으로서 사용된 관리형 정책을 분리하려면

1. (선택 사항)정책에 대한 정보를 보려면 다음 명령을 실행합니다.

   - 관리형 정책의 목록 보기: aws iam list-policies

- 관리형 정책에 대한 세부 정보 가져오기: aws iam get-policy

2. (선택 사항) 정책과 자격 증명 간의 관계에 대해 확인하려면 다음 명령을 실행합니다.

   - 관리형 정책이 연결된 자격 증명(사용자, 사용자 그룹 및 역할)의 목록을 보려면 다음과 같이 합니다.
     - aws iam list-entities-for-policy
   - 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결된 관리형 정책의 목록을 보려면 다음 명령 중 하나를 사용합니다.
     - aws iam list-attached-user-policies
     - aws iam list-attached-group-policies
     - aws iam list-attached-role-policies
3. 관리형 정책을 자격 증명(사용자, 사용자 그룹 또는 역할)에서 분리하려면 다음 명령 중 하나를 사용합니다.

   - aws iam detach-user-policy
   - aws iam detach-group-policy
   - aws iam detach-role-policy

권한 경계(AWS CLI)를 제거하려면

1. (선택 사항) 현재 어떤 관리형 정책을 사용하여 사용자 또는 역할에 대한 권한 경계를 설정하는지 보려면 다음 명령을 실행하세요.

   - aws iam get-user
   - aws iam get-role
2. (선택 사항) 현재 어떤 관리형 정책의 사용자 또는 역할이 권한 경계로 사용되는지 보려면 다음 명령을 실행하세요.

   - aws iam list-entities-for-policy
3. (선택 사항) 관리형 정책에 대한 정보를 보려면 다음 명령을 실행합니다.

   - 관리형 정책의 목록 보기: aws iam list-policies
   - 관리형 정책에 대한 세부 정보 가져오기: aws iam get-policy
4. 사용자 또는 역할에서 권한 경계를 제거하려면 다음 명령 중 하나를 사용합니다.

- aws iam delete-user-permissions-boundary
- aws iam delete-role-permissions-boundary

인라인 정책을 삭제하려면(AWS CLI)

1. (선택 사항) 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결된 모든 인라인 정책의 목록을 보려면 다음 명령 중 하나를 사용합니다.
   - aws iam list-user-policies
   - aws iam list-group-policies
   - aws iam list-role-policies
2. (선택 사항) 자격 증명(사용자, 사용자 그룹 또는 역할)에 포함된 인라인 정책 문서를 가져오려면 다음 명령 중 하나를 사용합니다.
   - aws iam get-user-policy
   - aws iam get-group-policy
   - aws iam get-role-policy
3. 자격 증명(사용자, 사용자 그룹 또는 서비스 연결 역할이 아닌 역할)에서 인라인 정책을 삭제하려면 다음 명령 중 하나를 사용합니다.
   - aws iam delete-user-policy
   - aws iam delete-group-policy
   - aws iam delete-role-policy

IAM 정책 추가(AWS API)

AWS API를 사용하여 권한을 제어하는 관리형 정책을 연결하거나 권한 경계 역할을 하는 정책을 지정할 수 있습니다. 인라인 정책을 포함할 수도 있습니다.

엔터티에 대한 권한 정책으로서 관리형 정책을 사용하려면(AWS API)

1. (선택 사항)정책에 대한 정보를 보려면 다음 작업을 호출합니다.
   - 관리형 정책의 목록 보기: ListPolicies
   - 관리형 정책에 대한 세부 정보 가져오기: GetPolicy

2. 관리형 정책을 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결하려면 다음 작업 중 하나를 호출합니다.

   - AttachUserPolicy
   - AttachGroupPolicy
   - AttachRolePolicy

보안 경계(AWS API)를 설정하기 위해서 관리형 정책을 사용하려면

1. (선택 사항) 관리형 정책에 대한 정보를 보려면 다음 작업을 호출합니다.

   - 관리형 정책의 목록 보기: ListPolicies
   - 관리형 정책에 대한 세부 정보 가져오기: GetPolicy
2. 관리형 정책을 사용하여 엔터티(사용자 또는 역할)에 대한 권한 경계를 설정하려면 다음 작업 중 하나를 호출합니다.

   - PutUserPermissionsBoundary
   - PutRolePermissionsBoundary

인라인 정책을 포함시키려면(AWS API)

인라인 정책을 자격 증명(사용자, 사용자 그룹 또는 서비스 연결 역할이 아닌 역할)에 포함시키려면 다음 작업 중 하나를 호출합니다.

- PutUserPolicy
- PutGroupPolicy
- PutRolePolicy

IAM 정책 제거(AWS API)

AWS API를 사용하여 권한을 제어하는 관리형 정책을 분리하거나 권한 경계 역할을 하는 정책을 제거할 수 있습니다. 인라인 정책을 삭제할 수도 있습니다.

권한 정책(AWS API)으로서 사용된 관리형 정책을 분리하려면

1. (선택 사항)정책에 대한 정보를 보려면 다음 작업을 호출합니다.

   - 관리형 정책의 목록 보기: ListPolicies

- 관리형 정책에 대한 세부 정보 가져오기: GetPolicy

2. (선택 사항) 정책과 자격 증명 간의 관계에 대해 확인하려면 다음 작업을 호출합니다.

   - 관리형 정책이 연결된 자격 증명(사용자, 사용자 그룹 및 역할)의 목록을 보려면 다음과 같이 합니다.
     - ListEntitiesForPolicy
   - 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결된 관리형 정책의 목록을 보려면 다음 작업 중 하나를 호출합니다.
     - ListAttachedUserPolicies
     - ListAttachedGroupPolicies
     - ListAttachedRolePolicies
3. 관리형 정책을 자격 증명(사용자, 사용자 그룹 또는 역할)에서 분리하려면 다음 작업 중 하나를 호출합니다.

   - DetachUserPolicy
   - DetachGroupPolicy
   - DetachRolePolicy

권한 경계(AWS API)를 제거하려면

1. (선택 사항) 현재 어떤 관리형 정책을 사용하여 사용자 또는 역할에 대한 권한 경계를 설정하는지 보려면 다음 작업을 호출하세요.

   - GetUser
   - GetRole
2. (선택 사항) 현재 어떤 관리형 정책의 사용자 또는 역할이 권한 경계로 사용되는지 보려면 다음 작업을 호출하세요.

   - ListEntitiesForPolicy
3. (선택 사항) 관리형 정책에 대한 정보를 보려면 다음 작업을 호출합니다.

   - 관리형 정책의 목록 보기: ListPolicies
   - 관리형 정책에 대한 세부 정보 가져오기: GetPolicy
4. 사용자 또는 역할에서 권한 경계를 제거하려면 다음 작업 중 하나를 호출합니다.

- DeleteUserPermissionsBoundary
- DeleteRolePermissionsBoundary

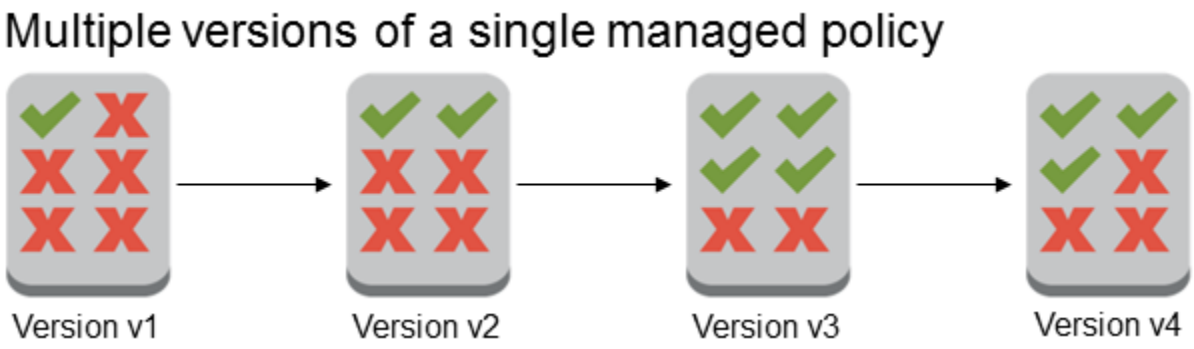
인라인 정책을 삭제하려면(AWS API)

1. (선택 사항) 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결된 모든 인라인 정책의 목록을 보려면 다음 작업 중 하나를 호출합니다.
   - ListUserPolicies
   - ListGroupPolicies
   - ListRolePolicies
2. (선택 사항) 자격 증명(사용자, 사용자 그룹 또는 역할)에 포함된 인라인 정책 문서를 가져오려면 다음 작업 중 하나를 호출합니다.
   - GetUserPolicy
   - GetGroupPolicy
   - GetRolePolicy
3. 인라인 정책을 자격 증명(사용자, 그룹 또는 서비스 연결 역할이 아닌 역할)에서 삭제하려면 다음 작업 중 하나를 호출합니다.
   - DeleteUserPolicy
   - DeleteGroupPolicy
   - DeleteRolePolicy

IAM 정책 버전 관리

IAM 고객 관리형 정책을 변경할 때, 그리고 AWS에서 AWS 관리형 정책을 변경할 때 변경된 정책은 기존 정책을 덮어쓰지 않습니다. 대신 IAM에서 관리형 정책의 새 버전을 생성합니다. IAM은 고객 관리형 정책을 최대 5개 버전까지 저장합니다. IAM은 인라인 정책에 대한 버전 관리를 지원하지 않습니다.

다음은 고객 관리형 정책의 버전 관리를 나타낸 다이어그램입니다. 이 예에서는 버전 1~4가 저장됩니다. IAM에 최대 5개의 관리형 정책 버전을 저장할 수 있습니다. 정책을 편집하여 여섯 번째 저장된 버전이 생성되는 경우 더 이상 저장하지 않을 이전 버전을 선택할 수 있습니다. 언제든지 다른 4개의 저장된 버전으로 되돌릴 수 있습니다.



정책 버전은 Version 정책 요소와 다릅니다. Version 정책 요소는 정책 내에서 사용되며 정책 언어의 버전을 정의합니다. Version 정책 요소에 대한 자세한 정보는 IAM JSON 정책 요소: Version을 참조하세요.

버전을 사용하여 관리형 정책에 대한 변경 사항을 추적할 수 있습니다. 예를 들어 관리형 정책을 변경한 다음 해당 변경 사항으로 인해 의도하지 않은 결과가 발생한 사실을 발견할 수 있습니다. 이 경우 이전 버전을 기본 버전으로 설정하여 관리형 정책의 이전 버전으로 롤백할 수 있습니다.

다음 주제에서는 관리형 정책에서 버전 관리를 사용하는 방법을 설명합니다.

주제

- 정책의 기본 버전을 설정할 수 있는 권한
- 고객 관리형 정책의 기본 버전 설정
- 버전을 사용하여 변경 사항 롤백
- 버전 제한

정책의 기본 버전을 설정할 수 있는 권한

정책의 기본 버전을 설정하는 데 필요한 권한은 작업에 대한 AWS API 작업에 해당합니다. CreatePolicyVersion 또는 SetDefaultPolicyVersion API 작업을 사용하여 정책의 기본 버전을 설정할 수 있습니다. 어떤 사람이 기존 정책의 기본 정책 버전을 설정할 수 있게 허용하려면 iam:CreatePolicyVersion 작업 또는 iam:SetDefaultPolicyVersion 작업에 대한 액세스 권한을 허용하면 됩니다. 그러면 iam:CreatePolicyVersion 작업을 이용해 새 버전의 정책을 생성하고 이 버전을 기본으로 설정할 수 있습니다. 또한 iam:SetDefaultPolicyVersion 작업을 통해서는 기존 버전의 정책을 기본으로 설정할 수 있습니다.

> **Important**
>
> 사용자의 정책에서 `iam:SetDefaultPolicyVersion` 작업을 거부해도 사용자가 새 정책 버전을 생성하고 이 버전을 기본으로 설정하는 작업을 하지 못하게 할 수는 없습니다.

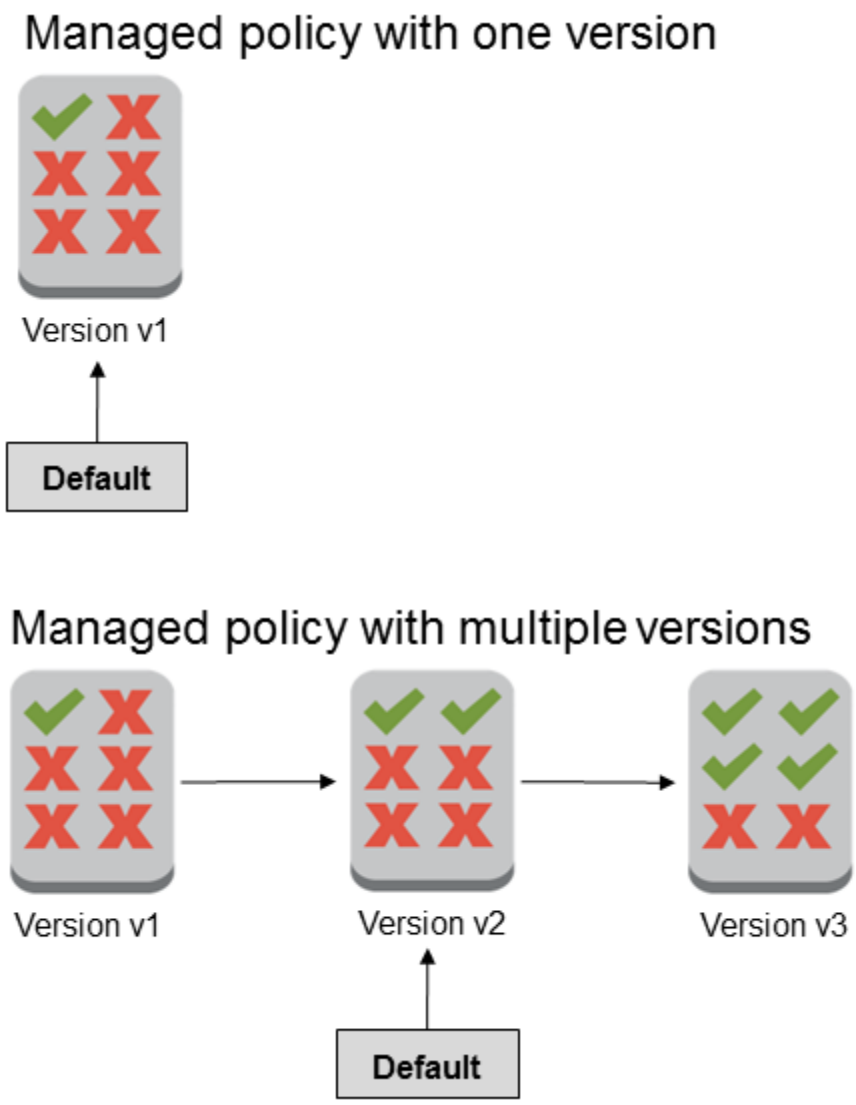
다음 정책을 사용하면 사용자가 기존 고객 관리형 정책을 변경하기 위해 액세스하는 것을 거부할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iam:CreatePolicyVersion",
                "iam:SetDefaultPolicyVersion"
            ],
            "Resource": "arn:aws:iam::*:policy/POLICY-NAME"
        }
    ]
}
```

고객 관리형 정책의 기본 버전 설정

관리형 정책의 버전 중 하나가 기본 버전으로 설정됩니다. 정책의 기본 버전은 유효한 버전입니다. 즉, 기본 버전은 관리형 정책이 연결된 모든 보안 주체 엔터티(사용자, 사용자 그룹 및 역할)에 적용되는 버전입니다.

고객 관리형 정책을 만들 때 정책은 v1로 식별되는 단일 버전으로 시작합니다. 버전이 하나뿐인 관리형 정책의 경우 해당 버전이 기본값으로 자동 설정됩니다. 버전이 둘 이상인 고객 관리형 정책의 경우에는 기본값으로 설정할 버전을 선택해야 합니다. AWS 관리형 정책의 경우 기본 버전은 AWS에서 설정됩니다. 다음 다이어그램에서는 이 개념을 보여 줍니다.



고객 관리형 정책의 기본 버전이 정책이 연결되는 모든 IAM 개체(사용자, 사용자 그룹 및 역할)에 적용되도록 해당 버전을 설정할 수 있습니다. 단, AWS 관리형 정책 또는 인라인 정책에는 기본 버전을 설정할 수 없습니다.

고객 관리형 정책의 기본 버전을 설정하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Policies(정책)을 선택합니다.
3. 정책 목록에서 기본 버전을 설정할 정책 이름을 선택합니다. 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.

4. Policy versions(정책 버전) 탭을 선택합니다. 기본 버전으로 설정할 버전 옆의 확인란을 선택한 후 기본값으로 설정을 선택합니다.

AWS Command Line Interface 또는 AWS API에서 고객 관리형 정책을 기본 버전으로 설정하는 방법을 알아보려면 고객 관리형 정책 편집(AWS CLI) 섹션을 참조하세요.

버전을 사용하여 변경 사항 롤백

변경 사항을 롤백하도록 고객 관리형 정책의 기본 버전을 설정할 수 있습니다. 예를 들어 다음 시나리오를 고려해 보세요:

사용자가 AWS Management Console을 사용하여 특정 Amazon S3 버킷을 관리할 수 있도록 허용하는 고객 관리형 정책을 생성합니다. 생성 시 고객 관리형 정책의 버전은 v1로 식별되는 한 버전뿐이어서 이 버전이 기본값으로 자동 설정됩니다. 정책이 의도대로 적용됩니다.

나중에 두 번째 Amazon S3 버킷을 관리하기 위한 권한을 추가하기 위해 정책을 업데이트합니다. IAM에서 변경 사항을 포함하고 v2로 식별되는 정책의 새 버전을 생성합니다. v2 버전을 기본값으로 설정하고 얼마 지나지 않아 사용자들이 Amazon S3 콘솔을 사용할 수 있는 권한이 없다고 보고합니다. 이 경우 의도대로 적용되는 정책의 v1 버전으로 롤백할 수 있습니다. 이렇게 하기 위해 v1 버전을 기본 버전으로 설정합니다. 이제 사용자들이 Amazon S3 콘솔을 사용하여 원래 버킷을 관리할 수 있습니다.

나중에 정책의 v2 버전에 있는 오류를 해결한 후 두 번째 Amazon S3 버킷을 관리하기 위한 권한을 추가하기 위해 다시 정책을 업데이트합니다. IAM에서 v3로 식별되는 정책의 새 버전을 하나 더 생성합니다. v3 버전을 기본값으로 설정합니다. 이 버전이 의도대로 적용됩니다. 이 시점에서 정책의 v2 버전을 삭제합니다.

버전 제한

관리형 정책에는 최대 5개의 버전이 있을 수 있습니다. 5개 버전을 만든 후에도 관리형 정책을 변경해야 할 경우 AWS Command Line Interface 또는 AWS API에서 먼저 기존 버전을 하나 이상 삭제해야 합니다. AWS Management Console을 사용할 경우에는 정책을 편집하기 전에 버전을 삭제할 필요가 없습니다. 6번째 버전을 저장할 경우 정책의 기본 버전이 아닌 버전을 한 개 이상 삭제하라는 메시지가 표시된 대화 상자가 나타납니다. 결정을 위해 각 버전의 JSON 정책 문서를 볼 수 있습니다. 이 대화 상자에 대한 자세한 내용은 the section called “IAM 정책 편집” 섹션을 참조하세요.

기본 버전을 제외하고 원하는 모든 관리형 정책 버전을 삭제할 수 있습니다. 버전을 삭제할 때 나머지 버전의 버전 식별자는 변경되지 않습니다. 따라서 버전 식별자가 순차적이지 않을 수 있습니다. 예를 들어 관리형 정책의 v2 및 v4 버전을 삭제하고 새 버전을 2개 추가하면 나머지 버전 식별자가 v1, v3, v5, v6 및 v7이 될 수 있습니다.

IAM 정책 편집

정책은 자격 증명 또는 리소스에 연결될 때 해당 권한을 정의하는 개체입니다. 정책은 AWS에 JSON 문서로 저장되고 보안 주체에 IAM의 자격 증명 기반 정책으로 연결됩니다. 자격 증명 기반 정책을 IAM 사용자 그룹, 사용자 또는 역할과 같은 보안 주체(또는 자격 증명)에 연결할 수 있습니다. 자격 증명 기반 정책에는 AWS 관리형 정책, 고객 관리형 정책 및 인라인 정책이 포함됩니다. IAM에서 고객 관리형 정책 및 인라인 정책을 편집할 수 있습니다. AWS 관리형 정책은 편집할 수 없습니다. AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 섹션을 참조하세요.

주제

정책 액세스 보기

정책에 대한 권한을 변경하기 전에 최근 서비스 수준 활동을 검토해야 합니다. 이 기능은 사용 중인 보안 주체(사람 또는 애플리케이션)의 액세스 권한을 제거하지 않으려는 경우 중요합니다. 마지막으로 액세스한 정보 보기에 대한 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 섹션을 참조하세요.

고객 관리형 정책 편집(콘솔)

고객 관리형 정책을 편집하여 정책에 정의된 권한을 변경할 수 있습니다. 고객 관리형 정책에 최대 5개의 버전을 사용할 수 있습니다. 이는 관리형 정책을 변경하여 버전이 5개 넘게 생성될 경우 AWS Management Console에서 어느 버전을 삭제할 것이지 결정하라는 메시지가 표시되므로 중요합니다. 메시지가 표시되지 않도록 편집하기 전에 기본 버전을 변경하거나 정책 버전을 삭제할 수도 있습니다. 버전에 대한 자세한 내용은 IAM 정책 버전 관리 섹션을 참조하세요.

고객 관리형 정책을 편집하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Policies(정책)을 선택합니다.

3. 정책 목록에서 편집할 정책 이름을 선택합니다. 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 권한 탭을 선택한 다음 편집을 선택합니다.
5. 다음 중 하나를 수행하세요.

   - 시각적 편집기 옵션을 선택하면 JSON 구문을 이해하지 않아도 정책을 변경할 수 있습니다. 정책의 각 권한 블록에 대한 서비스, 작업, 리소스 또는 조건(선택 사항)을 변경할 수 있습니다. 정책을 가져와 추가 권한을 정책 하단에 추가할 수도 있습니다. 변경이 완료되면 다음을 선택하여 계속 진행합니다.
   - JSON 옵션을 선택하고 JSON 텍스트 상자에 입력하거나 붙여 넣어 정책을 수정합니다. 정책을 가져와 추가 권한을 정책 하단에 추가할 수도 있습니다. 정책 검증 동안 생성된 모든 보안 경고, 오류 또는 일반 경고를 해결하고 다음을 선택합니다.

   > **Note**
   >
   > 언제든지 시각적 편집기 옵션과 JSON 편집기 옵션을 서로 전환할 수 있습니다. 그러나 변경을 적용하거나 시각적 편집기에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화되도록 정책을 재구성할 수 있습니다. 자세한 내용은 정책 재구성 섹션을 참조하세요.

6. 검토 및 저장 페이지에서 이 정책에 정의된 권한을 검토한 다음 변경 사항 저장을 선택하여 작업을 저장합니다.
7. 관리형 정책 버전이 이미 최댓값인 5개가 있을 경우 변경 사항 저장을 선택하면 대화 상자가 나타납니다. 새 버전을 저장하려면 기본이 아닌 가장 오래된 버전의 정책이 제거되고 새 버전으로 교체됩니다. 옵션으로 새로운 버전을 기본 정책 버전으로 설정할 수도 있습니다.

   변경 사항 저장을 선택하여 새 정책 버전을 저장합니다.

고객 관리형 정책의 기본 버전을 설정하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Policies(정책)을 선택합니다.
3. 정책 목록에서 기본 버전을 설정할 정책 이름을 선택합니다. 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.

4. Policy versions(정책 버전) 탭을 선택합니다. 기본 버전으로 설정할 버전 옆의 확인란을 선택한 후 기본값으로 설정을 선택합니다.

고객 관리형 정책의 버전을 삭제하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Policies(정책)을 선택합니다.
3. 버전을 삭제하려는 고객 관리형 정책의 이름을 선택합니다. 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. Policy versions(정책 버전) 탭을 선택합니다. 삭제하려는 버전 옆의 확인란을 선택합니다. 그런 다음 삭제(Delete)를 선택합니다.
5. 버전을 정말로 삭제할 것인지 다시 한 번 묻는 메시지가 나오면 확인 후 삭제를 선택합니다.

인라인 정책 편집(콘솔)

AWS Management Console에서 인라인 정책을 편집할 수 있습니다.

사용자, 사용자 그룹 또는 역할의 인라인 정책을 편집하려면(콘솔)

1. 탐색 창에서 사용자 그룹, 사용자 또는 역할을 선택합니다.
2. 정책을 변경하려는 사용자, 사용자 그룹 또는 역할 이름을 선택합니다. 그런 다음 권한 탭을 선택하고 정책을 확장합니다.
3. 인라인 정책을 편집하려면 정책 편집을 선택합니다.
4. 다음 중 하나를 수행하세요.

   - 시각적 편집기 옵션을 선택하면 JSON 구문을 이해하지 않아도 정책을 변경할 수 있습니다. 정책의 각 권한 블록에 대한 서비스, 작업, 리소스 또는 조건(선택 사항)을 변경할 수 있습니다. 정책을 가져와 추가 권한을 정책 하단에 추가할 수도 있습니다. 변경이 완료되면 다음을 선택하여 계속 진행합니다.
   - JSON 옵션을 선택하고 JSON 텍스트 상자에 입력하거나 붙여 넣어 정책을 수정합니다. 정책을 가져와 추가 권한을 정책 하단에 추가할 수도 있습니다. 정책 검증 동안 생성된 모든 보안 경고, 오류 또는 일반 경고를 해결하고 다음을 선택합니다. 현재 추가된 주체에 아무런 영향을 주지 않고 변경 사항만 저장하려면 Save as default version(기본 버전으로 저장) 확인란의 선택을 해제합니다.

> **Note**
>
> 언제든지 시각적 편집기 옵션과 JSON 편집기 옵션을 서로 전환할 수 있습니다. 그러나 변경을 적용하거나 시각적 편집기에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화되도록 정책을 재구성할 수 있습니다. 자세한 내용은 정책 재구성 섹션을 참조하세요.

5. 검토 페이지에서 정책 요약을 검토하고 나서 변경 사항 저장을 선택하여 작업을 저장합니다.

고객 관리형 정책 편집(AWS CLI)

AWS Command Line Interface에서 고객 관리형 정책을 편집할 수 있습니다(AWS CLI).

> **Note**
>
> 관리형 정책에는 최대 5개의 버전이 있을 수 있습니다. 5개 버전을 만든 후에도 고객 관리형 정책을 변경해야 할 경우 먼저 기존 버전을 1개 이상 삭제해야 합니다.

고객 관리형 정책을 편집하려면(AWS CLI)

1. (선택 사항)정책에 대한 정보를 보려면 다음 명령을 실행합니다.
   - 관리형 정책의 목록 보기: list-policies
   - 관리형 정책에 대한 세부 정보 가져오기: get-policy
2. (선택 사항) 정책과 자격 증명 간의 관계에 대해 확인하려면 다음 명령을 실행합니다.
   - 관리형 정책이 연결된 자격 증명(사용자, 사용자 그룹 및 역할)의 목록을 보려면 다음과 같이 합니다.
     - list-entities-for-policy
   - 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결된 관리형 정책의 목록을 보려면
     - list-attached-user-policies
     - list-attached-group-policies
     - list-attached-role-policies
3. 고객 관리형 정책을 편집하려면 다음 명령을 실행합니다.
   - create-policy-version

4. (선택 사항) 고객 관리형 정책을 검증하려면 다음 IAM Access Analyzer 명령을 실행합니다.

   - validate-policy

고객 관리형 정책의 기본 버전을 설정하려면(AWS CLI)

1. (선택 사항) 관리형 정책 목록을 보려면 다음 명령을 실행합니다.

   - list-policies
2. 고객 관리형 정책의 기본 버전을 설정하려면 다음 명령을 실행합니다.

   - set-default-policy-version

고객 관리형 정책의 한 버전을 삭제하려면(AWS CLI)

1. (선택 사항) 관리형 정책 목록을 보려면 다음 명령을 실행합니다.

   - list-policies
2. 고객 관리형 정책을 삭제하려면 다음 명령을 실행합니다.

   - delete-policy-version

고객 관리형 정책 편집(AWS API)

AWS API를 사용하여 고객 관리형 정책을 편집할 수 있습니다.

> **Note**
>
> 관리형 정책에는 최대 5개의 버전이 있을 수 있습니다. 5개 버전을 만든 후에도 고객 관리형 정책을 변경해야 할 경우 먼저 기존 버전을 1개 이상 삭제해야 합니다.

고객 관리형 정책을 편집하려면(AWS API)

1. (선택 사항)정책에 대한 정보를 보려면 다음 작업을 호출합니다.

   - 관리형 정책의 목록 보기: ListPolicies
   - 관리형 정책에 대한 세부 정보 가져오기: GetPolicy
2. (선택 사항) 정책과 자격 증명 간의 관계에 대해 확인하려면 다음 작업을 호출합니다.

- 관리형 정책이 연결된 자격 증명(사용자, 사용자 그룹 및 역할)의 목록을 보려면 다음과 같이 합니다.
  - ListEntitiesForPolicy
- 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결된 관리형 정책의 목록을 보려면
  - ListAttachedUserPolicies
  - ListAttachedGroupPolicies
  - ListAttachedRolePolicies

3. 고객 관리형 정책을 편집하려면 다음 작업을 호출합니다.
   - CreatePolicyVersion
4. (선택 사항) 고객 관리형 정책을 검증하려면 다음 IAM Access Analyzer 작업을 호출합니다.
   - ValidatePolicy

고객 관리형 정책의 기본 버전을 설정하려면(AWS API)

1. (선택 사항) 관리형 정책 목록을 보려면 다음 작업을 호출합니다.
   - ListPolicies
2. 고객 관리형 정책의 기본 버전을 설정하려면 다음 작업을 호출합니다.
   - SetDefaultPolicyVersion

고객 관리형 정책의 한 버전을 삭제하려면(AWS API)

1. (선택 사항) 관리형 정책 목록을 보려면 다음 작업을 호출합니다.
   - ListPolicies
2. 고객 관리형 정책을 삭제하려면 다음 작업을 호출합니다.
   - DeletePolicyVersion

IAM 정책 삭제

AWS Management Console, AWS Command Line Interface(AWS CLI) 또는 IAM API를 사용하여 IAM 정책을 삭제할 수 있습니다.

> **Note**
>
> IAM 정책 삭제는 영구적입니다. 정책을 삭제한 후에는 복구할 수 없습니다.

관리형 정책과 인라인 정책의 차이점에 대한 자세한 정보는 관리형 정책과 인라인 정책 섹션을 참조하세요.

IAM 정책에 대한 일반적인 내용은 IAM의 정책 및 권한 섹션을 참조하세요.

AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 섹션을 참조하세요.

주제

- 정책 액세스 보기
- IAM 정책 삭제(콘솔)
- IAM 정책 삭제(AWS CLI)
- IAM 정책 삭제(AWS API)

정책 액세스 보기

정책을 삭제하기 전에 최근 서비스 수준 활동을 검토해야 합니다. 이 기능은 사용 중인 보안 주체(사람 또는 애플리케이션)의 액세스 권한을 제거하지 않으려는 경우 중요합니다. 마지막으로 액세스한 정보 보기에 대한 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 섹션을 참조하세요.

IAM 정책 삭제(콘솔)

고객 관리형 정책은 삭제하여 AWS 계정에서 제거할 수 있습니다. 단, AWS 관리형 정책은 삭제할 수 없습니다.

고객 관리형 정책을 삭제하려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Policies(정책)을 선택합니다.
3. 삭제할 고객 관리형 정책 옆의 라디오 버튼을 선택합니다. 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.

4. [Actions]를 선택한 후 [Delete]를 선택합니다.
5. 지침에 따라 정책을 정말로 삭제할 것인지 확인 후 삭제를 선택합니다.

사용자 그룹, 사용자 또는 역할의 인라인 정책을 삭제하려면(콘솔)

1. 탐색 창에서 사용자 그룹, 사용자 또는 역할을 선택합니다.
2. 정책을 삭제하려는 사용자 그룹, 사용자 또는 역할 이름을 선택합니다. 그런 다음 권한 탭을 선택합니다.
3. 제거할 정책 옆에 있는 확인란을 선택하고 제거를 선택합니다. 사용자 또는 역할에서 인라인 정책을 삭제하려면 제거를 선택하여 삭제를 확인합니다. 사용자 그룹에서 단일 인라인 정책을 삭제하는 경우 정책 이름을 입력하고 삭제를 선택합니다. 사용자 그룹에서 여러 인라인 정책을 삭제하는 경우 **`inline policies`** 다음에 삭제할 정책 수를 입력한 다음 삭제를 선택합니다. 예를 들어, 세 개의 인라인 정책을 삭제하는 경우 **`3 inline policies`**를 입력합니다.

IAM 정책 삭제(AWS CLI)

AWS Command Line Interface에서 고객 관리형 정책을 삭제할 수 있습니다.

고객 관리형 정책을 삭제하려면(AWS CLI)

1. (선택 사항)정책에 대한 정보를 보려면 다음 명령을 실행합니다.
   - 관리형 정책의 목록 보기: list-policies
   - 관리형 정책에 대한 세부 정보 가져오기: get-policy
2. (선택 사항) 정책과 자격 증명 간의 관계에 대해 확인하려면 다음 명령을 실행합니다.
   - 관리형 정책이 연결된 자격 증명(사용자, 사용자 그룹 및 역할)의 목록을 보려면 다음 명령을 실행합니다.
     - list-entities-for-policy
   - 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결된 관리형 정책의 목록을 보려면 다음 명령 중 하나를 실행합니다.
     - list-attached-user-policies
     - list-attached-group-policies
     - list-attached-role-policies
3. 고객 관리형 정책을 삭제하려면 다음 명령을 실행합니다.

- delete-policy

인라인 정책을 삭제하려면(AWS CLI)

1. (선택 사항) 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결된 모든 인라인 정책의 목록을 보려면 다음 명령 중 하나를 사용합니다.
   - aws iam list-user-policies
   - aws iam list-group-policies
   - aws iam list-role-policies
2. (선택 사항) 자격 증명(사용자, 사용자 그룹 또는 역할)에 포함된 인라인 정책 문서를 가져오려면 다음 명령 중 하나를 사용합니다.
   - aws iam get-user-policy
   - aws iam get-group-policy
   - aws iam get-role-policy
3. 자격 증명(사용자, 사용자 그룹 또는 서비스 연결 역할이 아닌 역할)에서 인라인 정책을 삭제하려면 다음 명령 중 하나를 사용합니다.
   - aws iam delete-user-policy
   - aws iam delete-group-policy
   - aws iam delete-role-policy

IAM 정책 삭제(AWS API)

AWS API를 사용하여 고객 관리형 정책을 삭제할 수 있습니다.

고객 관리형 정책을 삭제하려면(AWS API)

1. (선택 사항)정책에 대한 정보를 보려면 다음 작업을 호출합니다.
   - 관리형 정책의 목록 보기: ListPolicies
   - 관리형 정책에 대한 세부 정보 가져오기: GetPolicy
2. (선택 사항) 정책과 자격 증명 간의 관계에 대해 확인하려면 다음 작업을 호출합니다.

- 관리형 정책이 연결된 자격 증명(사용자, 사용자 그룹 및 역할)의 목록을 보려면 다음 작업을 호출합니다.
  - ListEntitiesForPolicy
- 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결된 관리형 정책의 목록을 보려면 다음 작업 중 하나를 호출합니다.
  - ListAttachedUserPolicies
  - ListAttachedGroupPolicies
  - ListAttachedRolePolicies

3. 고객 관리형 정책을 삭제하려면 다음 작업을 호출합니다.

   - DeletePolicy

인라인 정책을 삭제하려면(AWS API)

1. (선택 사항) 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결된 모든 인라인 정책의 목록을 보려면 다음 작업 중 하나를 호출합니다.

   - ListUserPolicies
   - ListGroupPolicies
   - ListRolePolicies

2. (선택 사항) 자격 증명(사용자, 사용자 그룹 또는 역할)에 포함된 인라인 정책 문서를 가져오려면 다음 작업 중 하나를 호출합니다.

   - GetUserPolicy
   - GetGroupPolicy
   - GetRolePolicy

3. 인라인 정책을 자격 증명(사용자, 그룹 또는 서비스 연결 역할이 아닌 역할)에서 삭제하려면 다음 작업 중 하나를 호출합니다.

   - DeleteUserPolicy
   - DeleteGroupPolicy
   - DeleteRolePolicy

마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의

관리자는 IAM 리소스(역할, 사용자, 사용자 그룹 또는 정책)에 필요한 것 이상의 권한을 부여할 수 있습니다. IAM에서는 사용되지 않은 권한을 식별하여 제거할 수 있도록 마지막으로 액세스한 정보를 제공합니다. 마지막으로 액세스한 정보를 사용하여 정책을 재정의하고 IAM 자격 증명 및 정책에서 사용하는 서비스 및 작업에 대한 액세스만 허용할 수 있습니다. 그러면 최소 권한 모범 사례를 더 효과적으로 준수할 수 있습니다. IAM 또는 AWS Organizations에 있는 자격 증명 또는 정책에 대해 마지막으로 액세스한 정보를 볼 수 있습니다.

미사용 액세스 분석기를 사용하여 마지막으로 액세스한 정보를 지속적으로 모니터링할 수 있습니다. 자세한 내용은 외부 및 미사용 액세스에 대한 조사 결과를 참조하세요.

주제

- IAM에 대해 마지막으로 액세스한 정보 유형
- AWS Organizations에 대해 마지막으로 액세스한 정보
- 마지막으로 액세스한 정보에 대해 알아야 할 사항
- 필요한 권한
- IAM 및 Organizations 엔터티에 대한 문제 해결 활동
- AWS에서 마지막으로 액세스한 정보를 추적하는 위치
- IAM에 대해 마지막으로 액세스한 정보 보기
- Organizations에 대해 마지막으로 액세스한 정보 보기
- 마지막으로 액세스한 정보 사용에 대한 예제 시나리오
- IAM 작업에서 마지막으로 액세스한 정보와 관련된 서비스 및 작업

IAM에 대해 마지막으로 액세스한 정보 유형

IAM 자격 증명에 대해 마지막으로 액세스한 정보, 즉, 허용된 AWS 서비스 정보와 허용된 작업 정보와 같은 두 가지 유형을 볼 수 있습니다. 이 정보에는 AWS API에서 액세스를 시도한 날짜와 시간이 포함됩니다. 작업의 경우 마지막으로 액세스한 정보에서는 서비스 관리 작업을 보고합니다. 관리 작업에는 생성, 삭제 및 수정 작업이 포함됩니다. IAM에 대해 마지막으로 액세스한 정보를 보는 방법에 대한 자세한 내용은 IAM에 대해 마지막으로 액세스한 정보 보기 섹션을 참조하세요.

마지막으로 액세스한 정보를 사용하여 IAM 자격 증명에 부여할 권한을 결정하는 시나리오 예제는 마지막으로 액세스한 정보 사용에 대한 예제 시나리오 섹션을 참조하세요.

관리 작업에 대한 정보가 제공되는 방법에 대한 자세한 내용은 마지막으로 액세스한 정보에 대해 알아야 할 사항 섹션을 참조하세요.

AWS Organizations에 대해 마지막으로 액세스한 정보

관리 계정 자격 증명을 사용하여 로그인할 경우 AWS Organizations 엔터티 또는 정책에 대해 마지막으로 액세스한 서비스 정보를 볼 수 있습니다. AWS Organizations 엔터티에는 조직 루트, 조직 단위(OU) 또는 계정이 포함됩니다. AWS Organizations에 대해 마지막으로 액세스한 정보에는 SCP(서비스 제어 정책)에서 허용하는 서비스에 대한 정보가 포함됩니다. 이 정보는 조직 또는 계정에서 서비스에 마지막으로 액세스하려고 시도한 보안 주체(루트 사용자, IAM 사용자 또는 역할)와 그 시기를 나타냅니다. AWS Organizations에 대해 마지막으로 액세스한 정보를 보는 방법과 보고서에 대한 자세한 내용은 Organizations에 대해 마지막으로 액세스한 정보 보기 섹션을 참조하세요.

마지막으로 액세스한 정보를 사용하여 Organizations 엔터티에 부여할 권한을 결정하는 데 대한 예제 시나리오는 마지막으로 액세스한 정보 사용에 대한 예제 시나리오 섹션을 참조하세요.

마지막으로 액세스한 정보에 대해 알아야 할 사항

보고서에서 마지막으로 액세스한 정보를 사용하여 IAM 자격 증명 또는 조직 엔터티 조직의 권한을 변경하기 전에 먼저 정보에 대한 다음 세부 정보를 검토합니다.

- 추적 기간 - 최근 활동은 4시간 이내에 IAM 콘솔에 나타납니다. 서비스 정보의 추적 기간은 서비스가 작업 정보에 대한 추적을 시작한 시점에 따라 최소 400일입니다. Amazon S3 작업 정보에 대한 추적 기간은 2020년 4월 12일에 시작되었습니다. Amazon EC2, IAM 및 Lambda 작업의 추적 기간은 2021년 4월 7일부터 시작되었습니다. 다른 모든 서비스의 추적 기간은 2023년 5월 23일에 시작되었습니다. 작업에서 마지막으로 액세스한 정보를 사용할 수 있는 서비스 목록은 IAM 작업에서 마지막으로 액세스한 정보와 관련된 서비스 및 작업 섹션을 참조하세요. 작업에서 마지막으로 액세스한 정보를 사용할 수 있는 리전에 대한 자세한 내용은 AWS에서 마지막으로 액세스한 정보를 추적하는 위치 섹션을 참조하세요.
- 보고된 시도 횟수 - 서비스에서 마지막으로 액세스한 데이터에는 성공한 시도뿐만 아니라, 모든 AWS API 액세스 시도가 포함됩니다. 이 데이터에는 AWS Management Console, 임의의 SDK를 통한 AWS API, 또는 임의의 명령줄 도구를 사용한 모든 시도가 포함됩니다. 요청이 거부되었을 수도 있으므로, 서비스에서 마지막으로 액세스한 데이터에 예기치 않은 항목이 있다는 것이 계정이 훼손되었다는 의미는 아닙니다. 모든 API 호출에 대한 정보와 이런 호출이 성공했는지 또는 거부된 액세스인지에 대한 정보를 보여주는 신뢰할 수 있는 소스인 CloudTrail 로그를 참조하세요.
- PassRole - `iam:PassRole` 작업이 추적되지 않으며 마지막으로 액세스한 IAM 작업 정보에 포함되지 않습니다.

- 작업에서 마지막으로 액세스한 정보 - IAM 자격 증명에서 액세스한 서비스 관리 작업에 대해 작업에서 마지막으로 액세스한 정보를 사용할 수 있습니다. 작업에서 마지막으로 액세스한 보고서 정보는 서비스 목록 및 해당 작업의 목록을 참조하세요.

  > **Note**
  >
  > Amazon S3 데이터 이벤트의 경우 마지막으로 액세스한 작업 정보가 제공되지 않습니다.

- 관리 이벤트 - IAM은 CloudTrail에서 기록하는 서비스 관리 이벤트에 대한 작업 정보를 제공합니다. 경우에 따라 CloudTrail 관리 이벤트를 제어 영역 작업 또는 제어 영역 이벤트라고도 합니다. 관리 이벤트를 통해 AWS 계정의 리소스에 대해 수행한 관리 작업을 파악할 수 있습니다. CloudTrail의 관리 이벤트에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 Logging management events를 참조하세요.
- 보고서 소유자 - 보고서를 생성하는 보안 주체만 보고서 세부 정보를 볼 수 있습니다. 즉, AWS Management Console에서 정보를 볼 때 정보가 생성 및 로드될 때까지 기다려야 할 수 있습니다. AWS CLI 또는 AWS API를 사용하여 보고서 세부 정보를 가져오는 경우 자격 증명이 보고서를 생성한 보안 주체의 자격 증명과 일치해야 합니다. 역할 또는 페더레이션 사용자에 대해 임시 자격 증명을 사용하는 경우 동일한 세션 중에 보고서를 생성하고 검색해야 합니다. 위임된 역할 세션 보안 주체에 대한 자세한 내용은 AWS JSON 정책 요소: Principal 섹션을 참조하세요.
- IAM 리소스 - IAM에 대해 마지막으로 액세스한 정보에는 계정의 IAM 리소스(역할, 사용자, 사용자 그룹 및 정책)가 포함됩니다. 조직에 대해 마지막으로 액세스한 정보에는 지정된 조직 엔터티의 보안 주체(IAM 사용자, IAM 역할 또는 AWS 계정 루트 사용자)가 포함됩니다. 마지막으로 액세스한 정보에는 인증되지 않은 시도가 포함되지 않습니다.
- IAM 정책 유형 - IAM에 대해 마지막으로 액세스한 정보에는 IAM 자격 증명의 정책에서 허용하는 서비스가 포함됩니다. 이러한 정책은 역할에 연결되거나 사용자에게 직접 또는 그룹을 통해 연결됩니다. 다른 정책 유형에서 허용하는 액세스는 보고서에 포함되어 있지 않습니다. 제외된 정책 유형에는 리소스 기반 정책, 액세스 제어 목록, AWS Organizations SCP, IAM 권한 경계 및 세션 정책이 있습니다. 서비스 연결 역할에서 제공하는 권한은 연결된 서비스에 의해 정의되며 IAM에서 수정할 수 없습니다. 서비스 연결 역할에 대한 자세한 내용은 서비스 연결 역할 생성 섹션을 참조하세요. 다양한 정책 유형을 평가하여 액세스를 허용 또는 거부하는 방법을 알아보려면 정책 평가 로직 섹션을 참조하세요.
- Organizations 정책 유형 - AWS Organizations에 대한 정보에는 Organizations 엔터티의 상속된 서비스 제어 정책(SCP)이 허용하는 서비스만 포함됩니다. SCP는 루트, OU 또는 계정에 연결된 정책입니다. 다른 정책 유형에서 허용하는 액세스는 보고서에 포함되어 있지 않습니다. 제외된 정책 유형에는 자격 증명 기반 정책, 리소스 기반 정책, 액세스 제어 목록, IAM 권한 경계 및 세션 정책이 있습

니다. 다른 정책 유형이 액세스를 허용하거나 거부하는 방법을 알아보려면 정책 평가 로직 섹션을 참조하세요.

- 정책 ID 지정 - AWS CLI 또는 AWS API를 사용하여 조직에서 마지막으로 액세스한 정보에 대한 보고서를 생성할 때 선택적으로 정책 ID를 지정할 수 있습니다. 그러면 생성되는 보고서에는 해당 정책이 허용하는 서비스에 대한 정보만 포함됩니다. 이 정보에는 지정된 조직 엔터티 또는 엔터티 하위의 가장 최근 계정 활동이 포함됩니다. 자세한 내용은 aws iam generate-organizations-access-report 또는 GenerateOrganizationsAccessReport를 참조하세요.
- 조직 마스터 계정 - 마지막으로 액세스한 서비스 정보를 보려면 조직의 관리 계정에 로그인해야 합니다. IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 관리 계정에 대한 정보를 보도록 선택할 수 있습니다. 관리 계정은 SCP에 의해 제한을 받지 않으므로 생성되는 보고서에는 모든 AWS 서비스가 나열됩니다. CLI 또는 API에 정책 ID를 지정할 경우 해당 정책이 무시됩니다. 각 서비스에 대해 보고서에는 관리 계정에 대한 정보만 포함됩니다. 그러나 다른 조직 엔터티에 대한 보고서는 관리 계정의 활동에 대한 정보를 반환하지 않습니다.
- 조직 설정 - 관리자는 조직에 대한 데이터를 생성하려면 조직 루트에서 SCP를 활성화해야 합니다.

필요한 권한

AWS Management Console에서 마지막으로 액세스한 정보를 보려면 필요한 권한을 부여하는 정책이 있어야 합니다.

IAM 정보에 대한 권한

IAM 콘솔을 사용하여 IAM 사용자, 역할 또는 정책에 대해 마지막으로 액세스한 정보를 사용하려면 다음 작업을 포함하는 정책이 있어야 합니다.

- `iam:GenerateServiceLastAccessedDetails`
- `iam:Get*`
- `iam:List*`

이러한 권한을 사용하면 사용자가 다음을 확인할 수 있습니다.

- 관리형 정책에 연결된 사용자, 그룹 또는 역할
- 사용자 또는 역할이 액세스할 수 있는 서비스
- 서비스에 마지막으로 액세스한 시간
- 특정 Amazon EC2, IAM, Lambda, Amazon S3 작업을 마지막으로 사용하려고 시도한 시간

AWS CLI 또는 AWS API를 사용하여 IAM에 대해 마지막으로 액세스한 정보를 사용하려면 사용할 작업과 일치하는 권한이 있어야 합니다.

- iam:GenerateServiceLastAccessedDetails
- iam:GetServiceLastAccessedDetails
- iam:GetServiceLastAccessedDetailsWithEntities
- iam:ListPoliciesGrantingServiceAccess

이 예제는 IAM이 마지막으로 액세스한 정보를 볼 수 있도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 또한 모든 IAM에 대한 읽기 전용 액세스를 허용합니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:GenerateServiceLastAccessedDetails",
            "iam:Get*",
            "iam:List*"
        ],
        "Resource": "*"
    }
```

AWS Organizations 정보에 대한 권한

IAM 콘솔을 사용하여 조직의 루트, OU 또는 계정 엔터티에 대한 보고서를 보려면 다음 작업을 포함하는 정책이 있어야 합니다.

- iam:GenerateOrganizationsAccessReport
- iam:GetOrganizationsAccessReport
- organizations:DescribeAccount
- organizations:DescribeOrganization
- organizations:DescribeOrganizationalUnit
- organizations:DescribePolicy
- organizations:ListChildren
- organizations:ListParents

- organizations:ListPoliciesForTarget
- organizations:ListRoots
- organizations:ListTargetsForPolicy

AWS CLI 또는 AWS API를 사용하여 조직에 대해 마지막으로 액세스한 정보를 사용하려면 다음 작업을 포함하는 정책이 있어야 합니다.

- iam:GenerateOrganizationsAccessReport
- iam:GetOrganizationsAccessReport
- organizations:DescribePolicy
- organizations:ListChildren
- organizations:ListParents
- organizations:ListPoliciesForTarget
- organizations:ListRoots
- organizations:ListTargetsForPolicy

이 예제는 Organizations에 대해 마지막으로 액세스한 서비스 정보를 볼 수 있도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 또한 모든 조직에 대한 읽기 전용 액세스를 허용합니다. 이 정책은 프로그래밍 방식 및 콘솔 액세스에 대한 권한을 정의합니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:GenerateOrganizationsAccessReport",
            "iam:GetOrganizationsAccessReport",
            "organizations:Describe*",
            "organizations:List*"
        ],
        "Resource": "*"
    }
}
```

iam:OrganizationsPolicyId 조건 키를 사용하여 특정 조직 정책에 대해서만 보고서 생성을 허용할 수도 있습니다. 정책 예제는 IAM: Organizations 정책에 대해 마지막으로 액세스한 서비스 정보 보기을 참조하세요.

IAM 및 Organizations 엔터티에 대한 문제 해결 활동

경우에 따라 AWS Management Console에서 마지막으로 액세스한 정보 테이블이 비어 있을 수 있습니다. AWS CLI 또는 AWS API 요청이 빈 정보 세트 또는 null 필드를 반환할 수도 있습니다. 이러한 경우 다음 문제를 검토하세요.

- 마지막으로 액세스한 작업 정보의 경우 표시될 것으로 예상되는 작업이 목록에 반환되지 않을 수 있습니다. IAM 자격 증명이 작업에 대한 권한을 보유하지 않거나 마지막으로 액세스한 정보에 대한 작업을 AWS에서 아직 추적하지 않았기 때문에 이러한 문제가 발생할 수 있습니다.
- IAM 사용자의 경우 사용자가 직접 또는 그룹 멤버십을 통해 인라인 또는 관리형 정책이 하나 이상 연결되어 있는지 확인합니다.
- IAM 그룹의 경우 그룹에 인라인 또는 관리형 정책이 하나 이상 연결되어 있는지 확인합니다.
- IAM 그룹의 경우 보고서는 그룹 정책을 사용하여 서비스에 액세스한 멤버에 대해서만 마지막으로 액세스한 서비스 정보를 반환합니다. 멤버가 다른 정책을 사용했는지 여부를 확인하려면 해당 사용자에 대해 마지막으로 액세스한 정보를 검토하세요.
- IAM 역할의 경우 역할에 인라인 또는 관리형 정책이 하나 이상 연결되어 있는지 확인합니다.
- IAM 엔터티(사용자 또는 역할)의 경우 해당 엔터티의 사용 권한에 영향을 줄 수 있는 다른 정책 유형을 검토합니다. 여기에는 리소스 기반 정책, 액세스 제어 목록, AWS Organizations 정책, IAM 권한 경계 또는 세션 정책이 있습니다. 자세한 내용은 정책 유형 또는 단일 계정 내에서 정책 평가을 참조하세요.
- IAM 정책의 경우 지정된 관리형 정책이 하나 이상의 사용자, 멤버가 있는 그룹 또는 역할에 연결되어 있는지 확인합니다.
- 조직 엔터티(루트, OU 또는 계정)의 경우 조직 관리 계정 자격 증명을 사용하여 로그인되어 있는지 확인합니다.
- 조직 루트에서 SCP가 활성화되어 있는지 확인합니다.
- 마지막으로 액세스한 작업 정보는 IAM 작업에서 마지막으로 액세스한 정보와 관련된 서비스 및 작업에 설치된 작업에만 사용할 수 있습니다.

변경을 수행할 때 IAM 콘솔 보고서에 작업이 나타날 때까지 최소 4시간을 기다립니다. AWS CLI 또는 AWS API를 사용하는 경우 업데이트된 정보를 표시하려면 새 보고서를 생성해야 합니다.

AWS에서 마지막으로 액세스한 정보를 추적하는 위치

AWS는 표준 AWS 리전에 대해 마지막으로 액세스한 정보를 수집합니다. AWS에서 리전을 추가하면 AWS에서 각 리전의 정보 추적을 시작한 날짜와 함께 해당 리전이 다음 테이블에 추가됩니다.

- 서비스 정보 - 서비스 추적 기간은 최소 400일이지만, 해당 리전에서 지난 400일 이내에 이 기능 추적을 시작한 경우 400일보다 더 짧습니다.
- 작업 정보 - Amazon S3 관리 작업에 대한 추적 기간은 2020년 4월 12일에 시작되었습니다. Amazon EC2, IAM 및 Lambda 관리 작업에 대한 추적 기간은 2021년 4월 7일에 시작되었습니다. 다른 모든 서비스의 관리 작업에 대한 추적 기간은 2023년 5월 23일에 시작되었습니다. 리전의 추적 날짜가 2023년 5월 23일 이후인 경우 작업에서 마지막으로 액세스한 해당 리전의 정보는 이후 날짜로 시작됩니다.

지역명	지역	추적 시작 날짜
미국 동부(오하이오)	us-east-2	2017년 10월 27일
미국 동부(버지니아 북부)	us-east-1	2015년 10월 1일
미국 서부(캘리포니아 북부)	us-west-1	2015년 10월 1일
미국 서부(오레곤)	us-west-2	2015년 10월 1일
아프리카(케이프타운)	af-south-1	2020년 4월 22일
아시아 태평양(홍콩)	ap-east-1	2019년 4월 24일
아시아 태평양(하이데라바드)	ap-south-2	2022년 11월 22일
아시아 태평양(자카르타)	ap-southeast-3	2021년 12월 13일
아시아 태평양(멜버른)	ap-southeast-4	2023년 1월 23일
아시아 태평양(뭄바이)	ap-south-1	2016년 6월 27일
아시아 태평양(오사카)	ap-northeast-3	2018년 2월 11일
아시아 태평양(서울)	ap-northeast-2	2016년 1월 6일
아시아 태평양(싱가포르)	ap-southeast-1	2015년 10월 1일
아시아 태평양(시드니)	ap-southeast-2	2015년 10월 1일
아시아 태평양(도쿄)	ap-northeast-1	2015년 10월 1일

지역명	지역	추적 시작 날짜
캐나다(중부)	ca-central-1	2017년 10월 28일
유럽(프랑크푸르트)	eu-central-1	2015년 10월 1일
유럽(아일랜드)	eu-west-1	2015년 10월 1일
유럽(런던)	eu-west-2	2017년 10월 28일
유럽(밀라노)	eu-south-1	2020년 4월 28일
유럽(파리)	eu-west-3	2017년 12월 18일
유럽(스페인)	eu-south-2	2022년 11월 15일
유럽(스톡홀름)	eu-north-1	2018년 12월 12일
유럽(취리히)	eu-central-2	2022년 11월 8일
이스라엘(텔아비브)	il-central-1	2023년 8월 1일
중동(바레인)	me-south-1	2019년 7월 29일
중동(UAE)	me-central-1	2022년 8월 30일
남아메리카(상파울루)	sa-east-1	2015년 12월 11일
AWS GovCloud(미국 동부)	us-gov-east-1	2023년 7월 1일
AWS GovCloud(미국 서부)	us-gov-west-1	2023년 7월 1일

앞의 테이블에 나와 있지 않은 리전은 마지막으로 액세스한 정보를 아직 제공하지 않습니다.

AWS 리전은 AWS 리소스를 지리적 영역에 모아 놓은 것입니다. 리전은 파티션으로 그룹화됩니다. 표준 리전은 `aws` 파티션에 속하는 리전입니다. 여러 파티션에 대한 자세한 내용은 AWS 일반 참조의 Amazon 리소스 이름(ARN) 형식을 참조하십시오. 리전에 대한 자세한 내용은 AWS 일반 참조의 AWS 지역 정보를 참조하십시오.

IAM에 대해 마지막으로 액세스한 정보 보기

AWS Management Console, AWS CLI 또는 AWS API를 사용하여 IAM에 대해 마지막으로 액세스한 정보를 볼 수 있습니다. 마지막으로 액세스한 정보가 표시되는 서비스 및 해당 작업의 목록을 참조하세요. 마지막으로 액세스한 정보에 대한 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 섹션을 참조하세요.

IAM에서 다음 리소스 유형에 대한 정보를 볼 수 있습니다. 각각의 경우 정보에는 지정된 보고 기간 동안 허용된 서비스가 포함됩니다.

- 사용자 - 사용자가 허용된 각 서비스에 액세스하려고 시도한 마지막 시간을 표시합니다.
- 사용자 그룹 - 사용자 그룹 멤버가 허용된 각 서비스에 액세스하려고 시도한 마지막 시간에 대한 정보를 표시합니다. 또한 이 보고서에는 액세스를 시도한 총 멤버 수가 포함됩니다.
- 역할 - 해당 역할이 허용된 각 서비스에 액세스하려고 시도한 마지막 시간을 표시합니다.
- 정책 - 사용자 또는 역할이 허용된 각 서비스에 액세스하려고 시도한 마지막 시간에 대한 정보를 표시합니다. 또한 이 보고서에는 액세스를 시도한 총 엔터티 수가 포함됩니다.

> **Note**
>
> IAM의 리소스에 대한 액세스 정보를 보려면 먼저 보고 기간, 보고된 엔터티 및 정보에 대해 평가된 정책 유형을 이해해야 합니다. 자세한 내용은 the section called “마지막으로 액세스한 정보에 대해 알아야 할 사항” 섹션을 참조하세요.

IAM에 대한 정보 보기(콘솔)

IAM 콘솔의 액세스 관리자(Access Advisor) 탭에서 IAM에 대해 마지막으로 액세스한 정보를 볼 수 있습니다.

IAM에 대한 정보를 보려면(콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자 그룹, 사용자, 역할 또는 정책을 선택합니다.
3. 사용자, 사용자 그룹, 역할 또는 정책 이름을 선택하여 요약 페이지를 열고 액세스 관리자(Access Advisor) 탭을 선택합니다. 선택한 리소스를 기반으로 다음 정보를 확인합니다.

- 사용자 그룹 - 사용자 그룹 멤버가 액세스할 수 있는 서비스 목록을 봅니다. 또한 멤버가 서비스에 마지막으로 액세스한 시기, 사용한 사용자 그룹 정책 및 요청한 사용자 그룹 멤버를 볼 수 있습니다. 정책의 이름을 선택하여 정책이 관리형 정책인지 아니면 인라인 사용자 그룹 정책인지 확인합니다. 사용자 그룹 멤버의 이름을 선택하여 사용자 그룹의 모든 멤버를 확인하고 마지막으로 서비스에 액세스한 시간을 확인합니다.
- 사용자 - 사용자가 액세스할 수 있는 서비스 목록을 봅니다. 또한 서비스에 마지막으로 액세스한 시기 및 사용한 정책을 볼 수도 있습니다. 정책의 이름을 선택하여 정책이 관리형 정책인지, 인라인 사용자 정책인지, 아니면 사용자 그룹의 인라인 정책인지 확인합니다.
- 역할 - 역할이 액세스할 수 있는 서비스 목록, 서비스에 마지막으로 액세스한 역할 및 사용된 정책 목록을 표시합니다. 정책의 이름을 선택하여 정책이 관리형 정책인지 아니면 인라인 역할 정책인지 확인합니다.
- 정책 - 정책에서 허용된 작업이 있는 서비스 목록을 봅니다. 서비스에 액세스하는 데 정책이 마지막으로 사용된 시기와 해당 정책을 사용한 엔터티(사용자 또는 역할)도 볼 수 있습니다. 그 마지막 액세스 날짜에는 다른 정책을 통해 이 정책에 대한 액세스 권한이 부여된 시기도 포함됩니다. 엔터티의 이름을 선택하여 어떤 엔터티에 이 정책이 연결되어 있는지 그리고 마지막으로 서비스에 액세스한 시간을 확인합니다.

4. 테이블의 서비스 열을 선택하고 마지막으로 액세스한 작업 정보가 포함된 서비스 중 하나의 이름을 선택하여 IAM 엔터티가 액세스하려고 시도한 관리 작업 목록을 확인합니다. 사용자가 마지막으로 작업을 수행하려고 시도한 시점을 보여주는 AWS 리전 및 타임스탬프를 볼 수 있습니다.
5. 마지막으로 액세스한 작업 정보가 포함된 서비스의 서비스 및 관리 작업에 대해 마지막 액세스 열이 표시됩니다. 이 열에 반환되는 다음과 같은 가능한 결과를 검토합니다. 이러한 결과는 서비스나 작업이 허용되는지, 액세스되는지, 마지막으로 액세스한 정보에 대해 AWS가 추적하는지 여부에 따라 달라집니다.

   <number of>일 전

   추적 기간에 서비스 또는 작업이 사용된 이후의 일 수입니다. 서비스에 대한 추적 기간은 지난 400일입니다. Amazon S3 작업에 대한 추적 기간은 2020년 4월 12일에 시작되었습니다. Amazon EC2, IAM 및 Lambda 작업에 대한 추적 기간은 2021년 4월 7일에 시작되었습니다. 다른 모든 서비스의 추적 기간은 2023년 5월 23일에 시작되었습니다. 각 AWS 리전의 추적 시작 날짜에 대한 자세한 내용은 AWS에서 마지막으로 액세스한 정보를 추적하는 위치 섹션을 참조하세요.

   추적 기간에 액세스하지 않음

   추적된 서비스 또는 작업은 추적 기간 동안 엔터티에서 사용되지 않았습니다.

목록에 나타나지 않는 작업에 대한 권한이 있을 수 있습니다. 작업에 대한 추적 정보가 현재 AWS에 포함되지 않는 경우 이 문제가 발생할 수 있습니다. 추적 정보가 없는 경우 권한을 결정해서는 안 됩니다. 대신 이 정보를 사용하여 최소 권한 부여에 대한 전반적인 전략을 알리고 지원하는 것이 좋습니다. 정책을 확인하여 액세스 수준이 적절한지 확인합니다.

IAM에 대한 정보 보기(AWS CLI)

AWS CLI를 사용하여 AWS 서비스 및 Amazon S3, Amazon EC2, IAM 및 Lambda 작업에 액세스하기 위해 IAM 리소스가 사용된 마지막 시간에 대한 정보를 검색할 수 있습니다. IAM 리소스는 사용자, 사용자 그룹, 역할 또는 정책입니다.

IAM에 대한 정보를 보려면(AWS CLI)

1. 보고서를 생성합니다. 요청에는 보고서가 필요한 IAM 리소스(사용자, 사용자 그룹, 역할 또는 정책)의 ARN이 포함되어야 합니다. 보고서에 생성할 세부 수준을 지정하여 서비스 또는 서비스 및 작업 모두에 대한 액세스 세부 정보를 볼 수 있습니다. 이 요청은 작업이 완료될 때까지 `get-service-last-accessed-details` 및 `get-service-last-accessed-details-with-entities` 작업에서 `job-status`를 모니터링하기 위해 사용할 수 있는 `job-id`를 반환합니다.

   - aws iam generate-service-last-accessed-details
2. 이전 단계의 `job-id` 파라미터를 사용하여 보고서에 대한 세부 정보를 검색합니다.

   - aws iam get-service-last-accessed-details

   이 작업은 `generate-service-last-accessed-details` 작업에서 요청한 리소스 유형 및 세부 수준에 따라 다음 정보를 반환합니다.

   - 사용자 - 지정한 사용자가 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 사용자의 마지막 시도 날짜 및 시간과 사용자의 ARN을 반환합니다.
   - 사용자 그룹 - 사용자 그룹에 연결된 정책을 사용하여 지정된 사용자 그룹의 멤버가 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 사용자 그룹 멤버가 마지막으로 시도한 날짜와 시간을 반환합니다. 또한 해당 사용자의 ARN과 서비스에 액세스하려고 시도한 사용자 그룹 멤버의 총 수를 반환합니다. 모든 멤버 목록을 반환하려면 GetServiceLastAccessedDetailsWithEntities 작업을 사용합니다.
   - 역할 - 지정한 역할이 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 역할의 마지막 시도 날짜 및 시간과 역할의 ARN을 반환합니다.

- 정책 - 지정된 정책으로 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 엔터티(사용자 또는 역할)가 정책을 사용하여 마지막으로 서비스에 액세스하려고 시도한 날짜와 시간을 반환합니다. 또한 엔터티의 ARN과 액세스를 시도한 엔터티의 총 수를 반환합니다.

3. 특정 서비스에 액세스하기 위해 사용자 그룹 또는 정책 권한을 사용하는 엔터티에 대해 자세히 알아봅니다. 이 작업은 각 엔터티의 ARN, ID, 이름, 경로, 유형(사용자 또는 역할) 및 마지막으로 서비스에 액세스하려고 시도한 엔터티의 목록을 반환합니다. 사용자와 역할에 대해 이 작업을 사용할 수도 있지만 해당 엔터티에 대한 정보만 반환합니다.

   - aws iam get-service-last-accessed-details-with-entities

4. 특정 서비스에 액세스하기 위해 자격 증명(사용자, 사용자 그룹 또는 역할)이 사용하는 자격 기반 정책에 대해 자세히 알아봅니다. 자격 증명 및 서비스를 지정한 경우 이 작업은 해당 자격 증명이 지정된 서비스에 액세스하는 데 사용할 수 있는 권한 정책 목록을 반환합니다. 이 작업은 정책의 현재 상태를 제공하며 생성된 보고서에 의존하지 않습니다. 또한 리소스 기반 정책, 액세스 제어 목록, AWS Organizations 정책, IAM 권한 경계 또는 세션 정책 등의 다른 정책 유형을 반환하지 않습니다. 자세한 내용은 정책 유형 또는 단일 계정 내에서 정책 평가 섹션을 참조하세요.

   - aws iam list-policies-granting-service-access

IAM에 대한 정보 보기(AWS API)

AWS를 사용하여 AWS API 서비스 및 Amazon S3, Amazon EC2, IAM 및 Lambda 작업에 액세스하기 위해 IAM 리소스가 사용된 마지막 시간에 대한 정보를 검색할 수 있습니다. IAM 리소스는 사용자, 사용자 그룹, 역할 또는 정책입니다. 보고서에 생성할 세부 수준을 지정하여 서비스 또는 서비스 및 작업 모두에 대한 세부 정보를 볼 수 있습니다.

IAM에 대한 정보를 보려면(AWS API)

1. 보고서를 생성합니다. 요청에는 보고서가 필요한 IAM 리소스(사용자, 사용자 그룹, 역할 또는 정책)의 ARN이 포함되어야 합니다. 작업이 완료될 때까지 `GetServiceLastAccessedDetails` 및 `GetServiceLastAccessedDetailsWithEntities` 작업에서 `JobStatus`를 모니터링하기 위해 사용할 수 있는 `JobId`를 반환합니다.

   - GenerateServiceLastAccessedDetails

2. 이전 단계의 `JobId` 파라미터를 사용하여 보고서에 대한 세부 정보를 검색합니다.

   - GetServiceLastAccessedDetails

이 작업은 `GenerateServiceLastAccessedDetails` 작업에서 요청한 리소스 유형 및 세부 수준에 따라 다음 정보를 반환합니다.

- 사용자 - 지정한 사용자가 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 사용자의 마지막 시도 날짜 및 시간과 사용자의 ARN을 반환합니다.
- 사용자 그룹 - 사용자 그룹에 연결된 정책을 사용하여 지정된 사용자 그룹의 멤버가 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 사용자 그룹 멤버가 마지막으로 시도한 날짜와 시간을 반환합니다. 또한 해당 사용자의 ARN과 서비스에 액세스하려고 시도한 사용자 그룹 멤버의 총 수를 반환합니다. 모든 멤버 목록을 반환하려면 GetServiceLastAccessedDetailsWithEntities 작업을 사용합니다.
- 역할 - 지정한 역할이 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 역할의 마지막 시도 날짜 및 시간과 역할의 ARN을 반환합니다.
- 정책 - 지정된 정책으로 액세스할 수 있는 서비스 목록을 반환합니다. 각 서비스에 대해 작업은 엔터티(사용자 또는 역할)가 정책을 사용하여 마지막으로 서비스에 액세스하려고 시도한 날짜와 시간을 반환합니다. 또한 엔터티의 ARN과 액세스를 시도한 엔터티의 총 수를 반환합니다.

3. 특정 서비스에 액세스하기 위해 사용자 그룹 또는 정책 권한을 사용하는 엔터티에 대해 자세히 알아봅니다. 이 작업은 각 엔터티의 ARN, ID, 이름, 경로, 유형(사용자 또는 역할) 및 마지막으로 서비스에 액세스하려고 시도한 엔터티의 목록을 반환합니다. 사용자와 역할에 대해 이 작업을 사용할 수도 있지만 해당 엔터티에 대한 정보만 반환합니다.

   - GetServiceLastAccessedDetailsWithEntities

4. 특정 서비스에 액세스하기 위해 자격 증명(사용자, 사용자 그룹 또는 역할)이 사용하는 자격 기반 정책에 대해 자세히 알아봅니다. 자격 증명 및 서비스를 지정한 경우 이 작업은 해당 자격 증명이 지정된 서비스에 액세스하는 데 사용할 수 있는 권한 정책 목록을 반환합니다. 이 작업은 정책의 현재 상태를 제공하며 생성된 보고서에 의존하지 않습니다. 또한 리소스 기반 정책, 액세스 제어 목록, AWS Organizations 정책, IAM 권한 경계 또는 세션 정책 등의 다른 정책 유형을 반환하지 않습니다. 자세한 내용은 정책 유형 또는 단일 계정 내에서 정책 평가 섹션을 참조하세요.

   - ListPoliciesGrantingServiceAccess

Organizations에 대해 마지막으로 액세스한 정보 보기

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 AWS Organizations에 대해 마지막으로 액세스한 서비스 정보를 볼 수 있습니다. 데이터, 필요 권한, 문제 해결, 지원되는 리전에 대한 중요 정보는 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 섹션을 참조하세요.

AWS Organizations 관리 계정 자격 증명을 사용하여 IAM 콘솔에 로그인하는 경우 조직의 모든 엔터티에 대한 정보를 볼 수 있습니다. Organizations 엔터티에는 조직 루트, 조직 단위(OU) 및 계정이 포함됩니다. 또한 IAM 콘솔을 사용하여 조직의 모든 서비스 제어 정책(SCP)에 대한 정보를 볼 수 있습니다. IAM에는 엔터티에 적용되는 SCP가 허용하는 서비스의 목록이 표시됩니다. 각 서비스에 대해 선택한 조직 엔터티 또는 엔터티 하위에 대한 가장 최근의 계정 활동 정보를 볼 수 있습니다.

관리 계정 자격 증명으로 AWS CLI 또는 AWS API를 사용하면 조직의 모든 엔터티 또는 정책에 대한 보고서를 생성할 수 있습니다. 엔터티에 대한 프로그래밍 방식 보고서는 SCP가 엔터티에 적용되도록 허용하는 서비스의 목록을 포함합니다. 각 서비스에 대해, 보고서는 지정된 조직 엔터티 또는 엔터티 하위 트리의 계정에 대한 가장 최근의 활동을 포함합니다.

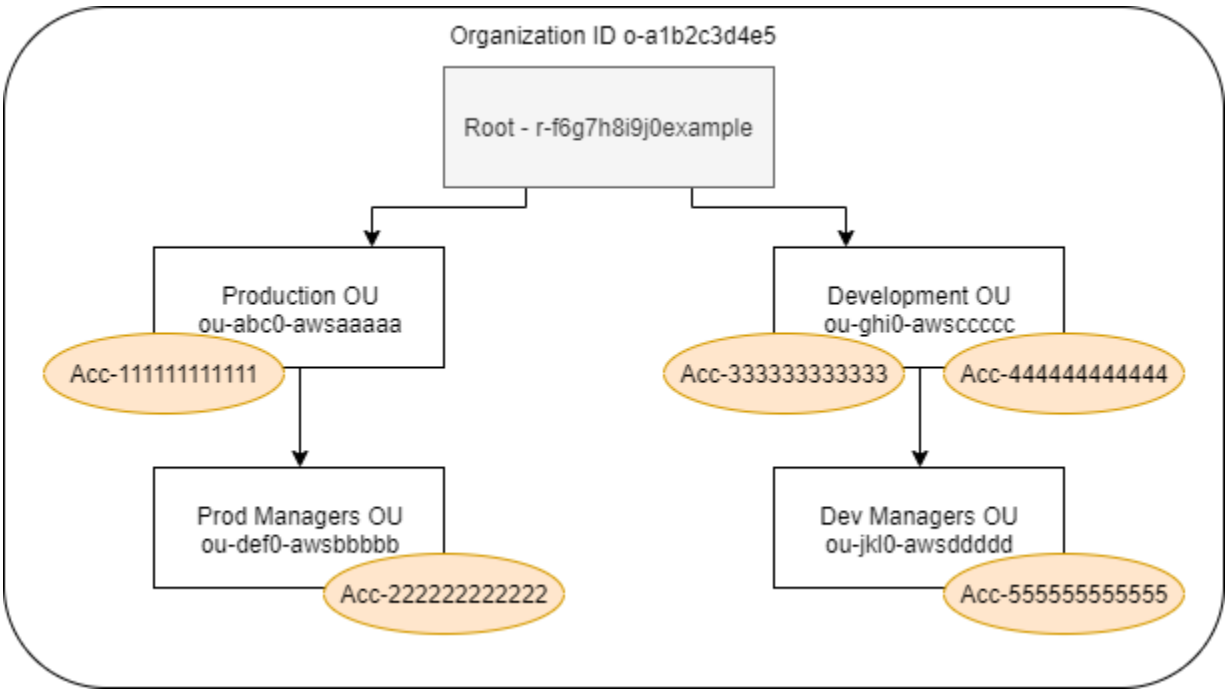
정책에 대한 프로그래밍 방식 보고서를 생성할 때 조직 엔터티를 지정해야 합니다. 이 보고서는 지정된 SCP가 허용하는 서비스의 목록을 포함합니다. 각 서비스에 대해, 해당 정책이 권한을 부여한 엔터티 또는 엔터티 하위의 가장 최근 계정 활동이 포함됩니다. 자세한 내용은 aws iam generate-organizations-access-report 또는 GenerateOrganizationsAccessReport를 참조하세요.

보고서를 보려면 관리 계정 요건 및 정보, 보고 주기, 보고된 엔터티 및 평가된 정책 유형을 이해하고 있어야 합니다. 자세한 내용은 the section called “마지막으로 액세스한 정보에 대해 알아야 할 사항” 섹션을 참조하세요.

AWS Organizations 엔터티 경로 이해

AWS CLI 또는 AWS API를 사용하여 AWS Organizations 액세스 보고서를 생성하는 경우 엔터티 경로를 지정해야 합니다. 경로는 조직 엔터티 구조의 텍스트 표현입니다.

조직의 알려진 구조를 사용하여 엔터티 경로를 작성할 수 있습니다. 예를 들어 AWS Organizations에 다음과 같은 조직 구조가 있다고 가정합니다.



Dev Managers(개발자 관리자) OU의 경로는 조직의 ID, 루트 및 경로에 있는 모든 OU(해당 OU 포함)를 사용하여 작성됩니다.

```
o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-ghi0-awsccccc/ou-jkl0-awsdddddd/
```

프로덕션 OU의 계정 경로는 조직의 ID, 루트, OU 및 계정 번호를 사용하여 작성됩니다.

```
o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-abc0-awsaaaaa/111111111111/
```

> **Note**
>
> 조직 ID는 전역적으로 고유하지만 OU ID와 루트 ID는 조직 내에서만 고유합니다. 즉, 두 조직이 동일한 조직 ID를 공유하지 않습니다. 그러나 다른 조직에는 사용자 ID와 동일한 OU 또는 루트가 있을 수 있습니다. OU 또는 루트를 지정할 때는 항상 조직 ID를 포함하는 것이 좋습니다.

Organizations에 대한 정보 보기(콘솔)

IAM 콘솔을 사용하여 루트, OU, 계정 또는 정책에 대해 마지막으로 액세스한 서비스 정보를 볼 수 있습니다.

루트에 대한 정보를 보려면(콘솔)

1. Organizations 관리 계정 자격 증명을 사용해 AWS Management Console에 로그인한 다음 https://console.aws.amazon.com/iam에서 IAM 콘솔을 엽니다.
2. Access reports(보고서 액세스) 섹션 아래의 탐색 창에서 Organization activity(조직 활동)를 선택합니다.
3. 조직 활동 페이지에서 루트를 선택합니다.
4. Details and activity(세부 정보 및 활동) 탭에서 Service access report(서비스 액세스 보고서) 섹션을 봅니다. 정보에는 루트에 직접 연결된 정책이 허용하는 서비스의 목록이 포함됩니다. 이 정보는 서비스에 마지막으로 액세스한 계정 및 그 시간을 보여줍니다. 서비스에 액세스한 보안 주체에 대한 세부 정보를 보려면 해당 계정의 관리자로 로그인하고 IAM에서 마지막으로 액세스한 서비스 정보를 봅니다.
5. 연결된 SCP(Attached SCPs) 탭을 선택하여 루트에 연결된 서비스 제어 정책(SCP)의 목록을 봅니다. IAM으로 각 정책이 연결된 대상 엔터티의 수를 볼 수 있습니다. 이 정보를 사용하여 어느 SCP를 검토할지 결정할 수 있습니다.
6. SCP의 이름을 선택하여 해당 정책이 허용하는 모든 서비스를 봅니다. 각 서비스에 대해, 서비스에 마지막으로 액세스한 계정 및 그 시간을 봅니다.
7. AWS Organizations에서 편집을 선택하여 추가 세부 정보를 보고 조직 콘솔에서 SCP를 편집합니다. 자세한 내용은 AWS Organizations 사용 설명서의 SCP 업데이트를 참조하세요.

OU 또는 계정에 대한 정보를 보려면(콘솔)

1. Organizations 관리 계정 자격 증명을 사용해 AWS Management Console에 로그인한 다음 https://console.aws.amazon.com/iam에서 IAM 콘솔을 엽니다.
2. Access reports(보고서 액세스) 섹션 아래의 탐색 창에서 Organization activity(조직 활동)를 선택합니다.
3. 조직 활동 페이지에서 조직의 구조를 확장합니다. 그런 다음 OU의 이름 또는 관리 계정을 제외하고 보려는 임의의 계정의 이름을 선택합니다.
4. Details and activity(세부 정보 및 활동) 탭에서 Service access report(서비스 액세스 보고서) 섹션을 봅니다. 정보에는 OU 또는 계정 및 모든 해당 상위 항목에 연결된 SCP가 허용하는 서비스의 목록이 포함됩니다. 이 정보는 서비스에 마지막으로 액세스한 계정 및 그 시간을 보여줍니다. 서비스에 액세스한 보안 주체에 대한 세부 정보를 보려면 해당 계정의 관리자로 로그인하고 IAM에서 마지막으로 액세스한 서비스 정보를 봅니다.

5. 연결된 SCP(Attached SCPs) 탭을 선택하여 OU 또는 계정에 직접 연결된 서비스 제어 정책(SCP)의 목록을 봅니다. IAM으로 각 정책이 연결된 대상 엔터티의 수를 볼 수 있습니다. 이 정보를 사용하여 어느 SCP를 검토할지 결정할 수 있습니다.
6. SCP의 이름을 선택하여 해당 정책이 허용하는 모든 서비스를 봅니다. 각 서비스에 대해, 서비스에 마지막으로 액세스한 계정 및 그 시간을 봅니다.
7. AWS Organizations에서 편집을 선택하여 추가 세부 정보를 보고 조직 콘솔에서 SCP를 편집합니다. 자세한 내용은 AWS Organizations 사용 설명서의 SCP 업데이트를 참조하세요.

관리 계정에 대한 정보를 보려면(콘솔)

1. Organizations 관리 계정 자격 증명을 사용해 AWS Management Console에 로그인한 다음 https://console.aws.amazon.com/iam에서 IAM 콘솔을 엽니다.
2. Access reports(보고서 액세스) 섹션 아래의 탐색 창에서 Organization activity(조직 활동)를 선택합니다.
3. 조직 활동 페이지에서 조직의 구조를 확장하고 관리 계정의 이름을 선택합니다.
4. Details and activity(세부 정보 및 활동) 탭에서 Service access report(서비스 액세스 보고서) 섹션을 봅니다. 정보에는 모든 AWS 서비스 목록이 포함됩니다. 관리 계정은 SCP에 의해 제한되지 않습니다. 이 정보는 계정이 서비스에 액세스했는지 여부와 마지막으로 액세스한 시간을 보여줍니다. 서비스에 액세스한 보안 주체에 대한 세부 정보를 보려면 해당 계정의 관리자로 로그인하고 IAM에서 마지막으로 액세스한 서비스 정보를 봅니다.
5. 연결된 SCP(Attached SCPs) 탭을 선택하여 연결된 SCP가 없음을 확인합니다(해당 계정이 관리 계정이기 때문).

정책에 대한 정보를 보려면(콘솔)

1. Organizations 관리 계정 자격 증명을 사용해 AWS Management Console에 로그인한 다음 https://console.aws.amazon.com/iam에서 IAM 콘솔을 엽니다.
2. Access reports(보고서 액세스) 섹션 아래의 탐색 창에서 Service control policies (SCPs)(서비스 제어 정책(SCP))를 선택합니다.
3. 서비스 제어 정책(SCP) 페이지에서 조직의 정책 목록을 봅니다. 각 정책이 연결된 대상 엔터티의 수를 볼 수 있습니다.
4. SCP의 이름을 선택하여 해당 정책이 허용하는 모든 서비스를 봅니다. 각 서비스에 대해, 서비스에 마지막으로 액세스한 계정 및 그 시간을 봅니다.

5. AWS Organizations에서 편집을 선택하여 추가 세부 정보를 보고 조직 콘솔에서 SCP를 편집합니다. 자세한 내용은 AWS Organizations 사용 설명서의 SCP 업데이트를 참조하세요.

Organizations에 대한 정보 보기(AWS CLI)

AWS CLI를 사용하여 조직 루트, OU, 계정 또는 정책에 대해 마지막으로 액세스한 서비스 정보를 검색할 수 있습니다.

조직에서 마지막으로 액세스한 서비스 정보를 보려면(AWS CLI)

1. 필요한 IAM 및 조직 권한이 있는 조직 관리 계정 자격 증명을 사용하여 루트에 대해 SCP가 활성화되어 있는지 확인합니다. 자세한 내용은 마지막으로 액세스한 정보에 대해 알아야 할 사항 섹션을 참조하세요.
2. 보고서를 생성합니다. 요청은 보고서를 작성할 조직 엔터티(루트, OU 또는 계정)의 경로를 포함해야 합니다. 선택적으로 `organization-policy-id` 파라미터를 포함하여 특정 정책에 대한 보고서를 볼 수 있습니다. 이 명령은 작업이 완료될 때까지 `get-organizations-access-report` 명령에 사용하여 `job-status`를 모니터링하기 위해 사용할 수 있는 `job-id`를 반환합니다.

   - aws iam generate-organizations-access-report
3. 이전 단계의 `job-id` 파라미터를 사용하여 보고서에 대한 세부 정보를 검색합니다.

   - aws iam get-organizations-access-report

   이 명령은 엔터티 멤버가 액세스할 수 있는 서비스의 목록을 반환합니다. 각 서비스에 대해, 계정 멤버에 의한 마지막 시도의 날짜 및 시간과 계정의 엔터티 경로가 반환됩니다. 또한 액세스 가능한 서비스의 총 개수와 액세스되지 않은 서비스의 수도 반환됩니다. 선택적으로 `organizations-policy-id` 파라미터를 지정한 경우 액세스 가능한 서비스는 지정된 정책이 허용하는 서비스입니다.

Organizations에 대한 정보 보기(AWS API)

AWS API를 사용하여 조직 루트, OU, 계정 또는 정책에 대해 마지막으로 액세스한 서비스 정보를 검색할 수 있습니다.

조직에서 마지막으로 액세스한 서비스 정보를 보려면(AWS API)

1. 필요한 IAM 및 조직 권한이 있는 조직 관리 계정 자격 증명을 사용하여 루트에 대해 SCP가 활성화되어 있는지 확인합니다. 자세한 내용은 마지막으로 액세스한 정보에 대해 알아야 할 사항 섹션을 참조하세요.
2. 보고서를 생성합니다. 요청은 보고서를 작성할 조직 엔터티(루트, OU 또는 계정)의 경로를 포함해야 합니다. 선택적으로 `OrganizationsPolicyId` 파라미터를 포함하여 특정 정책에 대한 보고서를 볼 수 있습니다. 이 작업은 작업이 완료될 때까지 `GetOrganizationsAccessReport` 작업에서 `JobStatus`를 모니터링하기 위해 사용할 수 있는 `JobId`를 반환합니다.

   - GenerateOrganizationsAccessReport
3. 이전 단계의 `JobId` 파라미터를 사용하여 보고서에 대한 세부 정보를 검색합니다.

   - GetOrganizationsAccessReport

   이 작업은 엔터티 멤버가 액세스할 수 있는 서비스의 목록을 반환합니다. 각 서비스에 대해, 계정 멤버에 의한 마지막 시도의 날짜 및 시간과 계정의 엔터티 경로가 반환됩니다. 또한 액세스 가능한 서비스의 총 개수와 액세스되지 않은 서비스의 수도 반환됩니다. 선택적으로 `OrganizationsPolicyId` 파라미터를 지정한 경우 액세스 가능한 서비스는 지정된 정책이 허용하는 서비스입니다.

마지막으로 액세스한 정보 사용에 대한 예제 시나리오

마지막으로 액세스한 정보를 사용하여 IAM 엔터티 또는 AWS Organizations 엔터티에 부여할 권한을 결정할 수 있습니다. 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 단원을 참조하십시오.

> **Note**
>
> IAM 또는 AWS Organizations의 엔터티 또는 정책에 대한 액세스 정보를 보려면 먼저 보고 기간, 보고된 엔터티 및 데이터에 대해 평가된 정책 유형을 이해해야 합니다. 자세한 내용은 the section called “마지막으로 액세스한 정보에 대해 알아야 할 사항”를 참조하세요.

회사에 적합한 액세스 가능성과 최소 권한 간에 적절한 균형을 유지하는 것은 관리자에게 달려 있습니다.

정보를 사용하여 IAM 그룹의 권한 축소

마지막으로 액세스한 정보를 사용하여 사용자에게 필요한 서비스만 포함하도록 IAM 그룹 권한을 축소할 수 있습니다. 이 방법은 서비스 수준에서 최소 권한을 부여하는 데 있어 중요한 단계입니다.

예를 들어, Paulo Santos는 Example Corp에 대한 AWS 사용자 권한 정의를 담당하는 관리자입니다. 이 회사는 이제 막 AWS 사용을 시작했으며 소프트웨어 개발 팀은 아직 어떤 AWS 서비스를 사용할지 정의하지 않았습니다. Paulo는 팀에게 필요한 서비스에만 액세스할 수 있는 권한을 부여하려고 하지만, 아직 정의되지 않았기 때문에 일시적으로 파워 사용자 권한을 부여합니다. 그런 다음 마지막으로 액세스한 정보를 사용하여 그룹 권한을 축소합니다.

Paulo는 다음 JSON 텍스트를 사용하여 `ExampleDevelopment` 관리 정책을 만듭니다. 그런 다음 이 정책을 `Development` 그룹에 연결하고 모든 개발자를 그룹에 추가합니다.

> **Note**
>
> Paulo의 고급 사용자가 일부 서비스 및 기능을 사용하려면 `iam:CreateServiceLinkedRole` 권한이 필요할 수 있습니다. 이 권한을 추가하면 사용자가 서비스 연결 역할을 생성할 수 있게 됩니다. Paulo는 파워 사용자에 대한 이 위험을 수락합니다.

```
{

    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessToAllServicesExceptPeopleManagement",
            "Effect": "Allow",
            "NotAction": [
                "iam:*",
                "organizations:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "RequiredIamAndOrgsActions",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole",
                "iam:ListRoles",
```

```
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        }
    ]
}
```

Paulo는 90일 후에 AWS Management Console을 사용하여 `Development` 그룹에 대해 서비스에서 마지막으로 액세스한 정보를 보기로 결정합니다. 그는 그룹 멤버가 액세스한 서비스 목록을 확인합니다. 그는 사용자가 지난주에 5개의 서비스, 즉 AWS CloudTrail, Amazon CloudWatch Logs, Amazon EC2, AWS KMS 및 Amazon S3에 액세스했다는 사실을 알게 되었습니다. 그들은 AWS를 처음 평가할 때 몇 가지 다른 서비스에 액세스했지만 그 이후에는 액세스하지 않았습니다.

Paulo는 5가지 서비스와 필요한 IAM 및 조직 작업만 포함하도록 정책 권한을 줄이기로 합니다. 그는 다음 JSON 텍스트를 사용하여 `ExampleDevelopment` 정책을 편집합니다.

> **Note**
>
> Paulo의 고급 사용자가 일부 서비스 및 기능을 사용하려면 `iam:CreateServiceLinkedRole` 권한이 필요할 수 있습니다. 이 권한을 추가하면 사용자가 서비스 연결 역할을 생성할 수 있게 됩니다. Paulo는 파워 사용자에 대한 이 위험을 수락합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessToListedServices",
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "kms:*",
                "cloudtrail:*",
                "logs:*",
                "ec2:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "RequiredIamAndOrgsActions",
```

```
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole",
                "iam:ListRoles",
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        }
    ]
}
```

권한을 더 줄이기 위해 Paulo는 AWS CloudTrail 이벤트 기록에서 계정의 이벤트를 확인할 수 있습니다. 여기서 그는 개발자가 필요로 하는 작업과 리소스만 포함하도록 정책 권한을 줄이기 위해 사용할 수 있는 자세한 이벤트 정보를 볼 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 콘솔에서 CloudTrail 이벤트 보기를 참조하세요.

정보를 사용하여 IAM 사용자의 사용 권한 축소

마지막으로 액세스한 정보를 사용하여 개별 IAM 사용자의 권한을 축소할 수 있습니다.

예를 들어 Martha Rivera는 회사 직원들의 AWS 권한이 초과되지 않도록 관리하는 IT 관리자입니다. 정기 보안 검사의 일환으로 Martha는 모든 IAM 사용자의 권한을 검토합니다. 이러한 사용자 가운데는 애플리케이션 개발자로, 이전에 보안 엔지니어의 역할을 담당했던 Nikhil Jayashankar 씨도 있습니다. 작업 요구 사항의 변화로 인해 Nikhil은 `app-dev` 그룹과 `security-team` 그룹의 멤버입니다. 새로운 직무와 관련된 `app-dev` 그룹은 Amazon EC2, Amazon EBS, Auto Scaling, Amazon S3, Route 53 및 Elastic Transcoder를 포함한 여러 서비스에 대한 권한을 부여합니다. 이전 직무와 관련된 `security-team` 그룹은 IAM 및 CloudTrail에 대한 권한을 부여합니다.

관리자인 Martha는 IAM 콘솔에 로그인하고 사용자(Users)를 선택한 다음 `nikhilj` 이름을 선택하고 액세스 관리자(Access Advisor) 탭을 선택합니다.

Martha는 마지막 액세스 열을 검토하여 Nikhil이 최근에 IAM, CloudTrail, Route 53, Amazon Elastic Transcoder 및 기타 여러 AWS 서비스에 액세스하지 않았다는 것을 확인합니다. Nikhil은 Amazon S3 액세스했습니다. Martha는 서비스 목록에서 S3를 선택하여 Nikhil이 지난 2주 동안 일부 S3 `List` 작업을 수행했음을 알게 됩니다. 회사 내에서 Martha는 Nikhil이 더 이상 내부 보안 팀의 멤버가 아니므로 업무적으로 IAM 및 CloudTrail에 액세스할 필요가 없다는 것을 확인합니다.

Martha는 이제 마지막으로 액세스한 서비스 및 작업 정보에 대해 조치를 취할 준비가 되었습니다. 그러나 이전 예제의 그룹과 달리 `nikhilj`과 같은 IAM 사용자는 여러 정책을 준수하고 여러 그룹의 멤버가 될 수 있습니다. Martha는 `nikhilj` 또는 다른 그룹 멤버의 액세스를 실수로 방해하지 않도록 주의해

서 진행해야 합니다. Nikhil에게 부여할 액세스 권한의 종류뿐만 아니라, 이러한 권한을 받는 방법을 결정해야 합니다.

Martha는 권한 탭을 선택합니다. 이 탭에서 nikhilj에 직접 연결된 정책 및 그룹을 통해 연결된 정책을 확인합니다. 그녀는 각 정책을 확장하고 Nikhil이 사용하지 않는 서비스에 대한 액세스를 허용하는 정책을 알아보기 위해 정책 요약을 확인합니다.

- IAM - IAMFullAccess AWS 관리형 정책은 nikhilj에 직접 연결되고 security-team 그룹에 연결됩니다.
- CloudTrail - AWSCloudTrailReadOnlyAccess AWS 관리형 정책은 security-team 그룹에 연결됩니다.
- Route 53 - App-Dev-Route53 고객 관리형 정책은 app-dev 그룹에 연결됩니다.
- Elastic Transcoder - App-Dev-ElasticTranscoder 고객 관리형 정책은 app-dev 그룹에 연결됩니다.

Martha는 nikhilj에 직접 연결된 IAMFullAccess AWS 관리형 정책을 제거하기로 결정했습니다. 또한 security-team 그룹에 대한 Nikhil의 멤버십을 제거합니다. 이 두 작업은 IAM 및 CloudTrail에 대한 불필요한 액세스를 제거합니다.

Route 53 및 Elastic Transcoder에 액세스할 수 있는 Nikhil의 권한은 app-dev 그룹에 의해 부여됩니다. Nikhil은 이러한 서비스를 사용하지 않지만 그룹의 다른 멤버에게는 필요할 수 있습니다. Martha는 app-dev 그룹에 대해 마지막으로 액세스한 정보를 검토하고 여러 멤버가 최근에 Route 53 및 Amazon S3에 액세스했음을 알게 됩니다. 그러나 작년에는 Elastic Transcoder에 액세스한 멤버가 없습니다. 그녀는 그룹에서 App-Dev-ElasticTranscoder 고객 관리형 정책을 제거합니다.

그런 다음 Martha는 고객 관리형 정책인 App-Dev-ElasticTranscoder에 대해 마지막으로 액세스한 정보를 검토합니다. 그녀는 정책이 다른 IAM 자격 증명에 연결되지 않았다는 것을 알게 됩니다. 그녀는 회사 내에서 앞으로 해당 정책이 필요하지 않다는 것을 조사한 다음 삭제합니다.

IAM 리소스를 삭제하기 전에 정보 사용

IAM 리소스를 삭제하기 전에 마지막으로 액세스한 정보를 사용하여 마지막으로 리소스를 사용한 이후로 일정 시간이 경과했는지 확인할 수 있습니다. 이는 사용자, 그룹, 역할 및 정책에 적용됩니다. 이러한 작업에 대한 자세한 내용은 다음 주제를 참조하세요.

- 사용자 - 사용자 삭제
- 그룹 - 그룹 삭제
- 역할 - 역할 삭제

- 정책 - 관리형 정책 삭제(이로 인해 자격 증명에서 정책이 분리됨)

IAM 정책을 편집하기 전에 정보 사용

해당 리소스에 영향을 미치는 정책을 편집하기 전에 IAM 자격 증명(사용자, 그룹 또는 역할) 또는 IAM 정책에 대해 마지막으로 액세스한 정보를 검토할 수 있습니다. 이 기능은 사용 중인 사용자의 액세스 권한을 제거하지 않으려는 경우 중요합니다.

예를 들어, Arnav Desai는 Example Corp의 개발자이자 AWS 관리자입니다. 그의 팀이 AWS를 사용하기 시작하여 모든 개발자에게 IAM 및 Organizations를 제외한 모든 서비스에 대한 완전한 액세스를 허용하는 고급 사용자 권한을 부여했습니다. Arnav는 최소 권한 부여를 위한 첫 걸음으로 AWS CLI를 사용하여 자신의 계정에서 관리형 정책을 검토하려고 합니다.

이렇게 하기 위해 Arnav는 먼저 다음 명령을 사용하여 자격 증명에 연결된 계정에 고객 관리형 권한 정책을 나열합니다.

```
aws iam list-policies --scope Local --only-attached --policy-usage-filter
 PermissionsPolicy
```

응답에서 그는 각 정책에 대한 ARN을 캡처합니다. 그런 다음 Arnav는 다음 명령을 사용하여 각 정책에 대해 마지막으로 액세스한 정보 보고서를 생성합니다.

```
aws iam generate-service-last-accessed-details --arn arn:aws:iam::123456789012:policy/
ExamplePolicy1
```

이 응답에서 그는 `JobId` 필드에서 생성된 보고서의 ID를 캡처합니다. 그런 다음 Arnav는 `JobStatus` 필드가 `COMPLETED` 또는 `FAILED` 값을 반환할 때까지 다음 명령을 폴링합니다. 작업이 실패할 경우 오류를 캡처합니다.

```
aws iam get-service-last-accessed-details --job-id 98a765b4-3cde-2101-2345-example678f9
```

작업의 상태가 `COMPLETED`이면 Arnav는 JSON 형식의 `ServicesLastAccessed` 배열에 대한 콘텐츠를 구문 분석합니다.

```
 "ServicesLastAccessed": [
       {
           "TotalAuthenticatedEntities": 1,
           "LastAuthenticated": 2018-11-01T21:24:33.222Z,
           "ServiceNamespace": "dynamodb",
```

```
            "LastAuthenticatedEntity": "arn:aws:iam::123456789012:user/IAMExampleUser",
            "ServiceName": "Amazon DynamoDB"
        },

        {
            "TotalAuthenticatedEntities": 0,
            "ServiceNamespace": "ec2",
            "ServiceName": "Amazon EC2"
        },

        {
            "TotalAuthenticatedEntities": 3,
            "LastAuthenticated": 2018-08-25T15:29:51.156Z,
            "ServiceNamespace": "s3",
            "LastAuthenticatedEntity": "arn:aws:iam::123456789012:role/IAMExampleRole",
            "ServiceName": "Amazon S3"
        }
    ]
```

이 정보를 통해 Arnav는 `ExamplePolicy1` 정책이 Amazon DynamoDB, Amazon S3 및 Amazon EC2의 세 가지 서비스에 대한 액세스를 허용한다는 것을 알게 됩니다. 11월 1일 IAM 사용자 `IAMExampleUser`가 DynamoDB에 마지막으로 액세스하려고 시도했으며, 8월 25일에는 어떤 사용자가 Amazon S3에 액세스하기 위해 `IAMExampleRole` 역할을 사용했습니다. 작년에 Amazon S3에 액세스하려고 시도한 엔터티가 두 개 더 있습니다. 그러나 작년에 Amazon EC2에 액세스하려고 시도한 사용자는 아무도 없었습니다.

이는 Arnav가 정책에서 Amazon EC2 작업을 안전하게 제거할 수 있음을 의미합니다. Arnav는 정책에 대한 현재 JSON 문서를 검토하려고 합니다. 먼저, 다음 명령을 사용하여 정책의 버전 번호를 결정해야 합니다.

```
aws iam list-policy-versions --policy-arn arn:aws:iam::123456789012:policy/
ExamplePolicy1
```

응답에서 Arnav는 `Versions` 배열에서 현재 기본 버전 번호를 수집합니다. 그런 다음, 다음 명령을 통해 해당 버전 번호(v2)를 사용하여 JSON 정책 문서를 요청합니다.

```
aws iam get-policy-version --policy-arn arn:aws:iam::123456789012:policy/ExamplePolicy1
 --version-id v2
```

Arnav는 반환된 JSON 정책 문서를 `PolicyVersion` 배열의 `Document` 필드에 저장합니다. 정책 문서에서 Arnav는 `ec2` 네임스페이스에서 작업을 검색합니다. 정책에 남아 있는 다른 네임스페이스의 작

업이 없는 경우 영향을 받는 자격 증명(사용자, 그룹 및 역할)에서 정책을 분리합니다. 그런 다음 정책을 삭제합니다. 이 경우, 정책에는 Amazon DynamoDB 및 Amazon S3 서비스가 포함됩니다. 그래서 Arnav는 문서에서 Amazon EC2 작업을 제거하고 변경 사항을 저장합니다. 그는 다음 명령을 사용하여 새 버전의 문서를 통해 정책을 업데이트하고 해당 버전을 기본 정책 버전으로 설정합니다.

```
aws iam create-policy-version --policy-arn arn:aws:iam::123456789012:policy/
ExamplePolicy1 --policy-document file://UpdatedPolicy.json --set-as-default
```

이제 `ExamplePolicy1` 정책이 업데이트되어 불필요한 Amazon EC2 서비스에 대한 액세스를 제거합니다.

기타 IAM 시나리오

IAM 리소스(사용자, 그룹, 역할 또는 정책)가 서비스에 마지막으로 액세스하려고 시도한 시점에 대한 정보는 다음 작업 중 하나를 완료할 때 도움이 될 수 있습니다.

- 정책 - 기존 고객 관리형 또는 인라인 정책을 편집하여 권한 제거
- 정책 - 인라인 정책을 관리형 정책으로 변환한 다음 삭제
- 정책 - 기존 정책에 명시적 거부 추가
- 정책 - 자격 증명(사용자, 그룹 또는 역할)에서 관리형 정책 분리
- 엔터티 - 엔터티(사용자 또는 역할)가 가질 수 있는 최대 권한을 제어하도록 권한 경계 설정
- 그룹 - 그룹에서 사용자 제거

정보를 사용하여 조직 단위의 권한 재정의

마지막으로 액세스한 정보를 사용하여 AWS Organizations의 조직 단위(OU)의 권한을 재정의할 수 있습니다.

예를 들어 John Stiles는 AWS Organizations 관리자입니다. 그는 회사 AWS 계정의 사람들에게 과도한 권한이 부여되지 않도록 할 책임이 있습니다. 정기 보안 검사의 일환으로 그는 자신의 조직에 부여된 권한을 검토합니다. 그의 `Development` OU에는 새로운 AWS 서비스를 테스트하는 데 자주 사용되는 계정이 포함되어 있습니다. John은 180일 이상 액세스되지 않은 서비스에 대한 보고서를 주기적으로 검토하기로 결정합니다. 그런 다음 OU 멤버가 이러한 서비스에 액세스할 수 있는 권한을 제거합니다.

John은 자신의 관리 계정 자격 증명을 사용하여 IAM 콘솔에 로그인합니다. IAM 콘솔에서 `Development` OU에 대한 조직 데이터를 찾습니다. 그는 서비스 액세스 보고서 표를 검토하여 180일 이상 액세스되지 않은 2개의 AWS 서비스를 발견합니다. 그는 개발 팀이 Amazon Lex 및 AWS

Database Migration Service에 액세스할 수 있는 권한을 추가한 것을 기억합니다. 그래서 개발 팀에 연락하여 더 이상 이러한 서비스를 테스트할 업무상 필요가 없는지 확인합니다.

John은 이제 마지막으로 액세스한 정보에 대해 조치를 취할 준비가 되었습니다. 그는 AWS Organizations에서 편집을 선택합니다. 그러면 이 SCP가 여러 엔터티에 연결되어 있다는 메시지가 표시됩니다. 계속을 선택합니다. 그는 AWS Organizations에서 대상을 검토하여 SCP에 어떤 조직 엔터티가 연결되어 있는지 확인합니다. 모든 엔터티가 `Development` OU에 속해 있습니다.

John은 `NewServiceTest` SCP에서 Amazon Lex 및 AWS Database Migration Service 작업에 대한 액세스를 거부하기로 합니다. 이 작업은 서비스에 대한 불필요한 액세스 권한을 제거합니다.

IAM 작업에서 마지막으로 액세스한 정보와 관련된 서비스 및 작업

다음 테이블에는 IAM 작업에서 마지막으로 액세스한 정보와 관련된 AWS 서비스가 표시됩니다. 각 서비스의 작업 목록은 서비스 승인 참조의 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

서비스	서비스 접두사
AWS Identity and Access Management 액세스 분석기	access-analyzer
AWS Account Management	account
AWS Certificate Manager	acm
Amazon Managed Workflows for Apache Airflow	airflow
AWS Amplify	amplify
AWS Amplify UI 빌더	amplifyuibuilder
Amazon AppIntegrations	app-integrations
AWS AppConfig	appconfig
Amazon AppFlow	appflow
AWS Application Cost Profiler	application-cost-profiler
Amazon CloudWatch Application Insights	applicationinsights

서비스	서비스 접두사
AWS App Mesh	appmesh
Amazon AppStream 2.0	appstream
AWS AppSync	appsync
Amazon Managed Service for Prometheus	aps
Amazon Athena	athena
AWS Audit Manager	auditmanager
AWS Auto Scaling	Auto Scaling
AWS Marketplace	aws-marketplace
AWS Backup	백업
AWS Batch	일괄
Amazon Braket	braket
AWS Budgets	예산
AWS Cloud9	cloud9
AWS CloudFormation	cloudformation
Amazon CloudFront	cloudfront
AWS CloudHSM	cloudhsm
Amazon CloudSearch	cloudsearch
AWS CloudTrail	cloudtrail
Amazon CloudWatch	cloudwatch
AWS CodeArtifact	codeartifact

서비스	서비스 접두사
AWS CodeDeploy	codedeploy
Amazon CodeGuru Profiler	codeguru-profiler
Amazon CodeGuru Reviewer	codeguru-reviewer
AWS CodePipeline	CodePipeline
AWS CodeStar	codestar
AWS CodeStar 알림	codestar-notifications
Amazon Cognito 자격 증명	cognito-identity
Amazon Cognito 사용자 풀	cognito-idp
Amazon Cognito Sync	cognito-sync
Amazon Comprehend Medical	comprehendmedical
AWS Compute Optimizer	compute-optimizer
AWS Config	config
Amazon Connect	연결
AWS Cost and Usage Report	cur
AWS Glue DataBrew	databrew
AWS Data Exchange	dataexchange
AWS Data Pipeline	datapipeline
DynamoDB Accelerator	dax
AWS Device Farm	devicefarm

서비스	서비스 접두사
Amazon DevOps Guru	devops-guru
AWS Direct Connect	directconnect
Amazon Data Lifecycle Manager	dlm
AWS Database Migration Service	DMS
Amazon DocumentDB Elastic Clusters	docdb-elastic
AWS Directory Service	ds
Amazon DynamoDB	dynamodb
Amazon Elastic Block Store	ebs
Amazon Elastic Compute Cloud	EC2
Amazon Elastic 컨테이너 레지스트리	ecr
Amazon Elastic Container Registry Public	ecr-public
Amazon Elastic Container Service	ecs
Amazon Elastic Kubernetes Service	eks
Amazon Elastic Inference	elastic-inference
Amazon ElastiCache	elasticache
AWS Elastic Beanstalk	elasticbeanstalk
Amazon Elastic File System	elasticfilesystem
Elastic Load Balancing	elasticloadbalancing
Amazon Elastic Transcoder	elastictranscoder
Amazon EMR on EKS(EMR 컨테이너)	emr-containers

서비스	서비스 접두사
Amazon EMR Serverless	emr-serverless
Amazon OpenSearch Service	es
Amazon EventBridge	이벤트
Amazon CloudWatch Evidently	evidently
Amazon FinSpace	finspace
Amazon Data Firehose	firehose
AWS Fault Injection Service	fis
AWS Firewall Manager	fms
Amazon Fraud Detector	frauddetector
Amazon FSx	fsx
Amazon GameLift	gamelift
Amazon Location Service	geo
Amazon S3 Glacier	glacier
Amazon Managed Grafana	grafana
AWS IoT Greengrass	greengrass
AWS Ground Station	groundstation
Amazon GuardDuty	guardduty
AWS HealthLake	healthlake
Amazon Honeycode	honeycode
AWS Identity and Access Management	iam

서비스	서비스 접두사
AWS 자격 증명 스토어	identitystore
EC2 Image Builder	imagebuilder
Amazon Inspector Classic	inspector
Amazon Inspector	inspector2
AWS IoT	iot
AWS IoT Analytics	iotanalytics
AWS IoT Core Device Advisor	iotdeviceadvisor
AWS IoT Events	iotevents
AWS IoT Fleet Hub	iotfleethub
AWS IoT SiteWise	iotsitewise
AWS IoT TwinMaker	iottwinmaker
AWS IoT Wireless	iotwireless
Amazon Interactive Video Service	ivs
Amazon Interactive Video Service Chat	ivschat
Amazon Managed Streaming for Apache Kafka	kafka
Amazon Managed Streaming for Kafka Connect	kafkaconnect
Amazon Kendra	kendra
Amazon Kinesis	kinesis
Amazon Kinesis Analytics V2	kinesisanalytics
AWS Key Management Service	kms

서비스	서비스 접두사
AWS Lambda	lambda
Amazon Lex	lex
AWS License Manager Linux Subscriptions Manager	license-manager-linux-subscriptions
Amazon Lightsail	lightsail
Amazon CloudWatch Logs	로그
Amazon Lookout for Equipment	lookoutequipment
Amazon Lookout for Metrics	lookoutmetrics
Amazon Lookout for Vision	lookoutvision
AWS Mainframe Modernization	m2
Amazon Managed Blockchain	managedblockchain
AWS Elemental MediaConnect	mediaconnect
AWS Elemental MediaConvert	mediaconvert
AWS Elemental MediaLive	medialive
AWS Elemental MediaStore	mediastore
AWS Elemental MediaTailor	mediatailor
Amazon MemoryDB	memorydb
AWS Application Migration Service	mgn
AWS Migration Hub	mgh
AWS Migration Hub Strategy Recommendations	migrationhub-strategy

서비스	서비스 접두사
Amazon Pinpoint	mobiletargeting
Amazon MQ	mq
AWS Network Manager	networkmanager
Amazon Nimble Studio	nimble
AWS HealthOmics	omics
AWS OpsWorks	opsworks
AWS OpsWorks CM	opsworks-cm
AWS Outposts	outposts
AWS Organizations	organizations
AWS Panorama	panorama
AWS 성능 개선 도우미	pi
Amazon EventBridge 파이프	pipes
Amazon Polly	polly
Amazon Connect Customer Profiles	profile
Amazon QLDB	qldb
AWS Resource Access Manager	ram
AWS 휴지통	rbin
Amazon Relational Database Service	RDS
Amazon Redshift	redshift
Amazon Redshift 데이터 API	redshift-data

서비스	서비스 접두사
AWS Migration Hub Refactor Spaces	refactor-spaces
Amazon Rekognition	rekognition
AWS Resilience Hub	resiliencehub
AWS 리소스 탐색기	resource-explorer-2
AWS Resource Groups	resource-groups
AWS RoboMaker	robomaker
AWS Identity and Access Management Roles Anywhere	rolesanywhere
Amazon Route 53	route53
Amazon Route 53 Recovery Controls	route53-recovery-control-config
Amazon Route 53 Recovery Readiness	route53-recovery-readiness
Amazon Route 53 Resolver	route53resolver
AWS CloudWatch RUM	rum
Amazon Simple Storage Service(S3)	s3
Amazon S3 on Outposts	s3-outposts
Amazon SageMaker 지리 공간 기능	sagemaker-geospatial
Savings Plans	savingsplans
Amazon EventBridge 스키마	schemas
Amazon SimpleDB	sdb

서비스	서비스 접두사
AWS Secrets Manager	secretsmanager
AWS Security Hub	securityhub
Amazon Security Lake	securitylake
AWS Serverless Application Repository	serverlessrepo
AWS Service Catalog	servicecatalog
AWS Cloud Map	servicediscovery
Service Quotas	servicequotas
Amazon Simple Email Service	ses
AWS Shield	shield
AWS Signer	signer
AWS SimSpace Weaver	simspaceweaver
AWS Server Migration Service	sms
Amazon Pinpoint SMS 및 음성 서비스	sms-voice
AWS Snowball	snowball
Amazon Simple Queue Service	sqs
AWS Systems Manager	ssm
AWS Systems Manager Incident Manager	ssm-incidents
AWS Systems Manager for SAP	ssm-sap
AWS Step Functions	states
AWS Security Token Service	sts

서비스	서비스 접두사
Amazon Simple Workflow Service	swf
Amazon CloudWatch Synthetics	synthetics
AWS Resource Groups Tagging API	tag
Amazon Textract	textract
Amazon Timestream	timestream
AWS 통신 네트워크 빌더	tnb
Amazon Transcribe	transcribe
AWS Transfer Family	전송
Amazon Translate	translate
Amazon Connect Voice ID	voiceid
Amazon VPC Lattice	vpc-lattice
AWS WAFV2	wafv2
AWS Well-Architected Tool	wellarchitected
Amazon Connect Wisdom	wisdom
Amazon WorkLink	worklink
Amazon WorkSpaces	WorkSpaces
AWS X-Ray	xray

작업에서 마지막으로 액세스한 정보와 관련된 작업

다음 테이블에는 작업에서 마지막으로 액세스한 정보를 사용할 수 있는 작업이 표시됩니다.

서비스 접두사	작업
access-analyzer	access-analyzer:ApplyArchiveRule
	access-analyzer:CancelPolicyGeneration
	access-analyzer:CheckAccessNotGranted
	access-analyzer:CheckNoNewAccess
	access-analyzer:CheckNoPublicAccess
	access-analyzer:CreateAccessPreview
	access-analyzer:CreateAnalyzer
	access-analyzer:CreateArchiveRule
	access-analyzer:DeleteAnalyzer
	access-analyzer:DeleteArchiveRule
	access-analyzer:GenerateFindingRecommendation
	access-analyzer:GetAccessPreview
	access-analyzer:GetAnalyzedResource
	access-analyzer:GetAnalyzer
	access-analyzer:GetArchiveRule
	access-analyzer:GetFinding
	access-analyzer:GetFindingRecommendation
	access-analyzer:GetGeneratedPolicy
	access-analyzer:ListAccessPreviewFindings
	access-analyzer:ListAccessPreviews
	access-analyzer:ListAnalyzedResources

서비스 접두사	작업
	access-analyzer:ListAnalyzers access-analyzer:ListArchiveRules access-analyzer:ListFindings access-analyzer:ListPolicyGenerations access-analyzer:StartPolicyGeneration access-analyzer:StartResourceScan access-analyzer:UpdateArchiveRule access-analyzer:UpdateFindings access-analyzer:ValidatePolicy
account	account:AcceptPrimaryEmailUpdate account:DeleteAlternateContact account:DisableRegion account:EnableRegion account:GetAlternateContact account:GetContactInformation account:GetPrimaryEmail account:GetRegionOptStatus account:ListRegions account:PutAlternateContact account:PutContactInformation account:StartPrimaryEmailUpdate

서비스 접두사	작업
acm	acm:DeleteCertificate acm:DescribeCertificate acm:ExportCertificate acm:GetAccountConfiguration acm:GetCertificate acm:ImportCertificate acm:ListCertificates acm:PutAccountConfiguration acm:RenewCertificate acm:RequestCertificate acm:ResendValidationEmail acm:UpdateCertificateOptions
airflow	airflow:CreateCliToken airflow:CreateEnvironment airflow:CreateWebLoginToken airflow:DeleteEnvironment airflow:GetEnvironment airflow:ListEnvironments airflow:PublishMetrics airflow:UpdateEnvironment

서비스 접두사	작업
amplify	amplify:CreateApp
	amplify:CreateBackendEnvironment
	amplify:CreateBranch
	amplify:CreateDeployment
	amplify:CreateDomainAssociation
	amplify:CreateWebHook
	amplify:DeleteApp
	amplify:DeleteBackendEnvironment
	amplify:DeleteBranch
	amplify:DeleteDomainAssociation
	amplify:DeleteJob
	amplify:DeleteWebHook
	amplify:GenerateAccessLogs
	amplify:GetApp
	amplify:GetArtifactUrl
	amplify:GetBackendEnvironment
	amplify:GetBranch
	amplify:GetDomainAssociation
	amplify:GetJob
	amplify:GetWebHook
	amplify:ListApps

서비스 접두사	작업
	amplify:ListArtifacts amplify:ListBackendEnvironments amplify:ListBranches amplify:ListDomainAssociations amplify:ListJobs amplify:ListWebHooks amplify:StartDeployment amplify:StartJob amplify:StopJob amplify:UpdateApp amplify:UpdateBranch amplify:UpdateDomainAssociation amplify:UpdateWebHook

서비스 접두사	작업
amplifyuibuilder	amplifyuibuilder:CreateComponent amplifyuibuilder:CreateForm amplifyuibuilder:CreateTheme amplifyuibuilder:DeleteComponent amplifyuibuilder:DeleteForm amplifyuibuilder:DeleteTheme amplifyuibuilder:ExportComponents amplifyuibuilder:ExportThemes amplifyuibuilder:GetCodegenJob amplifyuibuilder:ListCodegenJobs amplifyuibuilder:ListComponents amplifyuibuilder:ListForms amplifyuibuilder:ListThemes amplifyuibuilder:ResetMetadataFlag amplifyuibuilder:StartCodegenJob amplifyuibuilder:UpdateComponent amplifyuibuilder:UpdateForm amplifyuibuilder:UpdateTheme

서비스 접두사	작업
app-integrations	app-integrations:CreateApplication
	app-integrations:CreateDataIntegration
	app-integrations:CreateEventIntegration
	app-integrations:DeleteApplication
	app-integrations:DeleteDataIntegration
	app-integrations:DeleteEventIntegration
	app-integrations:GetApplication
	app-integrations:GetDataIntegration
	app-integrations:GetEventIntegration
	app-integrations:ListApplicationAssociations
	app-integrations:ListApplications
	app-integrations:ListDataIntegrationAssociations
	app-integrations:ListDataIntegrations
	app-integrations:ListEventIntegrationAssociations
	app-integrations:ListEventIntegrations
	app-integrations:UpdateApplication
	app-integrations:UpdateDataIntegration
	app-integrations:UpdateEventIntegration

서비스 접두사	작업
appconfig	appconfig:CreateApplication appconfig:CreateConfigurationProfile appconfig:CreateDeploymentStrategy appconfig:CreateEnvironment appconfig:CreateExtension appconfig:CreateExtensionAssociation appconfig:CreateHostedConfigurationVersion appconfig:DeleteApplication appconfig:DeleteConfigurationProfile appconfig:DeleteDeploymentStrategy appconfig:DeleteEnvironment appconfig:DeleteExtension appconfig:DeleteExtensionAssociation appconfig:DeleteHostedConfigurationVersion appconfig:GetApplication appconfig:GetConfiguration appconfig:GetConfigurationProfile appconfig:GetDeployment appconfig:GetDeploymentStrategy appconfig:GetEnvironment appconfig:GetExtension

서비스 접두사	작업
	appconfig:GetExtensionAssociation appconfig:GetHostedConfigurationVersion appconfig:ListApplications appconfig:ListConfigurationProfiles appconfig:ListDeployments appconfig:ListDeploymentStrategies appconfig:ListEnvironments appconfig:ListExtensionAssociations appconfig:ListExtensions appconfig:ListHostedConfigurationVersions appconfig:StartDeployment appconfig:StopDeployment appconfig:UpdateApplication appconfig:UpdateConfigurationProfile appconfig:UpdateDeploymentStrategy appconfig:UpdateEnvironment appconfig:UpdateExtension appconfig:UpdateExtensionAssociation appconfig:ValidateConfiguration

서비스 접두사	작업
appflow	appflow:CancelFlowExecutions
	appflow:CreateConnectorProfile
	appflow:CreateFlow
	appflow:DeleteConnectorProfile
	appflow:DeleteFlow
	appflow:DescribeConnector
	appflow:DescribeConnectorEntity
	appflow:DescribeConnectorProfiles
	appflow:DescribeConnectors
	appflow:DescribeFlow
	appflow:DescribeFlowExecutionRecords
	appflow:ListConnectorEntities
	appflow:ListConnectors
	appflow:ListFlows
	appflow:RegisterConnector
	appflow:ResetConnectorMetadataCache
	appflow:StartFlow
	appflow:StopFlow
	appflow:UnRegisterConnector
	appflow:UpdateConnectorProfile
	appflow:UpdateConnectorRegistration

서비스 접두사	작업
	appflow:UpdateFlow
application-cost-profiler	application-cost-profiler:DeleteReportDefinition application-cost-profiler:GetReportDefinition application-cost-profiler:ImportApplicationUsage application-cost-profiler:ListReportDefinitions application-cost-profiler:PutReportDefinition application-cost-profiler:UpdateReportDefinition

서비스 접두사	작업
applicationinsights	applicationinsights:AddWorkload
	applicationinsights:CreateApplication
	applicationinsights:CreateComponent
	applicationinsights:CreateLogPattern
	applicationinsights:DeleteApplication
	applicationinsights:DeleteComponent
	applicationinsights:DeleteLogPattern
	applicationinsights:DescribeApplication
	applicationinsights:DescribeComponent
	applicationinsights:DescribeComponentConfiguration
	applicationinsights:DescribeComponentConfigurationRecommendation
	applicationinsights:DescribeLogPattern
	applicationinsights:DescribeObservation
	applicationinsights:DescribeProblem
	applicationinsights:DescribeProblemObservations
	applicationinsights:DescribeWorkload
	applicationinsights:ListApplications
	applicationinsights:ListComponents
	applicationinsights:ListConfigurationHistory
	applicationinsights:ListLogPatterns

서비스 접두사	작업
	applicationinsights:ListLogPatternSets applicationinsights:ListProblems applicationinsights:ListWorkloads applicationinsights:RemoveWorkload applicationinsights:UpdateApplication applicationinsights:UpdateComponent applicationinsights:UpdateComponentConfiguration applicationinsights:UpdateLogPattern applicationinsights:UpdateWorkload

서비스 접두사	작업
appmesh	appmesh:CreateGatewayRoute appmesh:CreateMesh appmesh:CreateRoute appmesh:CreateVirtualGateway appmesh:CreateVirtualNode appmesh:CreateVirtualRouter appmesh:CreateVirtualService appmesh:DeleteGatewayRoute appmesh:DeleteMesh appmesh:DeleteRoute appmesh:DeleteVirtualGateway appmesh:DeleteVirtualNode appmesh:DeleteVirtualRouter appmesh:DeleteVirtualService appmesh:DescribeGatewayRoute appmesh:DescribeMesh appmesh:DescribeRoute appmesh:DescribeVirtualGateway appmesh:DescribeVirtualNode appmesh:DescribeVirtualRouter appmesh:DescribeVirtualService

서비스 접두사	작업
	appmesh:ListGatewayRoutes appmesh:ListMeshes appmesh:ListRoutes appmesh:ListVirtualGateways appmesh:ListVirtualNodes appmesh:ListVirtualRouters appmesh:ListVirtualServices appmesh:StreamAggregatedResources appmesh:UpdateGatewayRoute appmesh:UpdateMesh appmesh:UpdateRoute appmesh:UpdateVirtualGateway appmesh:UpdateVirtualNode appmesh:UpdateVirtualRouter appmesh:UpdateVirtualService

서비스 접두사	작업
appstream	appstream:AssociateAppBlockBuilderAppBlock
	appstream:AssociateApplicationFleet
	appstream:AssociateApplicationToEntitlement
	appstream:AssociateFleet
	appstream:BatchAssociateUserStack
	appstream:BatchDisassociateUserStack
	appstream:CopyImage
	appstream:CreateAppBlock
	appstream:CreateAppBlockBuilder
	appstream:CreateAppBlockBuilderStreamingURL
	appstream:CreateApplication
	appstream:CreateDirectoryConfig
	appstream:CreateEntitlement
	appstream:CreateFleet
	appstream:CreateImageBuilder
	appstream:CreateImageBuilderStreamingURL
	appstream:CreateStack
	appstream:CreateStreamingURL
	appstream:CreateUpdatedImage
	appstream:CreateUsageReportSubscription
	appstream:CreateUser

서비스 접두사	작업
	appstream:DeleteAppBlock
	appstream:DeleteAppBlockBuilder
	appstream:DeleteApplication
	appstream:DeleteDirectoryConfig
	appstream:DeleteEntitlement
	appstream:DeleteFleet
	appstream:DeleteImage
	appstream:DeleteImageBuilder
	appstream:DeleteImagePermissions
	appstream:DeleteStack
	appstream:DeleteUsageReportSubscription
	appstream:DeleteUser
	appstream:DescribeAppBlockBuilderAppBlockAssociations
	appstream:DescribeAppBlockBuilders
	appstream:DescribeAppBlocks
	appstream:DescribeApplicationFleetAssociations
	appstream:DescribeApplications
	appstream:DescribeDirectoryConfigs
	appstream:DescribeEntitlements
	appstream:DescribeFleets
	appstream:DescribeImageBuilders

서비스 접두사	작업
	appstream:DescribeImagePermissions
	appstream:DescribeImages
	appstream:DescribeSessions
	appstream:DescribeStacks
	appstream:DescribeUsageReportSubscriptions
	appstream:DescribeUsers
	appstream:DescribeUserStackAssociations
	appstream:DisableUser
	appstream:DisassociateAppBlockBuilderAppBlock
	appstream:DisassociateApplicationFleet
	appstream:DisassociateApplicationFromEntitlement
	appstream:DisassociateFleet
	appstream:EnableUser
	appstream:ExpireSession
	appstream:ListAssociatedFleets
	appstream:ListAssociatedStacks
	appstream:ListEntitledApplications
	appstream:StartAppBlockBuilder
	appstream:StartFleet
	appstream:StartImageBuilder
	appstream:StopAppBlockBuilder

서비스 접두사	작업
	appstream:StopFleet appstream:StopImageBuilder appstream:UpdateAppBlockBuilder appstream:UpdateApplication appstream:UpdateDirectoryConfig appstream:UpdateEntitlement appstream:UpdateFleet appstream:UpdateImagePermissions appstream:UpdateStack

서비스 접두사	작업
appsync	appsync:AssociateApi appsync:AssociateMergedGraphqlApi appsync:AssociateSourceGraphqlApi appsync:CreateApiCache appsync:CreateApiKey appsync:CreateDataSource appsync:CreateDomainName appsync:CreateFunction appsync:CreateGraphqlApi appsync:CreateResolver appsync:CreateType appsync:DeleteApiCache appsync:DeleteApiKey appsync:DeleteDataSource appsync:DeleteDomainName appsync:DeleteFunction appsync:DeleteGraphqlApi appsync:DeleteResolver appsync:DeleteType appsync:DisassociateApi appsync:DisassociateMergedGraphqlApi

서비스 접두사	작업
	appsync:DisassociateSourceGraphqlApi appsync:EvaluateCode appsync:EvaluateMappingTemplate appsync:FlushApiCache appsync:GetApiAssociation appsync:GetApiCache appsync:GetDataSource appsync:GetDataSourceIntrospection appsync:GetDomainName appsync:GetFunction appsync:GetGraphqlApi appsync:GetGraphqlApiEnvironmentVariables appsync:GetIntrospectionSchema appsync:GetResolver appsync:GetSchemaCreationStatus appsync:GetSourceApiAssociation appsync:GetType appsync:ListApiKeys appsync:ListDataSources appsync:ListDomainNames appsync:ListFunctions

서비스 접두사	작업
	appsync:ListGraphqlApis appsync:ListResolvers appsync:ListResolversByFunction appsync:ListSourceApiAssociations appsync:ListTypes appsync:ListTypesByAssociation appsync:PutGraphqlApiEnvironmentVariables appsync:StartDataSourceIntrospection appsync:StartSchemaCreation appsync:StartSchemaMerge appsync:UpdateApiCache appsync:UpdateApiKey appsync:UpdateDataSource appsync:UpdateDomainName appsync:UpdateFunction appsync:UpdateGraphqlApi appsync:UpdateResolver appsync:UpdateSourceApiAssociation appsync:UpdateType

서비스 접두사	작업
aps	aps:CreateAlertManagerDefinition
	aps:CreateLoggingConfiguration
	aps:CreateRuleGroupsNamespace
	aps:CreateScraper
	aps:CreateWorkspace
	aps:DeleteAlertManagerDefinition
	aps:DeleteLoggingConfiguration
	aps:DeleteRuleGroupsNamespace
	aps:DeleteScraper
	aps:DeleteWorkspace
	aps:DescribeAlertManagerDefinition
	aps:DescribeLoggingConfiguration
	aps:DescribeRuleGroupsNamespace
	aps:DescribeScraper
	aps:DescribeWorkspace
	aps:GetDefaultScraperConfiguration
	aps:ListRuleGroupsNamespaces
	aps:ListScrapers
	aps:ListWorkspaces
	aps:PutAlertManagerDefinition
	aps:PutRuleGroupsNamespace

서비스 접두사	작업
	aps:UpdateLoggingConfiguration aps:UpdateWorkspaceAlias

서비스 접두사	작업
athena	athena:BatchGetNamedQuery athena:BatchGetPreparedStatement athena:BatchGetQueryExecution athena:CancelCapacityReservation athena:CreateCapacityReservation athena:CreateDataCatalog athena:CreateNamedQuery athena:CreateNotebook athena:CreatePreparedStatement athena:CreatePresignedNotebookUrl athena:CreateWorkGroup athena:DeleteCapacityReservation athena:DeleteDataCatalog athena:DeleteNamedQuery athena:DeleteNotebook athena:DeletePreparedStatement athena:DeleteWorkGroup athena:ExportNotebook athena:GetCalculationExecution athena:GetCalculationExecutionCode athena:GetCalculationExecutionStatus

서비스 접두사	작업
	athena:GetCapacityAssignmentConfiguration athena:GetCapacityReservation athena:GetDatabase athena:GetDataCatalog athena:GetNamedQuery athena:GetNotebookMetadata athena:GetPreparedStatement athena:GetQueryExecution athena:GetQueryResults athena:GetQueryResultsStream athena:GetQueryRuntimeStatistics athena:GetSession athena:GetSessionStatus athena:GetTableMetadata athena:GetWorkGroup athena:ImportNotebook athena:ListApplicationDPUSizes athena:ListCalculationExecutions athena:ListCapacityReservations athena:ListDatabases athena:ListDataCatalogs

서비스 접두사	작업
	athena:ListEngineVersions athena:ListExecutors athena:ListNamedQueries athena:ListNotebookMetadata athena:ListNotebookSessions athena:ListPreparedStatements athena:ListQueryExecutions athena:ListSessions athena:ListTableMetadata athena:ListWorkGroups athena:PutCapacityAssignmentConfiguration athena:StartCalculationExecution athena:StartQueryExecution athena:StartSession athena:StopCalculationExecution athena:StopQueryExecution athena:TerminateSession athena:UpdateCapacityReservation athena:UpdateDataCatalog athena:UpdateNamedQuery athena:UpdateNotebook

서비스 접두사	작업
	athena:UpdateNotebookMetadata athena:UpdatePreparedStatement athena:UpdateWorkGroup

서비스 접두사	작업
auditmanager	auditmanager:AssociateAssessmentReportEvidenceFolder
	auditmanager:BatchAssociateAssessmentReportEvidence
	auditmanager:BatchCreateDelegationByAssessment
	auditmanager:BatchDeleteDelegationByAssessment
	auditmanager:BatchDisassociateAssessmentReportEvidence
	auditmanager:BatchImportEvidenceToAssessmentControl
	auditmanager:CreateAssessment
	auditmanager:CreateAssessmentFramework
	auditmanager:CreateAssessmentReport
	auditmanager:CreateControl
	auditmanager:DeleteAssessment
	auditmanager:DeleteAssessmentFramework
	auditmanager:DeleteAssessmentFrameworkShare
	auditmanager:DeleteAssessmentReport
	auditmanager:DeleteControl
	auditmanager:DeregisterAccount
	auditmanager:DeregisterOrganizationAdminAccount
	auditmanager:DisassociateAssessmentReportEvidenceFolder
	auditmanager:GetAccountStatus
	auditmanager:GetAssessment
	auditmanager:GetAssessmentFramework

서비스 접두사	작업
	auditmanager:GetAssessmentReportUrl
	auditmanager:GetChangeLogs
	auditmanager:GetControl
	auditmanager:GetDelegations
	auditmanager:GetEvidence
	auditmanager:GetEvidenceByEvidenceFolder
	auditmanager:GetEvidenceFileUploadUrl
	auditmanager:GetEvidenceFolder
	auditmanager:GetEvidenceFoldersByAssessment
	auditmanager:GetEvidenceFoldersByAssessmentControl
	auditmanager:GetInsights
	auditmanager:GetInsightsByAssessment
	auditmanager:GetOrganizationAdminAccount
	auditmanager:GetServicesInScope
	auditmanager:GetSettings
	auditmanager:ListAssessmentControlInsightsByControlDomain
	auditmanager:ListAssessmentFrameworks
	auditmanager:ListAssessmentFrameworkShareRequests
	auditmanager:ListAssessmentReports
	auditmanager:ListAssessments
	auditmanager:ListControlDomainInsights

서비스 접두사	작업
	auditmanager:ListControlDomainInsightsByAssessment
	auditmanager:ListControlInsightsByControlDomain
	auditmanager:ListControls
	auditmanager:ListKeywordsForDataSource
	auditmanager:ListNotifications
	auditmanager:RegisterAccount
	auditmanager:RegisterOrganizationAdminAccount
	auditmanager:StartAssessmentFrameworkShare
	auditmanager:UpdateAssessment
	auditmanager:UpdateAssessmentControl
	auditmanager:UpdateAssessmentControlSetStatus
	auditmanager:UpdateAssessmentFramework
	auditmanager:UpdateAssessmentFrameworkShare
	auditmanager:UpdateAssessmentStatus
	auditmanager:UpdateControl
	auditmanager:UpdateSettings
	auditmanager:ValidateAssessmentReportIntegrity

서비스 접두사	작업
Auto Scaling	autoscaling:AttachInstances autoscaling:AttachLoadBalancers autoscaling:AttachLoadBalancerTargetGroups autoscaling:AttachTrafficSources autoscaling:BatchDeleteScheduledAction autoscaling:BatchPutScheduledUpdateGroupAction autoscaling:CancelInstanceRefresh autoscaling:CompleteLifecycleAction autoscaling:CreateAutoScalingGroup autoscaling:CreateLaunchConfiguration autoscaling:DeleteAutoScalingGroup autoscaling:DeleteLaunchConfiguration autoscaling:DeleteLifecycleHook autoscaling:DeleteNotificationConfiguration autoscaling:DeletePolicy autoscaling:DeleteScheduledAction autoscaling:DeleteWarmPool autoscaling:DescribeAccountLimits autoscaling:DescribeAdjustmentTypes autoscaling:DescribeAutoScalingGroups autoscaling:DescribeAutoScalingInstances

서비스 접두사	작업
	autoscaling:DescribeAutoScalingNotificationTypes autoscaling:DescribeInstanceRefreshes autoscaling:DescribeLaunchConfigurations autoscaling:DescribeLifecycleHooks autoscaling:DescribeLifecycleHookTypes autoscaling:DescribeLoadBalancers autoscaling:DescribeLoadBalancerTargetGroups autoscaling:DescribeMetricCollectionTypes autoscaling:DescribeNotificationConfigurations autoscaling:DescribePolicies autoscaling:DescribeScalingActivities autoscaling:DescribeScalingProcessTypes autoscaling:DescribeScheduledActions autoscaling:DescribeTerminationPolicyTypes autoscaling:DescribeTrafficSources autoscaling:DescribeWarmPool autoscaling:DetachInstances autoscaling:DetachLoadBalancers autoscaling:DetachLoadBalancerTargetGroups autoscaling:DetachTrafficSources autoscaling:DisableMetricsCollection

서비스 접두사	작업
	autoscaling:EnableMetricsCollection autoscaling:EnterStandby autoscaling:ExecutePolicy autoscaling:ExitStandby autoscaling:GetPredictiveScalingForecast autoscaling:PutLifecycleHook autoscaling:PutNotificationConfiguration autoscaling:PutScalingPolicy autoscaling:PutScheduledUpdateGroupAction autoscaling:PutWarmPool autoscaling:RecordLifecycleActionHeartbeat autoscaling:ResumeProcesses autoscaling:RollbackInstanceRefresh autoscaling:SetDesiredCapacity autoscaling:SetInstanceHealth autoscaling:SetInstanceProtection autoscaling:StartInstanceRefresh autoscaling:SuspendProcesses autoscaling:TerminateInstanceInAutoScalingGroup autoscaling:UpdateAutoScalingGroup
aws-marketplace	aws-marketplace:GetEntitlements

서비스 접두사	작업
백업	backup:CancelLegalHold backup:CreateBackupPlan backup:CreateBackupSelection backup:CreateBackupVault backup:CreateFramework backup:CreateLegalHold backup:CreateLogicallyAirGappedBackupVault backup:CreateReportPlan backup:CreateRestoreTestingPlan backup:CreateRestoreTestingSelection backup:DeleteBackupPlan backup:DeleteBackupSelection backup:DeleteBackupVault backup:DeleteBackupVaultAccessPolicy backup:DeleteBackupVaultLockConfiguration backup:DeleteBackupVaultNotifications backup:DeleteFramework backup:DeleteRecoveryPoint backup:DeleteReportPlan backup:DeleteRestoreTestingPlan backup:DeleteRestoreTestingSelection

서비스 접두사	작업
	backup:DescribeBackupJob backup:DescribeBackupVault backup:DescribeCopyJob backup:DescribeFramework backup:DescribeGlobalSettings backup:DescribeProtectedResource backup:DescribeRecoveryPoint backup:DescribeRegionSettings backup:DescribeReportJob backup:DescribeReportPlan backup:DescribeRestoreJob backup:DisassociateRecoveryPoint backup:DisassociateRecoveryPointFromParent backup:ExportBackupPlanTemplate backup:GetBackupPlan backup:GetBackupPlanFromJSON backup:GetBackupPlanFromTemplate backup:GetBackupSelection backup:GetBackupVaultAccessPolicy backup:GetBackupVaultNotifications backup:GetLegalHold

서비스 접두사	작업
	backup:GetRecoveryPointRestoreMetadata backup:GetRestoreJobMetadata backup:GetRestoreTestingInferredMetadata backup:GetRestoreTestingPlan backup:GetRestoreTestingSelection backup:GetSupportedResourceTypes backup:ListBackupJobs backup:ListBackupJobSummaries backup:ListBackupPlans backup:ListBackupPlanTemplates backup:ListBackupPlanVersions backup:ListBackupSelections backup:ListBackupVaults backup:ListCopyJobs backup:ListCopyJobSummaries backup:ListFrameworks backup:ListLegalHolds backup:ListProtectedResources backup:ListRecoveryPointsByBackupVault backup:ListRecoveryPointsByLegalHold backup:ListRecoveryPointsByResource

서비스 접두사	작업
	backup:ListReportJobs backup:ListReportPlans backup:ListRestoreJobs backup:ListRestoreJobsByProtectedResource backup:ListRestoreJobSummaries backup:ListRestoreTestingPlans backup:ListRestoreTestingSelections backup:PutBackupVaultAccessPolicy backup:PutBackupVaultLockConfiguration backup:PutBackupVaultNotifications backup:PutRestoreValidationResult backup:StartBackupJob backup:StartCopyJob backup:StartReportJob backup:StartRestoreJob backup:StopBackupJob backup:UpdateBackupPlan backup:UpdateFramework backup:UpdateGlobalSettings backup:UpdateRecoveryPointLifecycle backup:UpdateRegionSettings

서비스 접두사	작업
	backup:UpdateReportPlan backup:UpdateRestoreTestingPlan backup:UpdateRestoreTestingSelection

서비스 접두사	작업
일괄	batch:CancelJob
	batch:CreateComputeEnvironment
	batch:CreateJobQueue
	batch:CreateSchedulingPolicy
	batch:DeleteComputeEnvironment
	batch:DeleteJobQueue
	batch:DeleteSchedulingPolicy
	batch:DeregisterJobDefinition
	batch:DescribeComputeEnvironments
	batch:DescribeJobDefinitions
	batch:DescribeJobQueues
	batch:DescribeJobs
	batch:DescribeSchedulingPolicies
	batch:GetJobQueueSnapshot
	batch:ListJobs
	batch:ListSchedulingPolicies
	batch:RegisterJobDefinition
	batch:SubmitJob
	batch:TerminateJob
	batch:UpdateComputeEnvironment
	batch:UpdateJobQueue

서비스 접두사	작업
	batch:UpdateSchedulingPolicy
braket	braket:AcceptUserAgreement braket:AccessBraketFeature braket:CancelJob braket:CancelQuantumTask braket:CreateJob braket:CreateQuantumTask braket:GetDevice braket:GetJob braket:GetQuantumTask braket:GetServiceLinkedRoleStatus braket:GetUserAgreementStatus braket:SearchDevices braket:SearchJobs braket:SearchQuantumTasks

서비스 접두사	작업
예산	budgets:ModifyBudget
	budgets:CreateBudgetAction
	budgets:ModifyBudget
	budgets:ModifyBudget
	budgets:ModifyBudget
	budgets:DeleteBudgetAction
	budgets:ModifyBudget
	budgets:ModifyBudget
	budgets:ViewBudget
	budgets:DescribeBudgetAction
	budgets:DescribeBudgetActionHistories
	budgets:DescribeBudgetActionsForAccount
	budgets:DescribeBudgetActionsForBudget
	budgets:ViewBudget
	budgets:ViewBudget
	budgets:ViewBudget
	budgets:ViewBudget
	budgets:ViewBudget
	budgets:ExecuteBudgetAction
	budgets:ModifyBudget
	budgets:UpdateBudgetAction

서비스 접두사	작업
	budgets:ModifyBudget budgets:ModifyBudget
cloud9	cloud9:CreateEnvironmentEC2 cloud9:CreateEnvironmentMembership cloud9:DeleteEnvironment cloud9:DeleteEnvironmentMembership cloud9:DescribeEnvironmentMemberships cloud9:DescribeEnvironments cloud9:DescribeEnvironmentStatus cloud9:ListEnvironments cloud9:UpdateEnvironment cloud9:UpdateEnvironmentMembership

서비스 접두사	작업
cloudformation	cloudformation:BatchDescribeTypeConfigurations
	cloudformation:CancelUpdateStack
	cloudformation:ContinueUpdateRollback
	cloudformation:CreateChangeSet
	cloudformation:CreateGeneratedTemplate
	cloudformation:CreateStack
	cloudformation:CreateStackInstances
	cloudformation:CreateStackSet
	cloudformation:DeactivateType
	cloudformation:DeleteChangeSet
	cloudformation:DeleteGeneratedTemplate
	cloudformation:DeleteStack
	cloudformation:DeleteStackInstances
	cloudformation:DeleteStackSet
	cloudformation:DeregisterType
	cloudformation:DescribeAccountLimits
	cloudformation:DescribeChangeSet
	cloudformation:DescribeChangeSetHooks
	cloudformation:DescribeGeneratedTemplate
	cloudformation:DescribeOrganizationsAccess
	cloudformation:DescribePublisher

서비스 접두사	작업
	cloudformation:DescribeResourceScan
	cloudformation:DescribeStackDriftDetectionStatus
	cloudformation:DescribeStackEvents
	cloudformation:DescribeStackInstance
	cloudformation:DescribeStackResource
	cloudformation:DescribeStackResourceDrifts
	cloudformation:DescribeStackResources
	cloudformation:DescribeStacks
	cloudformation:DescribeStackSet
	cloudformation:DescribeStackSetOperation
	cloudformation:DescribeType
	cloudformation:DescribeTypeRegistration
	cloudformation:DetectStackDrift
	cloudformation:DetectStackResourceDrift
	cloudformation:DetectStackSetDrift
	cloudformation:EstimateTemplateCost
	cloudformation:ExecuteChangeSet
	cloudformation:GetGeneratedTemplate
	cloudformation:GetStackPolicy
	cloudformation:GetTemplate
	cloudformation:GetTemplateSummary

서비스 접두사	작업
	cloudformation:ImportStacksToStackSet
	cloudformation:ListChangeSets
	cloudformation:ListExports
	cloudformation:ListGeneratedTemplates
	cloudformation:ListImports
	cloudformation:ListResourceScanRelatedResources
	cloudformation:ListResourceScanResources
	cloudformation:ListResourceScans
	cloudformation:ListStackInstanceResourceDrifts
	cloudformation:ListStackInstances
	cloudformation:ListStackResources
	cloudformation:ListStackSetAutoDeploymentTargets
	cloudformation:ListStackSetOperationResults
	cloudformation:ListStackSetOperations
	cloudformation:ListStackSets
	cloudformation:ListTypeRegistrations
	cloudformation:ListTypes
	cloudformation:ListTypeVersions
	cloudformation:PublishType
	cloudformation:RecordHandlerProgress
	cloudformation:RegisterPublisher

서비스 접두사	작업
	cloudformation:RegisterType
	cloudformation:RollbackStack
	cloudformation:SetStackPolicy
	cloudformation:SetTypeConfiguration
	cloudformation:SetTypeDefaultVersion
	cloudformation:SignalResource
	cloudformation:StartResourceScan
	cloudformation:StopStackSetOperation
	cloudformation:TestType
	cloudformation:UpdateGeneratedTemplate
	cloudformation:UpdateStack
	cloudformation:UpdateStackInstances
	cloudformation:UpdateStackSet
	cloudformation:UpdateTerminationProtection
	cloudformation:ValidateTemplate

서비스 접두사	작업
cloudfront	cloudfront:AssociateAlias cloudfront:CreateCachePolicy cloudfront:CreateCloudFrontOriginAccessIdentity cloudfront:CreateContinuousDeploymentPolicy cloudfront:CreateFieldLevelEncryptionConfig cloudfront:CreateFieldLevelEncryptionProfile cloudfront:CreateFunction cloudfront:CreateInvalidation cloudfront:CreateKeyGroup cloudfront:CreateKeyValueStore cloudfront:CreateMonitoringSubscription cloudfront:CreateOriginAccessControl cloudfront:CreateOriginRequestPolicy cloudfront:CreatePublicKey cloudfront:CreateRealtimeLogConfig cloudfront:CreateResponseHeadersPolicy cloudfront:DeleteCachePolicy cloudfront:DeleteCloudFrontOriginAccessIdentity cloudfront:DeleteContinuousDeploymentPolicy cloudfront:DeleteDistribution cloudfront:DeleteFieldLevelEncryptionConfig

서비스 접두사	작업
	cloudfront:DeleteFieldLevelEncryptionProfile cloudfront:DeleteFunction cloudfront:DeleteKeyGroup cloudfront:DeleteKeyValueStore cloudfront:DeleteMonitoringSubscription cloudfront:DeleteOriginAccessControl cloudfront:DeleteOriginRequestPolicy cloudfront:DeletePublicKey cloudfront:DeleteRealtimeLogConfig cloudfront:DeleteResponseHeadersPolicy cloudfront:DeleteStreamingDistribution cloudfront:DescribeFunction cloudfront:DescribeKeyValueStore cloudfront:GetCachePolicy cloudfront:GetCachePolicyConfig cloudfront:GetCloudFrontOriginAccessIdentity cloudfront:GetCloudFrontOriginAccessIdentityConfig cloudfront:GetContinuousDeploymentPolicy cloudfront:GetContinuousDeploymentPolicyConfig cloudfront:GetDistributionConfig cloudfront:GetFieldLevelEncryption

서비스 접두사	작업
	cloudfront:GetFieldLevelEncryptionConfig cloudfront:GetFieldLevelEncryptionProfile cloudfront:GetFieldLevelEncryptionProfileConfig cloudfront:GetFunction cloudfront:GetInvalidation cloudfront:GetKeyGroup cloudfront:GetKeyGroupConfig cloudfront:GetMonitoringSubscription cloudfront:GetOriginAccessControl cloudfront:GetOriginAccessControlConfig cloudfront:GetOriginRequestPolicy cloudfront:GetOriginRequestPolicyConfig cloudfront:GetPublicKey cloudfront:GetPublicKeyConfig cloudfront:GetRealtimeLogConfig cloudfront:GetResponseHeadersPolicy cloudfront:GetResponseHeadersPolicyConfig cloudfront:GetStreamingDistribution cloudfront:GetStreamingDistributionConfig cloudfront:ListCachePolicies cloudfront:ListCloudFrontOriginAccessIdentities

서비스 접두사	작업
	cloudfront:ListConflictingAliases cloudfront:ListContinuousDeploymentPolicies cloudfront:ListDistributions cloudfront:ListDistributionsByCachePolicyId cloudfront:ListDistributionsByKeyGroup cloudfront:ListDistributionsByOriginRequestPolicyId cloudfront:ListDistributionsByRealtimeLogConfig cloudfront:ListDistributionsByResponseHeadersPolicyId cloudfront:ListDistributionsByWebACLId cloudfront:ListFieldLevelEncryptionConfigs cloudfront:ListFieldLevelEncryptionProfiles cloudfront:ListFunctions cloudfront:ListInvalidations cloudfront:ListKeyGroups cloudfront:ListKeyValueStores cloudfront:ListOriginAccessControls cloudfront:ListOriginRequestPolicies cloudfront:ListPublicKeys cloudfront:ListRealtimeLogConfigs cloudfront:ListResponseHeadersPolicies cloudfront:ListStreamingDistributions

서비스 접두사	작업
	cloudfront:PublishFunction cloudfront:TestFunction cloudfront:UpdateCachePolicy cloudfront:UpdateCloudFrontOriginAccessIdentity cloudfront:UpdateContinuousDeploymentPolicy cloudfront:UpdateDistribution cloudfront:UpdateFieldLevelEncryptionConfig cloudfront:UpdateFieldLevelEncryptionProfile cloudfront:UpdateFunction cloudfront:UpdateKeyGroup cloudfront:UpdateKeyValueStore cloudfront:UpdateOriginAccessControl cloudfront:UpdateOriginRequestPolicy cloudfront:UpdatePublicKey cloudfront:UpdateRealtimeLogConfig cloudfront:UpdateResponseHeadersPolicy

서비스 접두사	작업
cloudhsm	cloudhsm:CreateHsm
	cloudhsm:DeleteBackup
	cloudhsm:DeleteHsm
	cloudhsm:DeleteResourcePolicy
	cloudhsm:DescribeBackups
	cloudhsm:DescribeClusters
	cloudhsm:GetResourcePolicy
	cloudhsm:InitializeCluster
	cloudhsm:ModifyBackupAttributes
	cloudhsm:ModifyCluster
	cloudhsm:PutResourcePolicy
	cloudhsm:RestoreBackup

서비스 접두사	작업
cloudsearch	cloudsearch:BuildSuggesters cloudsearch:CreateDomain cloudsearch:DefineAnalysisScheme cloudsearch:DefineExpression cloudsearch:DefineIndexField cloudsearch:DefineSuggester cloudsearch:DeleteAnalysisScheme cloudsearch:DeleteDomain cloudsearch:DeleteExpression cloudsearch:DeleteIndexField cloudsearch:DeleteSuggester cloudsearch:DescribeAnalysisSchemes cloudsearch:DescribeAvailabilityOptions cloudsearch:DescribeDomainEndpointOptions cloudsearch:DescribeDomains cloudsearch:DescribeExpressions cloudsearch:DescribeIndexFields cloudsearch:DescribeScalingParameters cloudsearch:DescribeServiceAccessPolicies cloudsearch:DescribeSuggesters cloudsearch:IndexDocuments

서비스 접두사	작업
	cloudsearch:ListDomainNames cloudsearch:UpdateAvailabilityOptions cloudsearch:UpdateDomainEndpointOptions cloudsearch:UpdateScalingParameters cloudsearch:UpdateServiceAccessPolicies

서비스 접두사	작업
cloudtrail	cloudtrail:CancelQuery
	cloudtrail:CreateChannel
	cloudtrail:CreateEventDataStore
	cloudtrail:CreateTrail
	cloudtrail:DeleteChannel
	cloudtrail:DeleteEventDataStore
	cloudtrail:DeleteResourcePolicy
	cloudtrail:DeleteTrail
	cloudtrail:DeregisterOrganizationDelegatedAdmin
	cloudtrail:DescribeQuery
	cloudtrail:DescribeTrails
	cloudtrail:DisableFederation
	cloudtrail:GetChannel
	cloudtrail:GetEventDataStore
	cloudtrail:GetEventDataStoreData
	cloudtrail:GetEventSelectors
	cloudtrail:GetImport
	cloudtrail:GetInsightSelectors
	cloudtrail:GetQueryResults
	cloudtrail:GetResourcePolicy
	cloudtrail:GetTrail

서비스 접두사	작업
	cloudtrail:GetTrailStatus
	cloudtrail:ListChannels
	cloudtrail:ListEventDataStores
	cloudtrail:ListImportFailures
	cloudtrail:ListImports
	cloudtrail:ListPublicKeys
	cloudtrail:ListQueries
	cloudtrail:ListTrails
	cloudtrail:LookupEvents
	cloudtrail:PutEventSelectors
	cloudtrail:PutInsightSelectors
	cloudtrail:PutResourcePolicy
	cloudtrail:RegisterOrganizationDelegatedAdmin
	cloudtrail:RestoreEventDataStore
	cloudtrail:StartEventDataStoreIngestion
	cloudtrail:StartImport
	cloudtrail:StartLogging
	cloudtrail:StartQuery
	cloudtrail:StopEventDataStoreIngestion
	cloudtrail:StopImport
	cloudtrail:StopLogging

서비스 접두사	작업
	cloudtrail:UpdateChannel cloudtrail:UpdateEventDataStore cloudtrail:UpdateTrail

서비스 접두사	작업
cloudwatch	cloudwatch:DeleteAlarms cloudwatch:DeleteAnomalyDetector cloudwatch:DeleteDashboards cloudwatch:DeleteInsightRules cloudwatch:DeleteMetricStream cloudwatch:DescribeAlarmHistory cloudwatch:DescribeAlarms cloudwatch:DescribeAlarmsForMetric cloudwatch:DescribeAnomalyDetectors cloudwatch:DescribeInsightRules cloudwatch:DisableAlarmActions cloudwatch:DisableInsightRules cloudwatch:EnableAlarmActions cloudwatch:EnableInsightRules cloudwatch:GetDashboard cloudwatch:GetInsightRuleReport cloudwatch:GetMetricStream cloudwatch:ListDashboards cloudwatch:ListManagedInsightRules cloudwatch:ListMetricStreams cloudwatch:PutAnomalyDetector

서비스 접두사	작업
	cloudwatch:PutCompositeAlarm cloudwatch:PutDashboard cloudwatch:PutInsightRule cloudwatch:PutManagedInsightRules cloudwatch:PutMetricAlarm cloudwatch:PutMetricStream cloudwatch:SetAlarmState cloudwatch:StartMetricStreams cloudwatch:StopMetricStreams

서비스 접두사	작업
codeartifact	codeartifact:AssociateExternalConnection
	codeartifact:CopyPackageVersions
	codeartifact:CreateDomain
	codeartifact:CreateRepository
	codeartifact:DeleteDomain
	codeartifact:DeleteDomainPermissionsPolicy
	codeartifact:DeletePackage
	codeartifact:DeletePackageVersions
	codeartifact:DeleteRepository
	codeartifact:DeleteRepositoryPermissionsPolicy
	codeartifact:DescribeDomain
	codeartifact:DescribePackage
	codeartifact:DescribePackageVersion
	codeartifact:DescribeRepository
	codeartifact:DisassociateExternalConnection
	codeartifact:DisposePackageVersions
	codeartifact:GetAssociatedPackageGroup
	codeartifact:GetAuthorizationToken
	codeartifact:GetDomainPermissionsPolicy
	codeartifact:GetPackageVersionAsset
	codeartifact:GetPackageVersionReadme

서비스 접두사	작업
	codeartifact:GetRepositoryEndpoint
	codeartifact:GetRepositoryPermissionsPolicy
	codeartifact:ListDomains
	codeartifact:ListPackageGroups
	codeartifact:ListPackages
	codeartifact:ListPackageVersionAssets
	codeartifact:ListPackageVersionDependencies
	codeartifact:ListPackageVersions
	codeartifact:ListRepositories
	codeartifact:ListRepositoriesInDomain
	codeartifact:PublishPackageVersion
	codeartifact:PutDomainPermissionsPolicy
	codeartifact:PutPackageMetadata
	codeartifact:PutPackageOriginConfiguration
	codeartifact:PutRepositoryPermissionsPolicy
	codeartifact:ReadFromRepository
	codeartifact:UpdatePackageVersionsStatus
	codeartifact:UpdateRepository

서비스 접두사	작업
codedeploy	codedeploy:BatchGetApplicationRevisions codedeploy:BatchGetApplications codedeploy:BatchGetDeploymentGroups codedeploy:BatchGetDeploymentInstances codedeploy:BatchGetDeployments codedeploy:BatchGetDeploymentTargets codedeploy:BatchGetOnPremisesInstances codedeploy:ContinueDeployment codedeploy:CreateApplication codedeploy:CreateDeployment codedeploy:CreateDeploymentConfig codedeploy:CreateDeploymentGroup codedeploy:DeleteApplication codedeploy:DeleteDeploymentConfig codedeploy:DeleteDeploymentGroup codedeploy:DeleteGitHubAccountToken codedeploy:DeleteResourcesByExternalId codedeploy:DeregisterOnPremisesInstance codedeploy:GetApplication codedeploy:GetApplicationRevision codedeploy:GetDeployment

서비스 접두사	작업
	codedeploy:GetDeploymentConfig codedeploy:GetDeploymentGroup codedeploy:GetDeploymentInstance codedeploy:GetDeploymentTarget codedeploy:GetOnPremisesInstance codedeploy:ListApplicationRevisions codedeploy:ListApplications codedeploy:ListDeploymentConfigs codedeploy:ListDeploymentGroups codedeploy:ListDeploymentInstances codedeploy:ListDeployments codedeploy:ListDeploymentTargets codedeploy:ListGitHubAccountTokenNames codedeploy:ListOnPremisesInstances codedeploy:PutLifecycleEventHookExecutionStatus codedeploy:RegisterApplicationRevision codedeploy:RegisterOnPremisesInstance codedeploy:SkipWaitTimeForInstanceTermination codedeploy:StopDeployment codedeploy:UpdateApplication codedeploy:UpdateDeploymentGroup

서비스 접두사	작업
codeguru-profiler	codeguru-profiler:AddNotificationChannels
	codeguru-profiler:BatchGetFrameMetricData
	codeguru-profiler:ConfigureAgent
	codeguru-profiler:CreateProfilingGroup
	codeguru-profiler:DeleteProfilingGroup
	codeguru-profiler:DescribeProfilingGroup
	codeguru-profiler:GetFindingsReportAccountSummary
	codeguru-profiler:GetNotificationConfiguration
	codeguru-profiler:GetPolicy
	codeguru-profiler:GetProfile
	codeguru-profiler:GetRecommendations
	codeguru-profiler:ListFindingsReports
	codeguru-profiler:ListProfileTimes
	codeguru-profiler:ListProfilingGroups
	codeguru-profiler:PutPermission
	codeguru-profiler:RemoveNotificationChannel
	codeguru-profiler:RemovePermission
	codeguru-profiler:SubmitFeedback
	codeguru-profiler:UpdateProfilingGroup

서비스 접두사	작업
codeguru-reviewer	codeguru-reviewer:AssociateRepository codeguru-reviewer:CreateCodeReview codeguru-reviewer:DescribeCodeReview codeguru-reviewer:DescribeRecommendationFeedback codeguru-reviewer:DescribeRepositoryAssociation codeguru-reviewer:DisassociateRepository codeguru-reviewer:ListCodeReviews codeguru-reviewer:ListRecommendationFeedback codeguru-reviewer:ListRecommendations codeguru-reviewer:ListRepositoryAssociations codeguru-reviewer:PutRecommendationFeedback

서비스 접두사	작업
CodePipeline	codepipeline:AcknowledgeJob
	codepipeline:AcknowledgeThirdPartyJob
	codepipeline:CreateCustomActionType
	codepipeline:CreatePipeline
	codepipeline:DeleteCustomActionType
	codepipeline:DeletePipeline
	codepipeline:DeleteWebhook
	codepipeline:DeregisterWebhookWithThirdParty
	codepipeline:GetActionType
	codepipeline:GetJobDetails
	codepipeline:GetPipeline
	codepipeline:GetPipelineExecution
	codepipeline:GetPipelineState
	codepipeline:GetThirdPartyJobDetails
	codepipeline:ListActionExecutions
	codepipeline:ListActionTypes
	codepipeline:ListPipelineExecutions
	codepipeline:ListPipelines
	codepipeline:ListWebhooks
	codepipeline:PollForJobs
	codepipeline:PollForThirdPartyJobs

서비스 접두사	작업
	codepipeline:PutActionRevision codepipeline:PutApprovalResult codepipeline:PutJobFailureResult codepipeline:PutJobSuccessResult codepipeline:PutThirdPartyJobFailureResult codepipeline:PutThirdPartyJobSuccessResult codepipeline:PutWebhook codepipeline:RegisterWebhookWithThirdParty codepipeline:RollbackStage codepipeline:StartPipelineExecution codepipeline:StopPipelineExecution codepipeline:UpdateActionType codepipeline:UpdatePipeline

서비스 접두사	작업
codestar	codestar:AssociateTeamMember codestar:CreateProject codestar:CreateUserProfile codestar:DeleteProject codestar:DeleteUserProfile codestar:DescribeProject codestar:DescribeUserProfile codestar:DisassociateTeamMember codestar:ListProjects codestar:ListResources codestar:ListTeamMembers codestar:ListUserProfiles codestar:UpdateProject codestar:UpdateTeamMember codestar:UpdateUserProfile

서비스 접두사	작업
codestar-notifications	codestar-notifications:CreateNotificationRule
	codestar-notifications:DeleteNotificationRule
	codestar-notifications:DeleteTarget
	codestar-notifications:DescribeNotificationRule
	codestar-notifications:ListEventTypes
	codestar-notifications:ListNotificationRules
	codestar-notifications:ListTargets
	codestar-notifications:Subscribe
	codestar-notifications:Unsubscribe
	codestar-notifications:UpdateNotificationRule

서비스 접두사	작업
cognito-identity	cognito-identity:CreateIdentityPool cognito-identity:DeleteIdentities cognito-identity:DeleteIdentityPool cognito-identity:DescribeIdentity cognito-identity:DescribeIdentityPool cognito-identity:GetIdentityPoolRoles cognito-identity:ListIdentities cognito-identity:ListIdentityPools cognito-identity:LookupDeveloperIdentity cognito-identity:MergeDeveloperIdentities cognito-identity:SetIdentityPoolRoles cognito-identity:UnlinkDeveloperIdentity cognito-identity:UpdateIdentityPool

서비스 접두사	작업
cognito-idp	cognito-idp:AddCustomAttributes
	cognito-idp:AdminAddUserToGroup
	cognito-idp:AdminConfirmSignUp
	cognito-idp:AdminCreateUser
	cognito-idp:AdminDeleteUser
	cognito-idp:AdminDeleteUserAttributes
	cognito-idp:AdminDisableProviderForUser
	cognito-idp:AdminDisableUser
	cognito-idp:AdminEnableUser
	cognito-idp:AdminForgetDevice
	cognito-idp:AdminGetDevice
	cognito-idp:AdminGetUser
	cognito-idp:AdminInitiateAuth
	cognito-idp:AdminLinkProviderForUser
	cognito-idp:AdminListDevices
	cognito-idp:AdminListGroupsForUser
	cognito-idp:AdminListUserAuthEvents
	cognito-idp:AdminRemoveUserFromGroup
	cognito-idp:AdminResetUserPassword
	cognito-idp:AdminRespondToAuthChallenge
	cognito-idp:AdminSetUserMFAPreference

서비스 접두사	작업
	cognito-idp:AdminSetUserPassword
	cognito-idp:AdminSetUserSettings
	cognito-idp:AdminUpdateAuthEventFeedback
	cognito-idp:AdminUpdateDeviceStatus
	cognito-idp:AdminUpdateUserAttributes
	cognito-idp:AdminUserGlobalSignOut
	cognito-idp:AssociateSoftwareToken
	cognito-idp:ChangePassword
	cognito-idp:ConfirmDevice
	cognito-idp:ConfirmForgotPassword
	cognito-idp:ConfirmSignUp
	cognito-idp:CreateGroup
	cognito-idp:CreateIdentityProvider
	cognito-idp:CreateResourceServer
	cognito-idp:CreateUserImportJob
	cognito-idp:CreateUserPool
	cognito-idp:CreateUserPoolClient
	cognito-idp:CreateUserPoolDomain
	cognito-idp:DeleteGroup
	cognito-idp:DeleteIdentityProvider
	cognito-idp:DeleteResourceServer

서비스 접두사	작업
	cognito-idp:DeleteUser
	cognito-idp:DeleteUserAttributes
	cognito-idp:DeleteUserPool
	cognito-idp:DeleteUserPoolClient
	cognito-idp:DeleteUserPoolDomain
	cognito-idp:DescribeIdentityProvider
	cognito-idp:DescribeResourceServer
	cognito-idp:DescribeRiskConfiguration
	cognito-idp:DescribeUserImportJob
	cognito-idp:DescribeUserPool
	cognito-idp:DescribeUserPoolClient
	cognito-idp:DescribeUserPoolDomain
	cognito-idp:ForgetDevice
	cognito-idp:ForgotPassword
	cognito-idp:GetCSVHeader
	cognito-idp:GetDevice
	cognito-idp:GetGroup
	cognito-idp:GetIdentityProviderByIdentifier
	cognito-idp:GetLogDeliveryConfiguration
	cognito-idp:GetSigningCertificate
	cognito-idp:GetUICustomization

서비스 접두사	작업
	cognito-idp:GetUser
	cognito-idp:GetUserAttributeVerificationCode
	cognito-idp:GetUserPoolMfaConfig
	cognito-idp:GlobalSignOut
	cognito-idp:InitiateAuth
	cognito-idp:ListDevices
	cognito-idp:ListGroups
	cognito-idp:ListIdentityProviders
	cognito-idp:ListResourceServers
	cognito-idp:ListUserImportJobs
	cognito-idp:ListUserPoolClients
	cognito-idp:ListUserPools
	cognito-idp:ListUsers
	cognito-idp:ListUsersInGroup
	cognito-idp:ResendConfirmationCode
	cognito-idp:RespondToAuthChallenge
	cognito-idp:RevokeToken
	cognito-idp:SetLogDeliveryConfiguration
	cognito-idp:SetRiskConfiguration
	cognito-idp:SetUICustomization
	cognito-idp:SetUserMFAPreference

서비스 접두사	작업
	cognito-idp:SetUserPoolMfaConfig
	cognito-idp:SetUserSettings
	cognito-idp:SignUp
	cognito-idp:StartUserImportJob
	cognito-idp:StopUserImportJob
	cognito-idp:UpdateAuthEventFeedback
	cognito-idp:UpdateDeviceStatus
	cognito-idp:UpdateGroup
	cognito-idp:UpdateIdentityProvider
	cognito-idp:UpdateResourceServer
	cognito-idp:UpdateUserAttributes
	cognito-idp:UpdateUserPool
	cognito-idp:UpdateUserPoolClient
	cognito-idp:UpdateUserPoolDomain
	cognito-idp:VerifySoftwareToken
	cognito-idp:VerifyUserAttribute

서비스 접두사	작업
cognito-sync	cognito-sync:BulkPublish cognito-sync:DeleteDataset cognito-sync:DescribeDataset cognito-sync:DescribeIdentityPoolUsage cognito-sync:DescribeIdentityUsage cognito-sync:GetBulkPublishDetails cognito-sync:GetCognitoEvents cognito-sync:GetIdentityPoolConfiguration cognito-sync:ListDatasets cognito-sync:ListIdentityPoolUsage cognito-sync:ListRecords cognito-sync:RegisterDevice cognito-sync:SetCognitoEvents cognito-sync:SetIdentityPoolConfiguration cognito-sync:SubscribeToDataset cognito-sync:UnsubscribeFromDataset cognito-sync:UpdateRecords

서비스 접두사	작업
comprehendmedical	comprehendmedical:DescribeEntitiesDetectionV2Job
	comprehendmedical:DescribeICD10CMInferenceJob
	comprehendmedical:DescribePHIDetectionJob
	comprehendmedical:DescribeRxNormInferenceJob
	comprehendmedical:DescribeSNOMEDCTInferenceJob
	comprehendmedical:DetectEntitiesV2
	comprehendmedical:DetectPHI
	comprehendmedical:InferICD10CM
	comprehendmedical:InferRxNorm
	comprehendmedical:InferSNOMEDCT
	comprehendmedical:ListEntitiesDetectionV2Jobs
	comprehendmedical:ListICD10CMInferenceJobs
	comprehendmedical:ListPHIDetectionJobs
	comprehendmedical:ListRxNormInferenceJobs
	comprehendmedical:ListSNOMEDCTInferenceJobs
	comprehendmedical:StartEntitiesDetectionV2Job
	comprehendmedical:StartICD10CMInferenceJob
	comprehendmedical:StartPHIDetectionJob
	comprehendmedical:StartRxNormInferenceJob
	comprehendmedical:StartSNOMEDCTInferenceJob
	comprehendmedical:StopEntitiesDetectionV2Job

서비스 접두사	작업
	comprehendmedical:StopICD10CMInferenceJob comprehendmedical:StopPHIDetectionJob comprehendmedical:StopRxNormInferenceJob comprehendmedical:StopSNOMEDCTInferenceJob

서비스 접두사	작업
compute-optimizer	compute-optimizer:DeleteRecommendationPreferences compute-optimizer:DescribeRecommendationExportJobs compute-optimizer:ExportAutoScalingGroupRecommendations compute-optimizer:ExportEBSVolumeRecommendations compute-optimizer:ExportEC2InstanceRecommendations compute-optimizer:ExportECSServiceRecommendations compute-optimizer:ExportLambdaFunctionRecommendations compute-optimizer:ExportLicenseRecommendations compute-optimizer:ExportRDSDatabaseRecommendations compute-optimizer:GetEC2RecommendationProjectedMetrics compute-optimizer:GetECSServiceRecommendationProjectedMetrics compute-optimizer:GetEffectiveRecommendationPreferences compute-optimizer:GetEnrollmentStatus compute-optimizer:GetEnrollmentStatusesForOrganization compute-optimizer:GetRDSDatabaseRecommendationProjectedMetrics compute-optimizer:GetRecommendationPreferences compute-optimizer:GetRecommendationSummaries compute-optimizer:PutRecommendationPreferences compute-optimizer:UpdateEnrollmentStatus

서비스 접두사	작업
config	config:BatchGetResourceConfig
	config:DeleteAggregationAuthorization
	config:DeleteConfigRule
	config:DeleteConfigurationAggregator
	config:DeleteConfigurationRecorder
	config:DeleteConformancePack
	config:DeleteDeliveryChannel
	config:DeleteEvaluationResults
	config:DeleteOrganizationConfigRule
	config:DeleteOrganizationConformancePack
	config:DeletePendingAggregationRequest
	config:DeleteRemediationConfiguration
	config:DeleteRemediationExceptions
	config:DeleteResourceConfig
	config:DeleteRetentionConfiguration
	config:DeleteStoredQuery
	config:DeliverConfigSnapshot
	config:DescribeAggregateComplianceByConfigRules
	config:DescribeAggregateComplianceByConformancePacks
	config:DescribeAggregationAuthorizations
	config:DescribeComplianceByConfigRule

서비스 접두사	작업
	config:DescribeComplianceByResource
	config:DescribeConfigRuleEvaluationStatus
	config:DescribeConfigRules
	config:DescribeConfigurationAggregators
	config:DescribeConfigurationAggregatorSourcesStatus
	config:DescribeConfigurationRecorders
	config:DescribeConfigurationRecorderStatus
	config:DescribeConformancePackCompliance
	config:DescribeConformancePacks
	config:DescribeConformancePackStatus
	config:DescribeDeliveryChannels
	config:DescribeDeliveryChannelStatus
	config:DescribeOrganizationConfigRules
	config:DescribeOrganizationConfigRuleStatuses
	config:DescribeOrganizationConformancePacks
	config:DescribeOrganizationConformancePackStatuses
	config:DescribePendingAggregationRequests
	config:DescribeRemediationConfigurations
	config:DescribeRemediationExceptions
	config:DescribeRemediationExecutionStatus
	config:DescribeRetentionConfigurations

서비스 접두사	작업
	config:GetComplianceDetailsByConfigRule
	config:GetComplianceDetailsByResource
	config:GetComplianceSummaryByConfigRule
	config:GetComplianceSummaryByResourceType
	config:GetConformancePackComplianceDetails
	config:GetConformancePackComplianceSummary
	config:GetCustomRulePolicy
	config:GetDiscoveredResourceCounts
	config:GetOrganizationConfigRuleDetailedStatus
	config:GetOrganizationConformancePackDetailedStatus
	config:GetOrganizationCustomRulePolicy
	config:GetResourceConfigHistory
	config:GetResourceEvaluationSummary
	config:GetStoredQuery
	config:ListConformancePackComplianceScores
	config:ListDiscoveredResources
	config:ListResourceEvaluations
	config:ListStoredQueries
	config:PutConfigRule
	config:PutConfigurationAggregator
	config:PutConfigurationRecorder

서비스 접두사	작업
	config:PutConformancePack
	config:PutDeliveryChannel
	config:PutEvaluations
	config:PutExternalEvaluation
	config:PutOrganizationConfigRule
	config:PutOrganizationConformancePack
	config:PutRemediationConfigurations
	config:PutRemediationExceptions
	config:PutResourceConfig
	config:PutRetentionConfiguration
	config:PutStoredQuery
	config:SelectResourceConfig
	config:StartConfigRulesEvaluation
	config:StartConfigurationRecorder
	config:StartRemediationExecution
	config:StartResourceEvaluation
	config:StopConfigurationRecorder

서비스 접두사	작업
연결	connect:ActivateEvaluationForm connect:AssociateApprovedOrigin connect:AssociateBot connect:AssociateDefaultVocabulary connect:AssociateFlow connect:AssociateInstanceStorageConfig connect:AssociateLambdaFunction connect:AssociateLexBot connect:AssociatePhoneNumberContactFlow connect:AssociateQueueQuickConnects connect:AssociateRoutingProfileQueues connect:AssociateSecurityKey connect:AssociateUserProficiencies connect:BatchGetFlowAssociation connect:BatchPutContact connect:ClaimPhoneNumber connect:CreateAgentStatus connect:CreateContactFlow connect:CreateContactFlowModule connect:CreateEvaluationForm connect:CreateHoursOfOperation

서비스 접두사	작업
	connect:CreateInstance connect:CreateIntegrationAssociation connect:CreateParticipant connect:CreatePersistentContactAssociation connect:CreatePredefinedAttribute connect:CreatePrompt connect:CreateQueue connect:CreateQuickConnect connect:CreateRoutingProfile connect:CreateRule connect:CreateSecurityProfile connect:CreateTaskTemplate connect:CreateTrafficDistributionGroup connect:CreateUseCase connect:CreateUser connect:CreateUserHierarchyGroup connect:CreateView connect:CreateViewVersion connect:CreateVocabulary connect:DeactivateEvaluationForm connect:DeleteContactEvaluation

서비스 접두사	작업
	connect:DeleteContactFlow connect:DeleteContactFlowModule connect:DeleteEvaluationForm connect:DeleteHoursOfOperation connect:DeleteInstance connect:DeleteIntegrationAssociation connect:DeletePredefinedAttribute connect:DeletePrompt connect:DeleteQueue connect:DeleteQuickConnect connect:DeleteRoutingProfile connect:DeleteRule connect:DeleteSecurityProfile connect:DeleteTaskTemplate connect:DeleteTrafficDistributionGroup connect:DeleteUseCase connect:DeleteUser connect:DeleteUserHierarchyGroup connect:DeleteView connect:DeleteVocabulary connect:DescribeAgentStatus

서비스 접두사	작업
	connect:DescribeAuthenticationProfile connect:DescribeContactEvaluation connect:DescribeEvaluationForm connect:DescribeInstanceAttribute connect:DescribeInstanceStorageConfig connect:DescribePhoneNumber connect:DescribeRule connect:DescribeTrafficDistributionGroup connect:DescribeUserHierarchyGroup connect:DescribeUserHierarchyStructure connect:DescribeView connect:DescribeVocabulary connect:DisassociateApprovedOrigin connect:DisassociateBot connect:DisassociateFlow connect:DisassociateInstanceStorageConfig connect:DisassociateLambdaFunction connect:DisassociateLexBot connect:DisassociatePhoneNumberContactFlow connect:DisassociateQueueQuickConnects connect:DisassociateRoutingProfileQueues

서비스 접두사	작업
	connect:DisassociateSecurityKey connect:DisassociateUserProficiencies connect:DismissUserContact connect:GetContactAttributes connect:GetCurrentMetricData connect:GetCurrentUserData connect:GetFederationToken connect:GetFlowAssociation connect:GetMetricData connect:GetMetricDataV2 connect:GetPromptFile connect:GetTaskTemplate connect:GetTrafficDistribution connect:ImportPhoneNumber connect:ListApprovedOrigins connect:ListAuthenticationProfiles connect:ListBots connect:ListContactEvaluations connect:ListContactFlowModules connect:ListContactFlows connect:ListContactReferences

서비스 접두사	작업
	connect:ListDefaultVocabularies connect:ListEvaluationForms connect:ListEvaluationFormVersions connect:ListFlowAssociations connect:ListHoursOfOperations connect:ListInstanceAttributes connect:ListInstanceStorageConfigs connect:ListIntegrationAssociations connect:ListLambdaFunctions connect:ListLexBots connect:ListPhoneNumbers connect:ListPhoneNumbersV2 connect:ListPredefinedAttributes connect:ListPrompts connect:ListQueueQuickConnects connect:ListQueues connect:ListQuickConnects connect:ListRealtimeContactAnalysisSegmentsV2 connect:ListRoutingProfileQueues connect:ListRoutingProfiles connect:ListRules

서비스 접두사	작업
	connect:ListSecurityKeys connect:ListSecurityProfileApplications connect:ListSecurityProfilePermissions connect:ListSecurityProfiles connect:ListTaskTemplates connect:ListTrafficDistributionGroups connect:ListUseCases connect:ListUserHierarchyGroups connect:ListUserProficiencies connect:ListUsers connect:ListViews connect:ListViewVersions connect:MonitorContact connect:PauseContact connect:PutUserStatus connect:ReleasePhoneNumber connect:ReplicateInstance connect:ResumeContact connect:ResumeContactRecording connect:SearchAvailablePhoneNumbers connect:SearchContactFlowModules

서비스 접두사	작업
	connect:SearchContactFlows connect:SearchContacts connect:SearchHoursOfOperations connect:SearchPredefinedAttributes connect:SearchPrompts connect:SearchQueues connect:SearchQuickConnects connect:SearchRoutingProfiles connect:SearchSecurityProfiles connect:SearchVocabularies connect:SendChatIntegrationEvent connect:StartChatContact connect:StartContactEvaluation connect:StartContactRecording connect:StartContactStreaming connect:StartOutboundVoiceContact connect:StartTaskContact connect:StartWebRTCContact connect:StopContact connect:StopContactRecording connect:StopContactStreaming

서비스 접두사	작업
	connect:SubmitContactEvaluation connect:SuspendContactRecording connect:TransferContact connect:UpdateAgentStatus connect:UpdateAuthenticationProfile connect:UpdateContact connect:UpdateContactAttributes connect:UpdateContactEvaluation connect:UpdateContactFlowContent connect:UpdateContactFlowMetadata connect:UpdateContactFlowModuleContent connect:UpdateContactFlowModuleMetadata connect:UpdateContactFlowName connect:UpdateContactRoutingData connect:UpdateContactSchedule connect:UpdateEvaluationForm connect:UpdateHoursOfOperation connect:UpdateInstanceAttribute connect:UpdateInstanceStorageConfig connect:UpdateParticipantRoleConfig connect:UpdatePhoneNumber

서비스 접두사	작업
	connect:UpdatePhoneNumberMetadata connect:UpdatePredefinedAttribute connect:UpdatePrompt connect:UpdateQueueHoursOfOperation connect:UpdateQueueMaxContacts connect:UpdateQueueName connect:UpdateQueueOutboundCallerConfig connect:UpdateQueueStatus connect:UpdateQuickConnectConfig connect:UpdateQuickConnectName connect:UpdateRoutingProfileAgentAvailabilityTimer connect:UpdateRoutingProfileConcurrency connect:UpdateRoutingProfileDefaultOutboundQueue connect:UpdateRoutingProfileName connect:UpdateRoutingProfileQueues connect:UpdateRule connect:UpdateSecurityProfile connect:UpdateTaskTemplate connect:UpdateTrafficDistribution connect:UpdateUserHierarchy connect:UpdateUserHierarchyGroupName

서비스 접두사	작업
	connect:UpdateUserHierarchyStructure connect:UpdateUserIdentityInfo connect:UpdateUserPhoneConfig connect:UpdateUserProficiencies connect:UpdateUserRoutingProfile connect:UpdateUserSecurityProfiles connect:UpdateViewContent connect:UpdateViewMetadata
cur	cur:DeleteReportDefinition cur:DescribeReportDefinitions cur:ModifyReportDefinition cur:PutReportDefinition

서비스 접두사	작업
databrew	databrew:BatchDeleteRecipeVersion databrew:CreateDataset databrew:CreateProfileJob databrew:CreateProject databrew:CreateRecipe databrew:CreateRecipeJob databrew:CreateRuleset databrew:CreateSchedule databrew:DeleteDataset databrew:DeleteJob databrew:DeleteProject databrew:DeleteRecipeVersion databrew:DeleteRuleset databrew:DeleteSchedule databrew:DescribeDataset databrew:DescribeJob databrew:DescribeJobRun databrew:DescribeProject databrew:DescribeRecipe databrew:DescribeRuleset databrew:DescribeSchedule

서비스 접두사	작업
	databrew:ListDatasets databrew:ListJobRuns databrew:ListJobs databrew:ListProjects databrew:ListRecipes databrew:ListRecipeVersions databrew:ListRulesets databrew:ListSchedules databrew:PublishRecipe databrew:SendProjectSessionAction databrew:StartJobRun databrew:StartProjectSession databrew:StopJobRun databrew:UpdateDataset databrew:UpdateProfileJob databrew:UpdateProject databrew:UpdateRecipe databrew:UpdateRecipeJob databrew:UpdateRuleset databrew:UpdateSchedule

서비스 접두사	작업
dataexchange	dataexchange:CancelJob
	dataexchange:CreateDataSet
	dataexchange:CreateEventAction
	dataexchange:CreateJob
	dataexchange:CreateRevision
	dataexchange:DeleteAsset
	dataexchange:DeleteEventAction
	dataexchange:DeleteRevision
	dataexchange:GetEventAction
	dataexchange:GetJob
	dataexchange:ListDataSetRevisions
	dataexchange:ListDataSets
	dataexchange:ListEventActions
	dataexchange:ListJobs
	dataexchange:ListRevisionAssets
	dataexchange:RevokeRevision
	dataexchange:SendDataSetNotification
	dataexchange:StartJob
	dataexchange:UpdateAsset
	dataexchange:UpdateDataSet
	dataexchange:UpdateEventAction

서비스 접두사	작업
	dataexchange:UpdateRevision
datapipeline	datapipeline:ActivatePipeline datapipeline:CreatePipeline datapipeline:DeactivatePipeline datapipeline:DeletePipeline datapipeline:DescribeObjects datapipeline:DescribePipelines datapipeline:EvaluateExpression datapipeline:GetPipelineDefinition datapipeline:ListPipelines datapipeline:PollForTask datapipeline:PutPipelineDefinition datapipeline:QueryObjects datapipeline:ReportTaskProgress datapipeline:ReportTaskRunnerHeartbeat datapipeline:SetStatus datapipeline:SetTaskStatus datapipeline:ValidatePipelineDefinition

서비스 접두사	작업
dax	dax:CreateCluster
	dax:DecreaseReplicationFactor
	dax:DeleteCluster
	dax:DeleteParameterGroup
	dax:DeleteSubnetGroup
	dax:DescribeClusters
	dax:DescribeDefaultParameters
	dax:DescribeEvents
	dax:DescribeParameterGroups
	dax:DescribeParameters
	dax:DescribeSubnetGroups
	dax:IncreaseReplicationFactor
	dax:RebootNode
	dax:UpdateCluster
	dax:UpdateParameterGroup
	dax:UpdateSubnetGroup

서비스 접두사	작업
devicefarm	devicefarm:CreateDevicePool devicefarm:CreateInstanceProfile devicefarm:CreateNetworkProfile devicefarm:CreateProject devicefarm:CreateRemoteAccessSession devicefarm:CreateTestGridProject devicefarm:CreateTestGridUrl devicefarm:CreateUpload devicefarm:CreateVPCEConfiguration devicefarm:DeleteDevicePool devicefarm:DeleteInstanceProfile devicefarm:DeleteNetworkProfile devicefarm:DeleteProject devicefarm:DeleteRemoteAccessSession devicefarm:DeleteRun devicefarm:DeleteTestGridProject devicefarm:DeleteUpload devicefarm:DeleteVPCEConfiguration devicefarm:GetAccountSettings devicefarm:GetDevice devicefarm:GetDeviceInstance

서비스 접두사	작업
	devicefarm:GetDevicePool devicefarm:GetDevicePoolCompatibility devicefarm:GetInstanceProfile devicefarm:GetJob devicefarm:GetNetworkProfile devicefarm:GetOfferingStatus devicefarm:GetProject devicefarm:GetRemoteAccessSession devicefarm:GetRun devicefarm:GetSuite devicefarm:GetTest devicefarm:GetTestGridProject devicefarm:GetTestGridSession devicefarm:GetUpload devicefarm:GetVPCEConfiguration devicefarm:ListArtifacts devicefarm:ListDeviceInstances devicefarm:ListDevicePools devicefarm:ListDevices devicefarm:ListInstanceProfiles devicefarm:ListJobs

서비스 접두사	작업
	devicefarm:ListNetworkProfiles devicefarm:ListOfferingPromotions devicefarm:ListOfferings devicefarm:ListOfferingTransactions devicefarm:ListProjects devicefarm:ListRemoteAccessSessions devicefarm:ListRuns devicefarm:ListSamples devicefarm:ListSuites devicefarm:ListTestGridProjects devicefarm:ListTestGridSessionActions devicefarm:ListTestGridSessionArtifacts devicefarm:ListTestGridSessions devicefarm:ListTests devicefarm:ListUniqueProblems devicefarm:ListUploads devicefarm:ListVPCEConfigurations devicefarm:PurchaseOffering devicefarm:RenewOffering devicefarm:ScheduleRun devicefarm:StopJob

서비스 접두사	작업
	devicefarm:StopRemoteAccessSession devicefarm:StopRun devicefarm:UpdateDeviceInstance devicefarm:UpdateDevicePool devicefarm:UpdateInstanceProfile devicefarm:UpdateNetworkProfile devicefarm:UpdateProject devicefarm:UpdateTestGridProject devicefarm:UpdateUpload devicefarm:UpdateVPCEConfiguration

서비스 접두사	작업
devops-guru	devops-guru:AddNotificationChannel
	devops-guru:DeleteInsight
	devops-guru:DescribeAccountHealth
	devops-guru:DescribeAccountOverview
	devops-guru:DescribeAnomaly
	devops-guru:DescribeEventSourcesConfig
	devops-guru:DescribeFeedback
	devops-guru:DescribeInsight
	devops-guru:DescribeOrganizationHealth
	devops-guru:DescribeOrganizationOverview
	devops-guru:DescribeOrganizationResourceCollectionHealth
	devops-guru:DescribeResourceCollectionHealth
	devops-guru:DescribeServiceIntegration
	devops-guru:GetCostEstimation
	devops-guru:GetResourceCollection
	devops-guru:ListAnomaliesForInsight
	devops-guru:ListAnomalousLogGroups
	devops-guru:ListEvents
	devops-guru:ListInsights
	devops-guru:ListMonitoredResources
	devops-guru:ListNotificationChannels

서비스 접두사	작업
	devops-guru:ListOrganizationInsights
	devops-guru:ListRecommendations
	devops-guru:PutFeedback
	devops-guru:RemoveNotificationChannel
	devops-guru:SearchInsights
	devops-guru:SearchOrganizationInsights
	devops-guru:StartCostEstimation
	devops-guru:UpdateEventSourcesConfig
	devops-guru:UpdateResourceCollection
	devops-guru:UpdateServiceIntegration

서비스 접두사	작업
directconnect	directconnect:AcceptDirectConnectGatewayAssociationProposal directconnect:AllocateConnectionOnInterconnect directconnect:AllocateHostedConnection directconnect:AllocatePrivateVirtualInterface directconnect:AllocatePublicVirtualInterface directconnect:AllocateTransitVirtualInterface directconnect:AssociateConnectionWithLag directconnect:AssociateHostedConnection directconnect:AssociateMacSecKey directconnect:AssociateVirtualInterface directconnect:ConfirmConnection directconnect:ConfirmCustomerAgreement directconnect:ConfirmPrivateVirtualInterface directconnect:ConfirmPublicVirtualInterface directconnect:ConfirmTransitVirtualInterface directconnect:CreateBGPPeer directconnect:CreateConnection directconnect:CreateDirectConnectGateway directconnect:CreateDirectConnectGatewayAssociation directconnect:CreateDirectConnectGatewayAssociationProposal directconnect:CreateInterconnect

서비스 접두사	작업
	directconnect:CreateLag directconnect:CreatePrivateVirtualInterface directconnect:CreatePublicVirtualInterface directconnect:CreateTransitVirtualInterface directconnect:DeleteBGPPeer directconnect:DeleteConnection directconnect:DeleteDirectConnectGateway directconnect:DeleteDirectConnectGatewayAssociation directconnect:DeleteDirectConnectGatewayAssociationProposal directconnect:DeleteInterconnect directconnect:DeleteLag directconnect:DeleteVirtualInterface directconnect:DescribeConnectionLoa directconnect:DescribeConnections directconnect:DescribeConnectionsOnInterconnect directconnect:DescribeCustomerMetadata directconnect:DescribeDirectConnectGatewayAssociationProposals directconnect:DescribeDirectConnectGatewayAssociations directconnect:DescribeDirectConnectGatewayAttachments directconnect:DescribeDirectConnectGateways directconnect:DescribeHostedConnections

서비스 접두사	작업
	directconnect:DescribeInterconnectLoa directconnect:DescribeInterconnects directconnect:DescribeLags directconnect:DescribeLoa directconnect:DescribeLocations directconnect:DescribeRouterConfiguration directconnect:DescribeVirtualGateways directconnect:DescribeVirtualInterfaces directconnect:DisassociateConnectionFromLag directconnect:DisassociateMacSecKey directconnect:ListVirtualInterfaceTestHistory directconnect:StartBgpFailoverTest directconnect:StopBgpFailoverTest directconnect:UpdateConnection directconnect:UpdateDirectConnectGateway directconnect:UpdateDirectConnectGatewayAssociation directconnect:UpdateLag directconnect:UpdateVirtualInterfaceAttributes

서비스 접두사	작업
dlm	dlm:CreateLifecyclePolicy
	dlm:DeleteLifecyclePolicy
	dlm:GetLifecyclePolicies
	dlm:GetLifecyclePolicy
	dlm:UpdateLifecyclePolicy

서비스 접두사	작업
DMS	dms:ApplyPendingMaintenanceAction dms:BatchStartRecommendations dms:CancelReplicationTaskAssessmentRun dms:CreateDataProvider dms:CreateEndpoint dms:CreateEventSubscription dms:CreateInstanceProfile dms:CreateMigrationProject dms:CreateReplicationConfig dms:CreateReplicationInstance dms:CreateReplicationSubnetGroup dms:CreateReplicationTask dms:DeleteCertificate dms:DeleteConnection dms:DeleteDataProvider dms:DeleteEndpoint dms:DeleteEventSubscription dms:DeleteFleetAdvisorCollector dms:DeleteFleetAdvisorDatabases dms:DeleteInstanceProfile dms:DeleteMigrationProject

서비스 접두사	작업
	dms:DeleteReplicationConfig dms:DeleteReplicationInstance dms:DeleteReplicationSubnetGroup dms:DeleteReplicationTask dms:DeleteReplicationTaskAssessmentRun dms:DescribeAccountAttributes dms:DescribeApplicableIndividualAssessments dms:DescribeCertificates dms:DescribeConnections dms:DescribeEndpoints dms:DescribeEndpointSettings dms:DescribeEndpointTypes dms:DescribeEngineVersions dms:DescribeEventCategories dms:DescribeEvents dms:DescribeEventSubscriptions dms:DescribeFleetAdvisorCollectors dms:DescribeFleetAdvisorDatabases dms:DescribeFleetAdvisorLsaAnalysis dms:DescribeFleetAdvisorSchemaObjectSummary dms:DescribeFleetAdvisorSchemas

서비스 접두사	작업
	dms:DescribeMetadataModelImports dms:DescribeOrderableReplicationInstances dms:DescribePendingMaintenanceActions dms:DescribeRecommendationLimitations dms:DescribeRecommendations dms:DescribeRefreshSchemasStatus dms:DescribeReplicationConfigs dms:DescribeReplicationInstances dms:DescribeReplicationInstanceTaskLogs dms:DescribeReplications dms:DescribeReplicationSubnetGroups dms:DescribeReplicationTableStatistics dms:DescribeReplicationTaskAssessmentResults dms:DescribeReplicationTaskAssessmentRuns dms:DescribeReplicationTaskIndividualAssessments dms:DescribeReplicationTasks dms:DescribeSchemas dms:DescribeTableStatistics dms:ExportMetadataModelAssessment dms:GetMetadataModel dms:ImportCertificate

서비스 접두사	작업
	dms:ListMetadataModelAssessmentActionItems dms:ModifyEndpoint dms:ModifyEventSubscription dms:ModifyReplicationConfig dms:ModifyReplicationInstance dms:ModifyReplicationSubnetGroup dms:ModifyReplicationTask dms:MoveReplicationTask dms:RebootReplicationInstance dms:RefreshSchemas dms:ReloadReplicationTables dms:ReloadTables dms:RunFleetAdvisorLsaAnalysis dms:StartMetadataModelAssessment dms:StartMetadataModelConversion dms:StartMetadataModelExportToTarget dms:StartRecommendations dms:StartReplication dms:StartReplicationTask dms:StartReplicationTaskAssessment dms:StopReplicationTask

서비스 접두사	작업
	dms:TestConnection dms:UpdateSubscriptionsToEventBridge
docdb-elastic	docdb-elastic:CopyClusterSnapshot docdb-elastic:DeleteCluster docdb-elastic:DeleteClusterSnapshot docdb-elastic:GetCluster docdb-elastic:GetClusterSnapshot docdb-elastic:ListClusters docdb-elastic:ListClusterSnapshots docdb-elastic:RestoreClusterFromSnapshot docdb-elastic:StartCluster docdb-elastic:StopCluster docdb-elastic:UpdateCluster

서비스 접두사	작업
ds	ds:AcceptSharedDirectory ds:AddIpRoutes ds:AddRegion ds:CancelSchemaExtension ds:ConnectDirectory ds:CreateAlias ds:CreateComputer ds:CreateConditionalForwarder ds:CreateDirectory ds:CreateLogSubscription ds:CreateMicrosoftAD ds:CreateSnapshot ds:CreateTrust ds:DeleteConditionalForwarder ds:DeleteDirectory ds:DeleteLogSubscription ds:DeleteSnapshot ds:DeleteTrust ds:DeregisterCertificate ds:DeregisterEventTopic ds:DescribeCertificate

서비스 접두사	작업
	ds:DescribeClientAuthenticationSettings ds:DescribeConditionalForwarders ds:DescribeDirectories ds:DescribeDomainControllers ds:DescribeEventTopics ds:DescribeLDAPSSettings ds:DescribeRegions ds:DescribeSettings ds:DescribeSharedDirectories ds:DescribeSnapshots ds:DescribeTrusts ds:DescribeUpdateDirectory ds:DisableClientAuthentication ds:DisableLDAPS ds:DisableRadius ds:DisableSso ds:EnableClientAuthentication ds:EnableLDAPS ds:EnableRadius ds:EnableSso ds:GetDirectoryLimits

서비스 접두사	작업
	ds:GetSnapshotLimits ds:ListCertificates ds:ListIpRoutes ds:ListLogSubscriptions ds:ListSchemaExtensions ds:RegisterCertificate ds:RegisterEventTopic ds:RejectSharedDirectory ds:RemoveIpRoutes ds:RemoveRegion ds:ResetUserPassword ds:RestoreFromSnapshot ds:ShareDirectory ds:StartSchemaExtension ds:UnshareDirectory ds:UpdateConditionalForwarder ds:UpdateDirectorySetup ds:UpdateNumberOfDomainControllers ds:UpdateRadius ds:UpdateSettings ds:UpdateTrust

서비스 접두사	작업
	ds:VerifyTrust

서비스 접두사	작업
dynamodb	dynamodb:CreateBackup
	dynamodb:CreateGlobalTable
	dynamodb:CreateTable
	dynamodb:DeleteBackup
	dynamodb:DeleteTable
	dynamodb:DescribeBackup
	dynamodb:DescribeContinuousBackups
	dynamodb:DescribeContributorInsights
	dynamodb:DescribeEndpoints
	dynamodb:DescribeExport
	dynamodb:DescribeGlobalTable
	dynamodb:DescribeGlobalTableSettings
	dynamodb:DescribeImport
	dynamodb:DescribeKinesisStreamingDestination
	dynamodb:DescribeLimits
	dynamodb:DescribeStream
	dynamodb:DescribeTable
	dynamodb:DescribeTableReplicaAutoScaling
	dynamodb:DescribeTimeToLive
	dynamodb:DisableKinesisStreamingDestination
	dynamodb:EnableKinesisStreamingDestination

서비스 접두사	작업
	dynamodb:ExportTableToPointInTime
	dynamodb:GetResourcePolicy
	dynamodb:ImportTable
	dynamodb:ListBackups
	dynamodb:ListContributorInsights
	dynamodb:ListExports
	dynamodb:ListGlobalTables
	dynamodb:ListImports
	dynamodb:ListStreams
	dynamodb:ListTables
	dynamodb:RestoreTableFromBackup
	dynamodb:RestoreTableToPointInTime
	dynamodb:UpdateContinuousBackups
	dynamodb:UpdateContributorInsights
	dynamodb:UpdateGlobalTable
	dynamodb:UpdateGlobalTableSettings
	dynamodb:UpdateKinesisStreamingDestination
	dynamodb:UpdateTable
	dynamodb:UpdateTableReplicaAutoScaling
	dynamodb:UpdateTimeToLive

서비스 접두사	작업
ebs	ebs:CompleteSnapshot ebs:StartSnapshot

서비스 접두사	작업
EC2	ec2:AcceptAddressTransfer
	ec2:AcceptReservedInstancesExchangeQuote
	ec2:AcceptTransitGatewayMulticastDomainAssociations
	ec2:AcceptTransitGatewayPeeringAttachment
	ec2:AcceptTransitGatewayVpcAttachment
	ec2:AcceptVpcEndpointConnections
	ec2:AcceptVpcPeeringConnection
	ec2:AdvertiseByoipCidr
	ec2:AllocateAddress
	ec2:AllocateHosts
	ec2:AllocateIpamPoolCidr
	ec2:ApplySecurityGroupsToClientVpnTargetNetwork
	ec2:AssignIpv6Addresses
	ec2:AssignPrivateIpAddresses
	ec2:AssignPrivateNatGatewayAddress
	ec2:AssociateAddress
	ec2:AssociateClientVpnTargetNetwork
	ec2:AssociateDhcpOptions
	ec2:AssociateEnclaveCertificateIamRole
	ec2:AssociateIamInstanceProfile
	ec2:AssociateInstanceEventWindow

서비스 접두사	작업
	ec2:AssociateIpamByoasn
	ec2:AssociateIpamResourceDiscovery
	ec2:AssociateNatGatewayAddress
	ec2:AssociateRouteTable
	ec2:AssociateSubnetCidrBlock
	ec2:AssociateTransitGatewayMulticastDomain
	ec2:AssociateTransitGatewayPolicyTable
	ec2:AssociateTransitGatewayRouteTable
	ec2:AssociateTrunkInterface
	ec2:AssociateVpcCidrBlock
	ec2:AttachClassicLinkVpc
	ec2:AttachInternetGateway
	ec2:AttachNetworkInterface
	ec2:AttachVerifiedAccessTrustProvider
	ec2:AttachVolume
	ec2:AttachVpnGateway
	ec2:AuthorizeClientVpnIngress
	ec2:AuthorizeSecurityGroupEgress
	ec2:AuthorizeSecurityGroupIngress
	ec2:BundleInstance
	ec2:CancelBundleTask

서비스 접두사	작업
	ec2:CancelCapacityReservation
	ec2:CancelCapacityReservationFleets
	ec2:CancelConversionTask
	ec2:CancelExportTask
	ec2:CancelImageLaunchPermission
	ec2:CancelImportTask
	ec2:CancelReservedInstancesListing
	ec2:CancelSpotFleetRequests
	ec2:CancelSpotInstanceRequests
	ec2:ConfirmProductInstance
	ec2:CopyFpgaImage
	ec2:CopyImage
	ec2:CopySnapshot
	ec2:CreateCapacityReservation
	ec2:CreateCapacityReservationFleet
	ec2:CreateCarrierGateway
	ec2:CreateClientVpnEndpoint
	ec2:CreateClientVpnRoute
	ec2:CreateCoipCidr
	ec2:CreateCoipPool
	ec2:CreateCustomerGateway

서비스 접두사	작업
	ec2:CreateDefaultSubnet
	ec2:CreateDefaultVpc
	ec2:CreateDhcpOptions
	ec2:CreateEgressOnlyInternetGateway
	ec2:CreateFleet
	ec2:CreateFlowLogs
	ec2:CreateFpgaImage
	ec2:CreateImage
	ec2:CreateInstanceConnectEndpoint
	ec2:CreateInstanceEventWindow
	ec2:CreateInstanceExportTask
	ec2:CreateInternetGateway
	ec2:CreateIpam
	ec2:CreateIpamPool
	ec2:CreateIpamResourceDiscovery
	ec2:CreateIpamScope
	ec2:CreateKeyPair
	ec2:CreateLaunchTemplate
	ec2:CreateLaunchTemplateVersion
	ec2:CreateLocalGatewayRoute
	ec2:CreateLocalGatewayRouteTable

서비스 접두사	작업
	ec2:CreateLocalGatewayRouteTableVirtualInterfaceGroupAssociation
	ec2:CreateLocalGatewayRouteTableVpcAssociation
	ec2:CreateManagedPrefixList
	ec2:CreateNatGateway
	ec2:CreateNetworkAcl
	ec2:CreateNetworkAclEntry
	ec2:CreateNetworkInsightsAccessScope
	ec2:CreateNetworkInsightsPath
	ec2:CreateNetworkInterface
	ec2:CreateNetworkInterfacePermission
	ec2:CreatePlacementGroup
	ec2:CreatePublicIpv4Pool
	ec2:CreateReplaceRootVolumeTask
	ec2:CreateReservedInstancesListing
	ec2:CreateRestoreImageTask
	ec2:CreateRoute
	ec2:CreateRouteTable
	ec2:CreateSecurityGroup
	ec2:CreateSnapshot
	ec2:CreateSnapshots

서비스 접두사	작업
	ec2:CreateSpotDatafeedSubscription
	ec2:CreateStoreImageTask
	ec2:CreateSubnet
	ec2:CreateSubnetCidrReservation
	ec2:CreateTrafficMirrorFilter
	ec2:CreateTrafficMirrorFilterRule
	ec2:CreateTrafficMirrorSession
	ec2:CreateTrafficMirrorTarget
	ec2:CreateTransitGateway
	ec2:CreateTransitGatewayConnect
	ec2:CreateTransitGatewayConnectPeer
	ec2:CreateTransitGatewayMulticastDomain
	ec2:CreateTransitGatewayPeeringAttachment
	ec2:CreateTransitGatewayPolicyTable
	ec2:CreateTransitGatewayPrefixListReference
	ec2:CreateTransitGatewayRoute
	ec2:CreateTransitGatewayRouteTable
	ec2:CreateTransitGatewayRouteTableAnnouncement
	ec2:CreateTransitGatewayVpcAttachment
	ec2:CreateVerifiedAccessEndpoint
	ec2:CreateVerifiedAccessGroup

서비스 접두사	작업
	ec2:CreateVerifiedAccessInstance
	ec2:CreateVerifiedAccessTrustProvider
	ec2:CreateVolume
	ec2:CreateVpc
	ec2:CreateVpcEndpoint
	ec2:CreateVpcEndpointConnectionNotification
	ec2:CreateVpcEndpointServiceConfiguration
	ec2:CreateVpcPeeringConnection
	ec2:CreateVpnConnection
	ec2:CreateVpnConnectionRoute
	ec2:CreateVpnGateway
	ec2:DeleteCarrierGateway
	ec2:DeleteClientVpnEndpoint
	ec2:DeleteClientVpnRoute
	ec2:DeleteCoipCidr
	ec2:DeleteCoipPool
	ec2:DeleteCustomerGateway
	ec2:DeleteDhcpOptions
	ec2:DeleteEgressOnlyInternetGateway
	ec2:DeleteFleets
	ec2:DeleteFlowLogs

서비스 접두사	작업
	ec2:DeleteFpgaImage
	ec2:DeleteInstanceConnectEndpoint
	ec2:DeleteInstanceEventWindow
	ec2:DeleteInternetGateway
	ec2:DeleteIpam
	ec2:DeleteIpamPool
	ec2:DeleteIpamResourceDiscovery
	ec2:DeleteIpamScope
	ec2:DeleteKeyPair
	ec2:DeleteLaunchTemplate
	ec2:DeleteLaunchTemplateVersions
	ec2:DeleteLocalGatewayRoute
	ec2:DeleteLocalGatewayRouteTable
	ec2:DeleteLocalGatewayRouteTableVirtualInterfaceGroupAssociation
	ec2:DeleteLocalGatewayRouteTableVpcAssociation
	ec2:DeleteManagedPrefixList
	ec2:DeleteNatGateway
	ec2:DeleteNetworkAcl
	ec2:DeleteNetworkAclEntry
	ec2:DeleteNetworkInsightsAccessScope

서비스 접두사	작업
	ec2:DeleteNetworkInsightsAccessScopeAnalysis
	ec2:DeleteNetworkInsightsAnalysis
	ec2:DeleteNetworkInsightsPath
	ec2:DeleteNetworkInterface
	ec2:DeleteNetworkInterfacePermission
	ec2:DeletePlacementGroup
	ec2:DeletePublicIpv4Pool
	ec2:DeleteQueuedReservedInstances
	ec2:DeleteRoute
	ec2:DeleteRouteTable
	ec2:DeleteSecurityGroup
	ec2:DeleteSnapshot
	ec2:DeleteSpotDatafeedSubscription
	ec2:DeleteSubnet
	ec2:DeleteSubnetCidrReservation
	ec2:DeleteTrafficMirrorFilter
	ec2:DeleteTrafficMirrorFilterRule
	ec2:DeleteTrafficMirrorSession
	ec2:DeleteTrafficMirrorTarget
	ec2:DeleteTransitGateway
	ec2:DeleteTransitGatewayConnect

서비스 접두사	작업
	ec2:DeleteTransitGatewayConnectPeer
	ec2:DeleteTransitGatewayMulticastDomain
	ec2:DeleteTransitGatewayPeeringAttachment
	ec2:DeleteTransitGatewayPolicyTable
	ec2:DeleteTransitGatewayPrefixListReference
	ec2:DeleteTransitGatewayRoute
	ec2:DeleteTransitGatewayRouteTable
	ec2:DeleteTransitGatewayRouteTableAnnouncement
	ec2:DeleteTransitGatewayVpcAttachment
	ec2:DeleteVerifiedAccessEndpoint
	ec2:DeleteVerifiedAccessGroup
	ec2:DeleteVerifiedAccessInstance
	ec2:DeleteVerifiedAccessTrustProvider
	ec2:DeleteVolume
	ec2:DeleteVpc
	ec2:DeleteVpcEndpointConnectionNotifications
	ec2:DeleteVpcEndpoints
	ec2:DeleteVpcEndpointServiceConfigurations
	ec2:DeleteVpcPeeringConnection
	ec2:DeleteVpnConnection
	ec2:DeleteVpnConnectionRoute

서비스 접두사	작업
	ec2:DeleteVpnGateway
	ec2:DeprovisionByoipCidr
	ec2:DeprovisionIpamByoasn
	ec2:DeprovisionIpamPoolCidr
	ec2:DeprovisionPublicIpv4PoolCidr
	ec2:DeregisterImage
	ec2:DeregisterInstanceEventNotificationAttributes
	ec2:DeregisterTransitGatewayMulticastGroupMembers
	ec2:DeregisterTransitGatewayMulticastGroupSources
	ec2:DescribeAccountAttributes
	ec2:DescribeAddresses
	ec2:DescribeAddressesAttribute
	ec2:DescribeAddressTransfers
	ec2:DescribeAggregateIdFormat
	ec2:DescribeAvailabilityZones
	ec2:DescribeAwsNetworkPerformanceMetricSubscriptions
	ec2:DescribeBundleTasks
	ec2:DescribeByoipCidrs
	ec2:DescribeCapacityReservationFleets
	ec2:DescribeCapacityReservations
	ec2:DescribeCarrierGateways

서비스 접두사	작업
	ec2:DescribeClassicLinkInstances
	ec2:DescribeClientVpnAuthorizationRules
	ec2:DescribeClientVpnConnections
	ec2:DescribeClientVpnEndpoints
	ec2:DescribeClientVpnRoutes
	ec2:DescribeClientVpnTargetNetworks
	ec2:DescribeCoipPools
	ec2:DescribeConversionTasks
	ec2:DescribeCustomerGateways
	ec2:DescribeDhcpOptions
	ec2:DescribeEgressOnlyInternetGateways
	ec2:DescribeElasticGpus
	ec2:DescribeExportImageTasks
	ec2:DescribeExportTasks
	ec2:DescribeFastLaunchImages
	ec2:DescribeFastSnapshotRestores
	ec2:DescribeFleetHistory
	ec2:DescribeFleetInstances
	ec2:DescribeFleets
	ec2:DescribeFlowLogs
	ec2:DescribeFpgaImageAttribute

서비스 접두사	작업
	ec2:DescribeFpgaImages
	ec2:DescribeHostReservationOfferings
	ec2:DescribeHostReservations
	ec2:DescribeHosts
	ec2:DescribeIamInstanceProfileAssociations
	ec2:DescribeIdentityIdFormat
	ec2:DescribeIdFormat
	ec2:DescribeImageAttribute
	ec2:DescribeImages
	ec2:DescribeImportImageTasks
	ec2:DescribeImportSnapshotTasks
	ec2:DescribeInstanceAttribute
	ec2:DescribeInstanceConnectEndpoints
	ec2:DescribeInstanceCreditSpecifications
	ec2:DescribeInstanceEventNotificationAttributes
	ec2:DescribeInstanceEventWindows
	ec2:DescribeInstances
	ec2:DescribeInstanceStatus
	ec2:DescribeInstanceTopology
	ec2:DescribeInstanceTypeOfferings
	ec2:DescribeInstanceTypes

서비스 접두사	작업
	ec2:DescribeInternetGateways
	ec2:DescribeIpamByoasn
	ec2:DescribeIpamPools
	ec2:DescribeIpamResourceDiscoveries
	ec2:DescribeIpamResourceDiscoveryAssociations
	ec2:DescribeIpams
	ec2:DescribeIpamScopes
	ec2:DescribeIpv6Pools
	ec2:DescribeKeyPairs
	ec2:DescribeLaunchTemplates
	ec2:DescribeLaunchTemplateVersions
	ec2:DescribeLocalGatewayRouteTables
	ec2:DescribeLocalGatewayRouteTableVirtualInterfaceGroupAssociations
	ec2:DescribeLocalGatewayRouteTableVpcAssociations
	ec2:DescribeLocalGateways
	ec2:DescribeLocalGatewayVirtualInterfaceGroups
	ec2:DescribeLocalGatewayVirtualInterfaces
	ec2:DescribeLockedSnapshots
	ec2:DescribeMacHosts
	ec2:DescribeManagedPrefixLists

서비스 접두사	작업
	ec2:DescribeMovingAddresses
	ec2:DescribeNatGateways
	ec2:DescribeNetworkAcls
	ec2:DescribeNetworkInsightsAccessScopeAnalyses
	ec2:DescribeNetworkInsightsAccessScopes
	ec2:DescribeNetworkInsightsAnalyses
	ec2:DescribeNetworkInsightsPaths
	ec2:DescribeNetworkInterfaceAttribute
	ec2:DescribeNetworkInterfacePermissions
	ec2:DescribeNetworkInterfaces
	ec2:DescribePlacementGroups
	ec2:DescribePrefixLists
	ec2:DescribePrincipalIdFormat
	ec2:DescribePublicIpv4Pools
	ec2:DescribeRegions
	ec2:DescribeReplaceRootVolumeTasks
	ec2:DescribeReservedInstances
	ec2:DescribeReservedInstancesListings
	ec2:DescribeReservedInstancesModifications
	ec2:DescribeReservedInstancesOfferings
	ec2:DescribeRouteTables

서비스 접두사	작업
	ec2:DescribeScheduledInstanceAvailability
	ec2:DescribeScheduledInstances
	ec2:DescribeSecurityGroupReferences
	ec2:DescribeSecurityGroupRules
	ec2:DescribeSecurityGroups
	ec2:DescribeSnapshotAttribute
	ec2:DescribeSnapshots
	ec2:DescribeSnapshotTierStatus
	ec2:DescribeSpotDatafeedSubscription
	ec2:DescribeSpotFleetInstances
	ec2:DescribeSpotFleetRequestHistory
	ec2:DescribeSpotFleetRequests
	ec2:DescribeSpotInstanceRequests
	ec2:DescribeSpotPriceHistory
	ec2:DescribeStaleSecurityGroups
	ec2:DescribeStoreImageTasks
	ec2:DescribeSubnets
	ec2:DescribeTrafficMirrorFilterRules
	ec2:DescribeTrafficMirrorFilters
	ec2:DescribeTrafficMirrorSessions
	ec2:DescribeTrafficMirrorTargets

서비스 접두사	작업
	ec2:DescribeTransitGatewayAttachments
	ec2:DescribeTransitGatewayConnectPeers
	ec2:DescribeTransitGatewayConnects
	ec2:DescribeTransitGatewayMulticastDomains
	ec2:DescribeTransitGatewayPeeringAttachments
	ec2:DescribeTransitGatewayPolicyTables
	ec2:DescribeTransitGatewayRouteTableAnnouncements
	ec2:DescribeTransitGatewayRouteTables
	ec2:DescribeTransitGateways
	ec2:DescribeTransitGatewayVpcAttachments
	ec2:DescribeTrunkInterfaceAssociations
	ec2:DescribeVerifiedAccessEndpoints
	ec2:DescribeVerifiedAccessGroups
	ec2:DescribeVerifiedAccessInstanceLoggingConfigurations
	ec2:DescribeVerifiedAccessInstances
	ec2:DescribeVerifiedAccessTrustProviders
	ec2:DescribeVolumeAttribute
	ec2:DescribeVolumes
	ec2:DescribeVolumesModifications
	ec2:DescribeVolumeStatus
	ec2:DescribeVpcAttribute

서비스 접두사	작업
	ec2:DescribeVpcClassicLink
	ec2:DescribeVpcClassicLinkDnsSupport
	ec2:DescribeVpcEndpointConnectionNotifications
	ec2:DescribeVpcEndpointConnections
	ec2:DescribeVpcEndpoints
	ec2:DescribeVpcEndpointServiceConfigurations
	ec2:DescribeVpcEndpointServicePermissions
	ec2:DescribeVpcEndpointServices
	ec2:DescribeVpcPeeringConnections
	ec2:DescribeVpcs
	ec2:DescribeVpnConnections
	ec2:DescribeVpnGateways
	ec2:DetachClassicLinkVpc
	ec2:DetachInternetGateway
	ec2:DeleteNetworkInterface
	ec2:DetachVerifiedAccessTrustProvider
	ec2:DetachVolume
	ec2:DetachVpnGateway
	ec2:DisableAddressTransfer
	ec2:DisableAwsNetworkPerformanceMetricSubscription
	ec2:DisableEbsEncryptionByDefault

서비스 접두사	작업
	ec2:DisableFastLaunch
	ec2:DisableFastSnapshotRestores
	ec2:DisableImage
	ec2:DisableImageBlockPublicAccess
	ec2:DisableImageDeprecation
	ec2:DisableImageDeregistrationProtection
	ec2:DisableIpamOrganizationAdminAccount
	ec2:DisableSerialConsoleAccess
	ec2:DisableSnapshotBlockPublicAccess
	ec2:DisableTransitGatewayRouteTablePropagation
	ec2:DisableVgwRoutePropagation
	ec2:DisableVpcClassicLink
	ec2:DisableVpcClassicLinkDnsSupport
	ec2:DisassociateAddress
	ec2:DisassociateClientVpnTargetNetwork
	ec2:DisassociateEnclaveCertificateIamRole
	ec2:DisassociateIamInstanceProfile
	ec2:DisassociateInstanceEventWindow
	ec2:DisassociateIpamByoasn
	ec2:DisassociateIpamResourceDiscovery
	ec2:DisassociateNatGatewayAddress

서비스 접두사	작업
	ec2:DisassociateRouteTable
	ec2:DisassociateSubnetCidrBlock
	ec2:DisassociateTransitGatewayMulticastDomain
	ec2:DisassociateTransitGatewayPolicyTable
	ec2:DisassociateTransitGatewayRouteTable
	ec2:DisassociateTrunkInterface
	ec2:DisassociateVpcCidrBlock
	ec2:EnableAddressTransfer
	ec2:EnableAwsNetworkPerformanceMetricSubscription
	ec2:EnableEbsEncryptionByDefault
	ec2:EnableFastLaunch
	ec2:EnableFastSnapshotRestores
	ec2:EnableImage
	ec2:EnableImageBlockPublicAccess
	ec2:EnableImageDeprecation
	ec2:EnableImageDeregistrationProtection
	ec2:EnableIpamOrganizationAdminAccount
	ec2:EnableReachabilityAnalyzerOrganizationSharing
	ec2:EnableSerialConsoleAccess
	ec2:EnableSnapshotBlockPublicAccess
	ec2:EnableTransitGatewayRouteTablePropagation

서비스 접두사	작업
	ec2:EnableVgwRoutePropagation
	ec2:EnableVolumeIO
	ec2:EnableVpcClassicLink
	ec2:EnableVpcClassicLinkDnsSupport
	ec2:ExportClientVpnClientCertificateRevocationList
	ec2:ExportClientVpnClientConfiguration
	ec2:ExportImage
	ec2:ExportTransitGatewayRoutes
	ec2:GetAssociatedEnclaveCertificateIamRoles
	ec2:GetAssociatedIpv6PoolCidrs
	ec2:GetAwsNetworkPerformanceData
	ec2:GetCapacityReservationUsage
	ec2:GetCoipPoolUsage
	ec2:GetConsoleOutput
	ec2:GetConsoleScreenshot
	ec2:GetDefaultCreditSpecification
	ec2:GetEbsDefaultKmsKeyId
	ec2:GetEbsEncryptionByDefault
	ec2:GetFlowLogsIntegrationTemplate
	ec2:GetGroupsForCapacityReservation
	ec2:GetHostReservationPurchasePreview

서비스 접두사	작업
	ec2:GetImageBlockPublicAccessState
	ec2:GetInstanceMetadataDefaults
	ec2:GetInstanceTpmEkPub
	ec2:GetInstanceTypesFromInstanceRequirements
	ec2:GetInstanceUefiData
	ec2:GetIpamAddressHistory
	ec2:GetIpamDiscoveredAccounts
	ec2:GetIpamDiscoveredPublicAddresses
	ec2:GetIpamDiscoveredResourceCidrs
	ec2:GetIpamPoolAllocations
	ec2:GetIpamPoolCidrs
	ec2:GetIpamResourceCidrs
	ec2:GetLaunchTemplateData
	ec2:GetManagedPrefixListAssociations
	ec2:GetManagedPrefixListEntries
	ec2:GetNetworkInsightsAccessScopeAnalysisFindings
	ec2:GetNetworkInsightsAccessScopeContent
	ec2:GetPasswordData
	ec2:GetReservedInstancesExchangeQuote
	ec2:GetSecurityGroupsForVpc
	ec2:GetSerialConsoleAccessStatus

서비스 접두사	작업
	ec2:GetSnapshotBlockPublicAccessState
	ec2:GetSpotPlacementScores
	ec2:GetSubnetCidrReservations
	ec2:GetTransitGatewayAttachmentPropagations
	ec2:GetTransitGatewayMulticastDomainAssociations
	ec2:GetTransitGatewayPolicyTableAssociations
	ec2:GetTransitGatewayPolicyTableEntries
	ec2:GetTransitGatewayPrefixListReferences
	ec2:GetTransitGatewayRouteTableAssociations
	ec2:GetTransitGatewayRouteTablePropagations
	ec2:GetVerifiedAccessEndpointPolicy
	ec2:GetVerifiedAccessGroupPolicy
	ec2:GetVpnConnectionDeviceSampleConfiguration
	ec2:GetVpnConnectionDeviceTypes
	ec2:GetVpnTunnelReplacementStatus
	ec2:ImportClientVpnClientCertificateRevocationList
	ec2:ImportImage
	ec2:ImportInstance
	ec2:ImportKeyPair
	ec2:ImportSnapshot
	ec2:ImportVolume

서비스 접두사	작업
	ec2:ListImagesInRecycleBin
	ec2:ListSnapshotsInRecycleBin
	ec2:LockSnapshot
	ec2:ModifyAddressAttribute
	ec2:ModifyAvailabilityZoneGroup
	ec2:ModifyCapacityReservation
	ec2:ModifyCapacityReservationFleet
	ec2:ModifyClientVpnEndpoint
	ec2:ModifyDefaultCreditSpecification
	ec2:ModifyEbsDefaultKmsKeyId
	ec2:ModifyFleet
	ec2:ModifyFpgaImageAttribute
	ec2:ModifyHosts
	ec2:ModifyIdentityIdFormat
	ec2:ModifyIdFormat
	ec2:ModifyImageAttribute
	ec2:ModifyInstanceAttribute
	ec2:ModifyInstanceCapacityReservationAttributes
	ec2:ModifyInstanceCreditSpecification
	ec2:ModifyInstanceEventStartTime
	ec2:ModifyInstanceEventWindow

서비스 접두사	작업
	ec2:ModifyInstanceMaintenanceOptions
	ec2:ModifyInstanceMetadataDefaults
	ec2:ModifyInstanceMetadataOptions
	ec2:ModifyInstancePlacement
	ec2:ModifyIpam
	ec2:ModifyIpamPool
	ec2:ModifyIpamResourceCidr
	ec2:ModifyIpamResourceDiscovery
	ec2:ModifyIpamScope
	ec2:ModifyLaunchTemplate
	ec2:ModifyLocalGatewayRoute
	ec2:ModifyManagedPrefixList
	ec2:ModifyNetworkInterfaceAttribute
	ec2:ModifyPrivateDnsNameOptions
	ec2:ModifyReservedInstances
	ec2:ModifySecurityGroupRules
	ec2:ModifySnapshotAttribute
	ec2:ModifySnapshotTier
	ec2:ModifySpotFleetRequest
	ec2:ModifySubnetAttribute
	ec2:ModifyTrafficMirrorFilterNetworkServices

서비스 접두사	작업
	ec2:ModifyTrafficMirrorFilterRule
	ec2:ModifyTrafficMirrorSession
	ec2:ModifyTransitGateway
	ec2:ModifyTransitGatewayPrefixListReference
	ec2:ModifyTransitGatewayVpcAttachment
	ec2:ModifyVerifiedAccessEndpoint
	ec2:ModifyVerifiedAccessEndpointPolicy
	ec2:ModifyVerifiedAccessGroup
	ec2:ModifyVerifiedAccessGroupPolicy
	ec2:ModifyVerifiedAccessInstance
	ec2:ModifyVerifiedAccessInstanceLoggingConfiguration
	ec2:ModifyVerifiedAccessTrustProvider
	ec2:ModifyVolume
	ec2:ModifyVolumeAttribute
	ec2:ModifyVpcAttribute
	ec2:ModifyVpcEndpoint
	ec2:ModifyVpcEndpointConnectionNotification
	ec2:ModifyVpcEndpointServiceConfiguration
	ec2:ModifyVpcEndpointServicePayerResponsibility
	ec2:ModifyVpcEndpointServicePermissions
	ec2:ModifyVpcPeeringConnectionOptions

서비스 접두사	작업
	ec2:ModifyVpcTenancy
	ec2:ModifyVpnConnection
	ec2:ModifyVpnConnectionOptions
	ec2:ModifyVpnTunnelCertificate
	ec2:ModifyVpnTunnelOptions
	ec2:MonitorInstances
	ec2:MoveAddressToVpc
	ec2:MoveByoipCidrToIpam
	ec2:ProvisionByoipCidr
	ec2:ProvisionIpamByoasn
	ec2:ProvisionIpamPoolCidr
	ec2:ProvisionPublicIpv4PoolCidr
	ec2:PurchaseHostReservation
	ec2:PurchaseReservedInstancesOffering
	ec2:PurchaseScheduledInstances
	ec2:RebootInstances
	ec2:RegisterImage
	ec2:RegisterInstanceEventNotificationAttributes
	ec2:RegisterTransitGatewayMulticastGroupMembers
	ec2:RegisterTransitGatewayMulticastGroupSources
	ec2:RejectTransitGatewayMulticastDomainAssociations

서비스 접두사	작업
	ec2:RejectTransitGatewayPeeringAttachment
	ec2:RejectTransitGatewayVpcAttachment
	ec2:RejectVpcEndpointConnections
	ec2:RejectVpcPeeringConnection
	ec2:ReleaseAddress
	ec2:ReleaseHosts
	ec2:ReleaseIpamPoolAllocation
	ec2:ReplaceIamInstanceProfileAssociation
	ec2:ReplaceNetworkAclAssociation
	ec2:ReplaceNetworkAclEntry
	ec2:ReplaceRoute
	ec2:ReplaceRouteTableAssociation
	ec2:ReplaceTransitGatewayRoute
	ec2:ReplaceVpnTunnel
	ec2:ReportInstanceStatus
	ec2:RequestSpotFleet
	ec2:RequestSpotInstances
	ec2:ResetAddressAttribute
	ec2:ResetEbsDefaultKmsKeyId
	ec2:ResetFpgaImageAttribute
	ec2:ResetImageAttribute

서비스 접두사	작업
	ec2:ResetInstanceAttribute
	ec2:ResetNetworkInterfaceAttribute
	ec2:ResetSnapshotAttribute
	ec2:RestoreAddressToClassic
	ec2:RestoreImageFromRecycleBin
	ec2:RestoreManagedPrefixListVersion
	ec2:RestoreSnapshotFromRecycleBin
	ec2:RestoreSnapshotTier
	ec2:RevokeClientVpnIngress
	ec2:RevokeSecurityGroupEgress
	ec2:RevokeSecurityGroupIngress
	ec2:RunInstances
	ec2:RunScheduledInstances
	ec2:SearchLocalGatewayRoutes
	ec2:SearchTransitGatewayMulticastGroups
	ec2:SearchTransitGatewayRoutes
	ec2:SendDiagnosticInterrupt
	ec2:StartInstances
	ec2:StartNetworkInsightsAccessScopeAnalysis
	ec2:StartNetworkInsightsAnalysis
	ec2:StartVpcEndpointServicePrivateDnsVerification

서비스 접두사	작업
	ec2:StopInstances ec2:TerminateClientVpnConnections ec2:TerminateInstances ec2:UnassignIpv6Addresses ec2:UnassignPrivateIpAddresses ec2:UnassignPrivateNatGatewayAddress ec2:UnlockSnapshot ec2:UnmonitorInstances ec2:UpdateSecurityGroupRuleDescriptionsEgress ec2:UpdateSecurityGroupRuleDescriptionsIngress ec2:WithdrawByoipCidr

서비스 접두사	작업
ecr	ecr:BatchCheckLayerAvailability ecr:BatchDeleteImage ecr:BatchGetImage ecr:BatchGetRepositoryScanningConfiguration ecr:CompleteLayerUpload ecr:CreatePullThroughCacheRule ecr:CreateRepository ecr:CreateRepositoryCreationTemplate ecr:DeleteLifecyclePolicy ecr:DeletePullThroughCacheRule ecr:DeleteRegistryPolicy ecr:DeleteRepository ecr:DeleteRepositoryCreationTemplate ecr:DeleteRepositoryPolicy ecr:DescribeImageReplicationStatus ecr:DescribeImages ecr:DescribeImageScanFindings ecr:DescribePullThroughCacheRules ecr:DescribeRegistry ecr:DescribeRepositories ecr:GetAuthorizationToken

서비스 접두사	작업
	ecr:GetDownloadUrlForLayer ecr:GetLifecyclePolicy ecr:GetLifecyclePolicyPreview ecr:GetRegistryPolicy ecr:GetRegistryScanningConfiguration ecr:GetRepositoryPolicy ecr:InitiateLayerUpload ecr:ListImages ecr:PutImage ecr:PutImageScanningConfiguration ecr:PutRegistryPolicy ecr:PutRegistryScanningConfiguration ecr:PutReplicationConfiguration ecr:StartImageScan ecr:StartLifecyclePolicyPreview ecr:UpdatePullThroughCacheRule ecr:UploadLayerPart ecr:ValidatePullThroughCacheRule

서비스 접두사	작업
ecr-public	ecr-public:BatchCheckLayerAvailability
	ecr-public:BatchDeleteImage
	ecr-public:CompleteLayerUpload
	ecr-public:CreateRepository
	ecr-public:DeleteRepository
	ecr-public:DeleteRepositoryPolicy
	ecr-public:DescribeImages
	ecr-public:DescribeRegistries
	ecr-public:DescribeRepositories
	ecr-public:GetAuthorizationToken
	ecr-public:GetRegistryCatalogData
	ecr-public:GetRepositoryCatalogData
	ecr-public:GetRepositoryPolicy
	ecr-public:InitiateLayerUpload
	ecr-public:PutImage
	ecr-public:PutRegistryCatalogData
	ecr-public:PutRepositoryCatalogData
	ecr-public:SetRepositoryPolicy
	ecr-public:UploadLayerPart

서비스 접두사	작업
ecs	ecs:CreateCapacityProvider ecs:CreateCluster ecs:CreateService ecs:CreateTaskSet ecs:DeleteAccountSetting ecs:DeleteAttributes ecs:DeleteCapacityProvider ecs:DeleteCluster ecs:DeleteService ecs:DeleteTaskDefinitions ecs:DeleteTaskSet ecs:DeregisterContainerInstance ecs:DeregisterTaskDefinition ecs:DescribeCapacityProviders ecs:DescribeClusters ecs:DescribeContainerInstances ecs:DescribeServices ecs:DescribeTaskDefinition ecs:DescribeTasks ecs:DescribeTaskSets ecs:DiscoverPollEndpoint

서비스 접두사	작업
	ecs:ExecuteCommand ecs:GetTaskProtection ecs:ListAccountSettings ecs:ListAttributes ecs:ListClusters ecs:ListContainerInstances ecs:ListServices ecs:ListServicesByNamespace ecs:ListTaskDefinitionFamilies ecs:ListTaskDefinitions ecs:ListTasks ecs:PutAccountSetting ecs:PutAccountSettingDefault ecs:PutAttributes ecs:PutClusterCapacityProviders ecs:RegisterContainerInstance ecs:RegisterTaskDefinition ecs:RunTask ecs:StartTask ecs:StopTask ecs:SubmitAttachmentStateChanges

서비스 접두사	작업
	ecs:SubmitContainerStateChange ecs:SubmitTaskStateChange ecs:UpdateCapacityProvider ecs:UpdateCluster ecs:UpdateClusterSettings ecs:UpdateContainerAgent ecs:UpdateContainerInstancesState ecs:UpdateService ecs:UpdateServicePrimaryTaskSet ecs:UpdateTaskProtection ecs:UpdateTaskSet

서비스 접두사	작업
eks	eks:AssociateAccessPolicy
	eks:AssociateEncryptionConfig
	eks:AssociateIdentityProviderConfig
	eks:CreateAccessEntry
	eks:CreateAddon
	eks:CreateCluster
	eks:CreateEksAnywhereSubscription
	eks:CreateFargateProfile
	eks:CreateNodegroup
	eks:DeleteAccessEntry
	eks:DeleteAddon
	eks:DeleteCluster
	eks:DeleteEksAnywhereSubscription
	eks:DeleteFargateProfile
	eks:DeleteNodegroup
	eks:DeletePodIdentityAssociation
	eks:DeregisterCluster
	eks:DescribeAccessEntry
	eks:DescribeAddon
	eks:DescribeAddonConfiguration
	eks:DescribeAddonVersions

서비스 접두사	작업
	eks:DescribeCluster
	eks:DescribeEksAnywhereSubscription
	eks:DescribeFargateProfile
	eks:DescribeIdentityProviderConfig
	eks:DescribeInsight
	eks:DescribeNodegroup
	eks:DescribePodIdentityAssociation
	eks:DescribeUpdate
	eks:DisassociateAccessPolicy
	eks:DisassociateIdentityProviderConfig
	eks:ListAccessEntries
	eks:ListAccessPolicies
	eks:ListAddons
	eks:ListAssociatedAccessPolicies
	eks:ListClusters
	eks:ListEksAnywhereSubscriptions
	eks:ListFargateProfiles
	eks:ListIdentityProviderConfigs
	eks:ListInsights
	eks:ListNodegroups
	eks:ListPodIdentityAssociations

서비스 접두사	작업
	eks:ListUpdates eks:RegisterCluster eks:UpdateAccessEntry eks:UpdateAddon eks:UpdateClusterConfig eks:UpdateClusterVersion eks:UpdateEksAnywhereSubscription eks:UpdateNodegroupConfig eks:UpdateNodegroupVersion eks:UpdatePodIdentityAssociation
elastic-inference	elastic-inference:DescribeAcceleratorOfferings elastic-inference:DescribeAccelerators elastic-inference:DescribeAcceleratorTypes

서비스 접두사	작업
elasticache	elasticache:AuthorizeCacheSecurityGroupIngress
	elasticache:BatchApplyUpdateAction
	elasticache:BatchStopUpdateAction
	elasticache:CompleteMigration
	elasticache:CopyServerlessCacheSnapshot
	elasticache:CopySnapshot
	elasticache:CreateCacheCluster
	elasticache:CreateCacheParameterGroup
	elasticache:CreateCacheSecurityGroup
	elasticache:CreateCacheSubnetGroup
	elasticache:CreateGlobalReplicationGroup
	elasticache:CreateReplicationGroup
	elasticache:CreateServerlessCache
	elasticache:CreateServerlessCacheSnapshot
	elasticache:CreateSnapshot
	elasticache:CreateUser
	elasticache:CreateUserGroup
	elasticache:DecreaseNodeGroupsInGlobalReplicationGroup
	elasticache:DecreaseReplicaCount
	elasticache:DeleteCacheCluster
	elasticache:DeleteCacheParameterGroup

서비스 접두사	작업
	elasticache:DeleteCacheSecurityGroup
	elasticache:DeleteCacheSubnetGroup
	elasticache:DeleteGlobalReplicationGroup
	elasticache:DeleteReplicationGroup
	elasticache:DeleteServerlessCache
	elasticache:DeleteServerlessCacheSnapshot
	elasticache:DeleteSnapshot
	elasticache:DeleteUser
	elasticache:DeleteUserGroup
	elasticache:DescribeCacheClusters
	elasticache:DescribeCacheEngineVersions
	elasticache:DescribeCacheParameterGroups
	elasticache:DescribeCacheParameters
	elasticache:DescribeCacheSecurityGroups
	elasticache:DescribeCacheSubnetGroups
	elasticache:DescribeEngineDefaultParameters
	elasticache:DescribeEvents
	elasticache:DescribeGlobalReplicationGroups
	elasticache:DescribeReplicationGroups
	elasticache:DescribeReservedCacheNodes
	elasticache:DescribeReservedCacheNodesOfferings

서비스 접두사	작업
	elasticache:DescribeServerlessCaches
	elasticache:DescribeServerlessCacheSnapshots
	elasticache:DescribeServiceUpdates
	elasticache:DescribeSnapshots
	elasticache:DescribeUpdateActions
	elasticache:DescribeUserGroups
	elasticache:DescribeUsers
	elasticache:DisassociateGlobalReplicationGroup
	elasticache:ExportServerlessCacheSnapshot
	elasticache:FailoverGlobalReplicationGroup
	elasticache:IncreaseNodeGroupsInGlobalReplicationGroup
	elasticache:IncreaseReplicaCount
	elasticache:ListAllowedNodeTypeModifications
	elasticache:ModifyCacheCluster
	elasticache:ModifyCacheParameterGroup
	elasticache:ModifyCacheSubnetGroup
	elasticache:ModifyGlobalReplicationGroup
	elasticache:ModifyReplicationGroup
	elasticache:ModifyReplicationGroupShardConfiguration
	elasticache:ModifyServerlessCache
	elasticache:ModifyUser

서비스 접두사	작업
	elasticache:ModifyUserGroup elasticache:PurchaseReservedCacheNodesOffering elasticache:RebalanceSlotsInGlobalReplicationGroup elasticache:RebootCacheCluster elasticache:ResetCacheParameterGroup elasticache:RevokeCacheSecurityGroupIngress elasticache:StartMigration elasticache:TestFailover elasticache:TestMigration

서비스 접두사	작업
elasticbeanstalk	elasticbeanstalk:AbortEnvironmentUpdate elasticbeanstalk:ApplyEnvironmentManagedAction elasticbeanstalk:AssociateEnvironmentOperationsRole elasticbeanstalk:CheckDNSAvailability elasticbeanstalk:ComposeEnvironments elasticbeanstalk:CreateApplication elasticbeanstalk:CreateApplicationVersion elasticbeanstalk:CreateConfigurationTemplate elasticbeanstalk:CreateEnvironment elasticbeanstalk:CreatePlatformVersion elasticbeanstalk:CreateStorageLocation elasticbeanstalk:DeleteApplication elasticbeanstalk:DeleteApplicationVersion elasticbeanstalk:DeleteConfigurationTemplate elasticbeanstalk:DeleteEnvironmentConfiguration elasticbeanstalk:DeletePlatformVersion elasticbeanstalk:DescribeAccountAttributes elasticbeanstalk:DescribeApplications elasticbeanstalk:DescribeApplicationVersions elasticbeanstalk:DescribeConfigurationOptions elasticbeanstalk:DescribeConfigurationSettings

서비스 접두사	작업
	elasticbeanstalk:DescribeEnvironmentHealth elasticbeanstalk:DescribeEnvironmentManagedActionHistory elasticbeanstalk:DescribeEnvironmentManagedActions elasticbeanstalk:DescribeEnvironmentResources elasticbeanstalk:DescribeEnvironments elasticbeanstalk:DescribeEvents elasticbeanstalk:DescribeInstancesHealth elasticbeanstalk:DescribePlatformVersion elasticbeanstalk:DisassociateEnvironmentOperationsRole elasticbeanstalk:ListAvailableSolutionStacks elasticbeanstalk:ListPlatformBranches elasticbeanstalk:ListPlatformVersions elasticbeanstalk:RebuildEnvironment elasticbeanstalk:RequestEnvironmentInfo elasticbeanstalk:RestartAppServer elasticbeanstalk:RetrieveEnvironmentInfo elasticbeanstalk:SwapEnvironmentCNAMEs elasticbeanstalk:TerminateEnvironment elasticbeanstalk:UpdateApplication elasticbeanstalk:UpdateApplicationResourceLifecycle elasticbeanstalk:UpdateApplicationVersion

서비스 접두사	작업
	elasticbeanstalk:UpdateConfigurationTemplate elasticbeanstalk:UpdateEnvironment elasticbeanstalk:ValidateConfigurationSettings

서비스 접두사	작업
elasticfilesystem	elasticfilesystem:CreateAccessPoint
	elasticfilesystem:CreateFileSystem
	elasticfilesystem:CreateMountTarget
	elasticfilesystem:CreateReplicationConfiguration
	elasticfilesystem:DeleteAccessPoint
	elasticfilesystem:DeleteFileSystem
	elasticfilesystem:DeleteFileSystemPolicy
	elasticfilesystem:DeleteMountTarget
	elasticfilesystem:DeleteReplicationConfiguration
	elasticfilesystem:DescribeAccessPoints
	elasticfilesystem:DescribeAccountPreferences
	elasticfilesystem:DescribeBackupPolicy
	elasticfilesystem:DescribeFileSystemPolicy
	elasticfilesystem:DescribeFileSystems
	elasticfilesystem:DescribeLifecycleConfiguration
	elasticfilesystem:DescribeMountTargets
	elasticfilesystem:DescribeMountTargetSecurityGroups
	elasticfilesystem:DescribeReplicationConfigurations
	elasticfilesystem:ModifyMountTargetSecurityGroups
	elasticfilesystem:PutAccountPreferences
	elasticfilesystem:PutBackupPolicy

서비스 접두사	작업
	elasticfilesystem:PutFileSystemPolicy elasticfilesystem:PutLifecycleConfiguration elasticfilesystem:UpdateFileSystem elasticfilesystem:UpdateFileSystemProtection

서비스 접두사	작업
elasticloadbalancing	elasticloadbalancing:AddListenerCertificates elasticloadbalancing:AddTrustStoreRevocations elasticloadbalancing:ApplySecurityGroupsToLoadBalancer elasticloadbalancing:AttachLoadBalancerToSubnets elasticloadbalancing:ConfigureHealthCheck elasticloadbalancing:CreateAppCookieStickinessPolicy elasticloadbalancing:CreateLBCookieStickinessPolicy elasticloadbalancing:CreateListener elasticloadbalancing:CreateLoadBalancer elasticloadbalancing:CreateLoadBalancerListeners elasticloadbalancing:CreateLoadBalancerPolicy elasticloadbalancing:CreateRule elasticloadbalancing:CreateTargetGroup elasticloadbalancing:CreateTrustStore elasticloadbalancing:DeleteListener elasticloadbalancing:DeleteLoadBalancer elasticloadbalancing:DeleteLoadBalancerListeners elasticloadbalancing:DeleteLoadBalancerPolicy elasticloadbalancing:DeleteRule elasticloadbalancing:DeleteTargetGroup elasticloadbalancing:DeleteTrustStore

서비스 접두사	작업
	elasticloadbalancing:DeregisterInstancesFromLoadBalancer elasticloadbalancing:DeregisterTargets elasticloadbalancing:DescribeAccountLimits elasticloadbalancing:DescribeInstanceHealth elasticloadbalancing:DescribeListenerCertificates elasticloadbalancing:DescribeListeners elasticloadbalancing:DescribeLoadBalancerAttributes elasticloadbalancing:DescribeLoadBalancerPolicies elasticloadbalancing:DescribeLoadBalancerPolicyTypes elasticloadbalancing:DescribeLoadBalancers elasticloadbalancing:DescribeRules elasticloadbalancing:DescribeSSLPolicies elasticloadbalancing:DescribeTargetGroupAttributes elasticloadbalancing:DescribeTargetGroups elasticloadbalancing:DescribeTargetHealth elasticloadbalancing:DescribeTrustStoreAssociations elasticloadbalancing:DescribeTrustStoreRevocations elasticloadbalancing:DescribeTrustStores elasticloadbalancing:DetachLoadBalancerFromSubnets elasticloadbalancing:DisableAvailabilityZonesForLoadBalancer elasticloadbalancing:EnableAvailabilityZonesForLoadBalancer

서비스 접두사	작업
	elasticloadbalancing:GetTrustStoreCaCertificatesBundle elasticloadbalancing:GetTrustStoreRevocationContent elasticloadbalancing:ModifyListener elasticloadbalancing:ModifyLoadBalancerAttributes elasticloadbalancing:ModifyRule elasticloadbalancing:ModifyTargetGroup elasticloadbalancing:ModifyTargetGroupAttributes elasticloadbalancing:ModifyTrustStore elasticloadbalancing:RegisterInstancesWithLoadBalancer elasticloadbalancing:RegisterTargets elasticloadbalancing:RemoveListenerCertificates elasticloadbalancing:RemoveTrustStoreRevocations elasticloadbalancing:SetIpAddressType elasticloadbalancing:SetLoadBalancerListenerSSLCertificate elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer elasticloadbalancing:SetLoadBalancerPoliciesOfListener elasticloadbalancing:SetRulePriorities elasticloadbalancing:SetSecurityGroups elasticloadbalancing:SetSubnets

서비스 접두사	작업
elastictranscoder	elastictranscoder:CancelJob
	elastictranscoder:CreateJob
	elastictranscoder:CreatePipeline
	elastictranscoder:CreatePreset
	elastictranscoder:DeletePipeline
	elastictranscoder:DeletePreset
	elastictranscoder:ListJobsByPipeline
	elastictranscoder:ListJobsByStatus
	elastictranscoder:ListPipelines
	elastictranscoder:ListPresets
	elastictranscoder:ReadJob
	elastictranscoder:ReadPipeline
	elastictranscoder:ReadPreset
	elastictranscoder:TestRole
	elastictranscoder:UpdatePipeline
	elastictranscoder:UpdatePipelineNotifications
	elastictranscoder:UpdatePipelineStatus

서비스 접두사	작업
emr-containers	emr-containers:CancelJobRun emr-containers:CreateJobTemplate emr-containers:CreateManagedEndpoint emr-containers:CreateSecurityConfiguration emr-containers:CreateVirtualCluster emr-containers:DeleteJobTemplate emr-containers:DeleteManagedEndpoint emr-containers:DeleteVirtualCluster emr-containers:DescribeJobRun emr-containers:DescribeJobTemplate emr-containers:DescribeManagedEndpoint emr-containers:DescribeSecurityConfiguration emr-containers:DescribeVirtualCluster emr-containers:GetManagedEndpointSessionCredentials emr-containers:ListJobRuns emr-containers:ListJobTemplates emr-containers:ListManagedEndpoints emr-containers:ListSecurityConfigurations emr-containers:ListVirtualClusters emr-containers:StartJobRun

서비스 접두사	작업
emr-serverless	emr-serverless:CancelJobRun
	emr-serverless:CreateApplication
	emr-serverless:DeleteApplication
	emr-serverless:GetApplication
	emr-serverless:GetDashboardForJobRun
	emr-serverless:GetJobRun
	emr-serverless:ListApplications
	emr-serverless:ListJobRunAttempts
	emr-serverless:ListJobRuns
	emr-serverless:StartApplication
	emr-serverless:StartJobRun
	emr-serverless:StopApplication
	emr-serverless:UpdateApplication

서비스 접두사	작업
es	es:AcceptInboundConnection es:AcceptInboundCrossClusterSearchConnection es:AssociatePackage es:AuthorizeVpcEndpointAccess es:CancelElasticsearchServiceSoftwareUpdate es:CancelServiceSoftwareUpdate es:CreateDomain es:CreateElasticsearchDomain es:CreateOutboundConnection es:CreateOutboundCrossClusterSearchConnection es:CreatePackage es:CreateVpcEndpoint es:DeleteDomain es:DeleteElasticsearchDomain es:DeleteElasticsearchServiceRole es:DeleteInboundConnection es:DeleteInboundCrossClusterSearchConnection es:DeleteOutboundConnection es:DeleteOutboundCrossClusterSearchConnection es:DeletePackage es:DeleteVpcEndpoint

서비스 접두사	작업
	es:DescribeDomain es:DescribeDomainAutoTunes es:DescribeDomainChangeProgress es:DescribeDomainConfig es:DescribeDomainHealth es:DescribeDomainNodes es:DescribeDomains es:DescribeDryRunProgress es:DescribeElasticsearchDomain es:DescribeElasticsearchDomainConfig es:DescribeElasticsearchDomains es:DescribeElasticsearchInstanceTypeLimits es:DescribeInboundConnections es:DescribeInboundCrossClusterSearchConnections es:DescribeInstanceTypeLimits es:DescribeOutboundConnections es:DescribeOutboundCrossClusterSearchConnections es:DescribePackages es:DescribeReservedElasticsearchInstanceOfferings es:DescribeReservedElasticsearchInstances es:DescribeReservedInstanceOfferings

서비스 접두사	작업
	es:DescribeReservedInstances es:DescribeVpcEndpoints es:DissociatePackage es:GetCompatibleElasticsearchVersions es:GetCompatibleVersions es:GetDataSource es:GetDomainMaintenanceStatus es:GetPackageVersionHistory es:GetUpgradeHistory es:GetUpgradeStatus es:ListDataSources es:ListDomainNames es:ListDomainsForPackage es:ListElasticsearchInstanceTypes es:ListElasticsearchVersions es:ListInstanceTypeDetails es:ListPackagesForDomain es:ListScheduledActions es:ListVersions es:ListVpcEndpointAccess es:ListVpcEndpoints

서비스 접두사	작업
	es:ListVpcEndpointsForDomain es:PurchaseReservedElasticsearchInstanceOffering es:PurchaseReservedInstanceOffering es:RejectInboundConnection es:RejectInboundCrossClusterSearchConnection es:RevokeVpcEndpointAccess es:StartDomainMaintenance es:StartElasticsearchServiceSoftwareUpdate es:StartServiceSoftwareUpdate es:UpdateDataSource es:UpdateDomainConfig es:UpdateElasticsearchDomainConfig es:UpdatePackage es:UpdateScheduledAction es:UpdateVpcEndpoint es:UpgradeDomain es:UpgradeElasticsearchDomain

서비스 접두사	작업
이벤트	events:ActivateEventSource
	events:CancelReplay
	events:CreateApiDestination
	events:CreateArchive
	events:CreateConnection
	events:CreateEndpoint
	events:CreateEventBus
	events:CreatePartnerEventSource
	events:DeactivateEventSource
	events:DeauthorizeConnection
	events:DeleteApiDestination
	events:DeleteArchive
	events:DeleteConnection
	events:DeleteEndpoint
	events:DeleteEventBus
	events:DeletePartnerEventSource
	events:DeleteRule
	events:DescribeApiDestination
	events:DescribeArchive
	events:DescribeConnection
	events:DescribeEndpoint

서비스 접두사	작업
	events:DescribeEventBus
	events:DescribeEventSource
	events:DescribePartnerEventSource
	events:DescribeReplay
	events:DescribeRule
	events:DisableRule
	events:EnableRule
	events:ListApiDestinations
	events:ListArchives
	events:ListConnections
	events:ListEndpoints
	events:ListEventBuses
	events:ListEventSources
	events:ListPartnerEventSourceAccounts
	events:ListPartnerEventSources
	events:ListReplays
	events:ListRuleNamesByTarget
	events:ListRules
	events:ListTargetsByRule
	events:PutPermission
	events:PutRule

서비스 접두사	작업
	events:PutTargets
	events:RemovePermission
	events:RemoveTargets
	events:StartReplay
	events:TestEventPattern
	events:UpdateApiDestination
	events:UpdateArchive
	events:UpdateConnection
	events:UpdateEndpoint

서비스 접두사	작업
evidently	evidently:CreateExperiment evidently:CreateFeature evidently:CreateLaunch evidently:CreateProject evidently:CreateSegment evidently:DeleteExperiment evidently:DeleteFeature evidently:DeleteLaunch evidently:DeleteProject evidently:DeleteSegment evidently:GetExperiment evidently:GetExperimentResults evidently:GetFeature evidently:GetLaunch evidently:GetProject evidently:GetSegment evidently:ListExperiments evidently:ListFeatures evidently:ListLaunches evidently:ListProjects evidently:ListSegmentReferences

서비스 접두사	작업
	evidently:ListSegments evidently:StartExperiment evidently:StartLaunch evidently:StopExperiment evidently:StopLaunch evidently:TestSegmentPattern evidently:UpdateExperiment evidently:UpdateFeature evidently:UpdateLaunch evidently:UpdateProject evidently:UpdateProjectDataDelivery

서비스 접두사	작업
finspace	finspace:CreateEnvironment
	finspace:CreateKxChangeset
	finspace:CreateKxCluster
	finspace:CreateKxDatabase
	finspace:CreateKxDataview
	finspace:CreateKxEnvironment
	finspace:CreateKxScalingGroup
	finspace:CreateKxUser
	finspace:CreateKxVolume
	finspace:CreateUser
	finspace:DeleteEnvironment
	finspace:DeleteKxCluster
	finspace:DeleteKxClusterNode
	finspace:DeleteKxDatabase
	finspace:DeleteKxDataview
	finspace:DeleteKxEnvironment
	finspace:DeleteKxScalingGroup
	finspace:DeleteKxUser
	finspace:DeleteKxVolume
	finspace:GetEnvironment
	finspace:GetKxChangeset

서비스 접두사	작업
	finspace:GetKxCluster
	finspace:GetKxConnectionString
	finspace:GetKxDatabase
	finspace:GetKxDataview
	finspace:GetKxEnvironment
	finspace:GetKxScalingGroup
	finspace:GetKxUser
	finspace:GetKxVolume
	finspace:GetLoadSampleDataSetGroupIntoEnvironmentStatus
	finspace:GetUser
	finspace:ListEnvironments
	finspace:ListKxChangesets
	finspace:ListKxClusterNodes
	finspace:ListKxClusters
	finspace:ListKxDatabases
	finspace:ListKxDataviews
	finspace:ListKxEnvironments
	finspace:ListKxScalingGroups
	finspace:ListKxUsers
	finspace:ListKxVolumes
	finspace:ListUsers

서비스 접두사	작업
	finspace:LoadSampleDataSetGroupIntoEnvironment finspace:ResetUserPassword finspace:UpdateEnvironment finspace:UpdateKxClusterCodeConfiguration finspace:UpdateKxClusterDatabases finspace:UpdateKxDatabase finspace:UpdateKxDataview finspace:UpdateKxEnvironment finspace:UpdateKxEnvironmentNetwork finspace:UpdateKxUser finspace:UpdateKxVolume finspace:UpdateUser
firehose	firehose:CreateDeliveryStream firehose:DeleteDeliveryStream firehose:DescribeDeliveryStream firehose:ListDeliveryStreams firehose:StartDeliveryStreamEncryption firehose:StopDeliveryStreamEncryption firehose:UpdateDestination

서비스 접두사	작업
fis	fis:CreateExperimentTemplate
	fis:CreateTargetAccountConfiguration
	fis:DeleteExperimentTemplate
	fis:DeleteTargetAccountConfiguration
	fis:GetAction
	fis:GetExperiment
	fis:GetExperimentTargetAccountConfiguration
	fis:GetExperimentTemplate
	fis:GetTargetAccountConfiguration
	fis:GetTargetResourceType
	fis:ListActions
	fis:ListExperimentResolvedTargets
	fis:ListExperiments
	fis:ListExperimentTargetAccountConfigurations
	fis:ListExperimentTemplates
	fis:ListTargetAccountConfigurations
	fis:ListTargetResourceTypes
	fis:StartExperiment
	fis:StopExperiment
	fis:UpdateExperimentTemplate
	fis:UpdateTargetAccountConfiguration

서비스 접두사	작업
fms	fms:AssociateAdminAccount fms:AssociateThirdPartyFirewall fms:BatchAssociateResource fms:BatchDisassociateResource fms:DeleteAppsList fms:DeleteNotificationChannel fms:DeletePolicy fms:DeleteProtocolsList fms:DeleteResourceSet fms:DisassociateAdminAccount fms:DisassociateThirdPartyFirewall fms:GetAdminAccount fms:GetAdminScope fms:GetAppsList fms:GetComplianceDetail fms:GetNotificationChannel fms:GetPolicy fms:GetProtectionStatus fms:GetProtocolsList fms:GetResourceSet fms:GetThirdPartyFirewallAssociationStatus

서비스 접두사	작업
	fms:GetViolationDetails fms:ListAdminAccountsForOrganization fms:ListAdminsManagingAccount fms:ListAppsLists fms:ListComplianceStatus fms:ListDiscoveredResources fms:ListMemberAccounts fms:ListPolicies fms:ListProtocolsLists fms:ListResourceSetResources fms:ListResourceSets fms:ListThirdPartyFirewallFirewallPolicies fms:PutAdminAccount fms:PutAppsList fms:PutNotificationChannel fms:PutPolicy fms:PutProtocolsList fms:PutResourceSet

서비스 접두사	작업
frauddetector	frauddetector:BatchCreateVariable
	frauddetector:BatchGetVariable
	frauddetector:CancelBatchImportJob
	frauddetector:CancelBatchPredictionJob
	frauddetector:CreateBatchImportJob
	frauddetector:CreateBatchPredictionJob
	frauddetector:CreateDetectorVersion
	frauddetector:CreateList
	frauddetector:CreateModel
	frauddetector:CreateModelVersion
	frauddetector:CreateRule
	frauddetector:CreateVariable
	frauddetector:DeleteBatchImportJob
	frauddetector:DeleteBatchPredictionJob
	frauddetector:DeleteDetector
	frauddetector:DeleteDetectorVersion
	frauddetector:DeleteEntityType
	frauddetector:DeleteEvent
	frauddetector:DeleteEventsByEventType
	frauddetector:DeleteEventType
	frauddetector:DeleteExternalModel

서비스 접두사	작업
	frauddetector:DeleteLabel
	frauddetector:DeleteList
	frauddetector:DeleteModel
	frauddetector:DeleteModelVersion
	frauddetector:DeleteOutcome
	frauddetector:DeleteRule
	frauddetector:DeleteVariable
	frauddetector:DescribeDetector
	frauddetector:DescribeModelVersions
	frauddetector:GetBatchImportJobs
	frauddetector:GetBatchPredictionJobs
	frauddetector:GetDeleteEventsByEventTypeStatus
	frauddetector:GetDetectors
	frauddetector:GetDetectorVersion
	frauddetector:GetEntityTypes
	frauddetector:GetEvent
	frauddetector:GetEventPrediction
	frauddetector:GetEventPredictionMetadata
	frauddetector:GetEventTypes
	frauddetector:GetExternalModels
	frauddetector:GetKMSEncryptionKey

서비스 접두사	작업
	frauddetector:GetLabels
	frauddetector:GetListElements
	frauddetector:GetListsMetadata
	frauddetector:GetModels
	frauddetector:GetModelVersion
	frauddetector:GetOutcomes
	frauddetector:GetRules
	frauddetector:GetVariables
	frauddetector:ListEventPredictions
	frauddetector:PutDetector
	frauddetector:PutEntityType
	frauddetector:PutEventType
	frauddetector:PutExternalModel
	frauddetector:PutKMSEncryptionKey
	frauddetector:PutLabel
	frauddetector:PutOutcome
	frauddetector:SendEvent
	frauddetector:UpdateDetectorVersion
	frauddetector:UpdateDetectorVersionMetadata
	frauddetector:UpdateDetectorVersionStatus
	frauddetector:UpdateEventLabel

서비스 접두사	작업
	frauddetector:UpdateList
	frauddetector:UpdateModel
	frauddetector:UpdateModelVersion
	frauddetector:UpdateModelVersionStatus
	frauddetector:UpdateRuleMetadata
	frauddetector:UpdateRuleVersion
	frauddetector:UpdateVariable

서비스 접두사	작업
fsx	fsx:AssociateFileSystemAliases fsx:CancelDataRepositoryTask fsx:CopyBackup fsx:CreateDataRepositoryTask fsx:CreateFileCache fsx:CreateFileSystem fsx:CreateFileSystemFromBackup fsx:CreateSnapshot fsx:CreateStorageVirtualMachine fsx:CreateVolume fsx:CreateVolumeFromBackup fsx:DeleteBackup fsx:DeleteFileCache fsx:DeleteFileSystem fsx:DeleteSnapshot fsx:DeleteStorageVirtualMachine fsx:DeleteVolume fsx:DescribeBackups fsx:DescribeDataRepositoryAssociations fsx:DescribeDataRepositoryTasks fsx:DescribeFileCaches

서비스 접두사	작업
	fsx:DescribeFileSystemAliases fsx:DescribeFileSystems fsx:DescribeSharedVpcConfiguration fsx:DescribeSnapshots fsx:DescribeStorageVirtualMachines fsx:DescribeVolumes fsx:DisassociateFileSystemAliases fsx:ReleaseFileSystemNfsV3Locks fsx:RestoreVolumeFromSnapshot fsx:StartMisconfiguredStateRecovery fsx:UpdateDataRepositoryAssociation fsx:UpdateFileCache fsx:UpdateFileSystem fsx:UpdateSharedVpcConfiguration fsx:UpdateSnapshot fsx:UpdateStorageVirtualMachine fsx:UpdateVolume

서비스 접두사	작업
gamelift	gamelift:AcceptMatch
	gamelift:ClaimGameServer
	gamelift:CreateAlias
	gamelift:CreateBuild
	gamelift:CreateContainerGroupDefinition
	gamelift:CreateFleet
	gamelift:CreateFleetLocations
	gamelift:CreateGameServerGroup
	gamelift:CreateGameSession
	gamelift:CreateGameSessionQueue
	gamelift:CreateLocation
	gamelift:CreateMatchmakingConfiguration
	gamelift:CreateMatchmakingRuleSet
	gamelift:CreatePlayerSession
	gamelift:CreatePlayerSessions
	gamelift:CreateScript
	gamelift:CreateVpcPeeringAuthorization
	gamelift:CreateVpcPeeringConnection
	gamelift:DeleteAlias
	gamelift:DeleteBuild
	gamelift:DeleteContainerGroupDefinition

서비스 접두사	작업
	gamelift:DeleteFleet
	gamelift:DeleteFleetLocations
	gamelift:DeleteGameServerGroup
	gamelift:DeleteGameSessionQueue
	gamelift:DeleteLocation
	gamelift:DeleteMatchmakingConfiguration
	gamelift:DeleteMatchmakingRuleSet
	gamelift:DeleteScalingPolicy
	gamelift:DeleteScript
	gamelift:DeleteVpcPeeringAuthorization
	gamelift:DeleteVpcPeeringConnection
	gamelift:DeregisterCompute
	gamelift:DeregisterGameServer
	gamelift:DescribeAlias
	gamelift:DescribeBuild
	gamelift:DescribeCompute
	gamelift:DescribeContainerGroupDefinition
	gamelift:DescribeEC2InstanceLimits
	gamelift:DescribeFleetAttributes
	gamelift:DescribeFleetCapacity
	gamelift:DescribeFleetEvents

서비스 접두사	작업
	gamelift:DescribeFleetLocationAttributes
	gamelift:DescribeFleetLocationCapacity
	gamelift:DescribeFleetLocationUtilization
	gamelift:DescribeFleetPortSettings
	gamelift:DescribeFleetUtilization
	gamelift:DescribeGameServer
	gamelift:DescribeGameServerGroup
	gamelift:DescribeGameServerInstances
	gamelift:DescribeGameSessionDetails
	gamelift:DescribeGameSessionPlacement
	gamelift:DescribeGameSessionQueues
	gamelift:DescribeGameSessions
	gamelift:DescribeInstances
	gamelift:DescribeMatchmaking
	gamelift:DescribeMatchmakingConfigurations
	gamelift:DescribeMatchmakingRuleSets
	gamelift:DescribePlayerSessions
	gamelift:DescribeRuntimeConfiguration
	gamelift:DescribeScalingPolicies
	gamelift:DescribeScript
	gamelift:DescribeVpcPeeringAuthorizations

서비스 접두사	작업
	gamelift:DescribeVpcPeeringConnections
	gamelift:GetComputeAccess
	gamelift:GetComputeAuthToken
	gamelift:GetGameSessionLogUrl
	gamelift:GetInstanceAccess
	gamelift:ListAliases
	gamelift:ListBuilds
	gamelift:ListCompute
	gamelift:ListContainerGroupDefinitions
	gamelift:ListFleets
	gamelift:ListGameServerGroups
	gamelift:ListGameServers
	gamelift:ListLocations
	gamelift:ListScripts
	gamelift:PutScalingPolicy
	gamelift:RegisterCompute
	gamelift:RegisterGameServer
	gamelift:RequestUploadCredentials
	gamelift:ResolveAlias
	gamelift:ResumeGameServerGroup
	gamelift:SearchGameSessions

서비스 접두사	작업
	gamelift:StartFleetActions
	gamelift:StartGameSessionPlacement
	gamelift:StartMatchBackfill
	gamelift:StartMatchmaking
	gamelift:StopFleetActions
	gamelift:StopGameSessionPlacement
	gamelift:StopMatchmaking
	gamelift:SuspendGameServerGroup
	gamelift:UpdateAlias
	gamelift:UpdateBuild
	gamelift:UpdateFleetAttributes
	gamelift:UpdateFleetCapacity
	gamelift:UpdateFleetPortSettings
	gamelift:UpdateGameServer
	gamelift:UpdateGameServerGroup
	gamelift:UpdateGameSession
	gamelift:UpdateGameSessionQueue
	gamelift:UpdateMatchmakingConfiguration
	gamelift:UpdateRuntimeConfiguration
	gamelift:UpdateScript
	gamelift:ValidateMatchmakingRuleSet

서비스 접두사	작업
geo	geo:AssociateTrackerConsumer geo:BatchDeleteDevicePositionHistory geo:BatchDeleteGeofence geo:BatchEvaluateGeofences geo:BatchGetDevicePosition geo:BatchPutGeofence geo:BatchUpdateDevicePosition geo:CalculateRoute geo:CalculateRouteMatrix geo:CreateGeofenceCollection geo:CreateMap geo:CreatePlaceIndex geo:CreateRouteCalculator geo:CreateTracker geo:DeleteGeofenceCollection geo:DeleteKey geo:DeleteMap geo:DeletePlaceIndex geo:DeleteRouteCalculator geo:DeleteTracker geo:DescribeGeofenceCollection

서비스 접두사	작업
	geo:DescribeKey geo:DescribeMap geo:DescribePlaceIndex geo:DescribeRouteCalculator geo:DescribeTracker geo:DisassociateTrackerConsumer geo:ForecastGeofenceEvents geo:GetDevicePosition geo:GetDevicePositionHistory geo:GetGeofence geo:GetMapGlyphs geo:GetMapSprites geo:GetMapStyleDescriptor geo:GetMapTile geo:GetPlace geo:ListDevicePositions geo:ListGeofenceCollections geo:ListGeofences geo:ListKeys geo:ListMaps geo:ListPlaceIndexes

서비스 접두사	작업
	geo:ListRouteCalculators geo:ListTrackerConsumers geo:ListTrackers geo:PutGeofence geo:SearchPlaceIndexForPosition geo:SearchPlaceIndexForSuggestions geo:SearchPlaceIndexForText geo:UpdateGeofenceCollection geo:UpdateKey geo:UpdateMap geo:UpdatePlaceIndex geo:UpdateRouteCalculator geo:UpdateTracker geo:VerifyDevicePosition

서비스 접두사	작업
glacier	glacier:AbortMultipartUpload
	glacier:AbortVaultLock
	glacier:CompleteMultipartUpload
	glacier:CompleteVaultLock
	glacier:CreateVault
	glacier:DeleteArchive
	glacier:DeleteVault
	glacier:DeleteVaultAccessPolicy
	glacier:DeleteVaultNotifications
	glacier:DescribeJob
	glacier:DescribeVault
	glacier:GetDataRetrievalPolicy
	glacier:GetJobOutput
	glacier:GetVaultAccessPolicy
	glacier:GetVaultLock
	glacier:GetVaultNotifications
	glacier:InitiateJob
	glacier:InitiateMultipartUpload
	glacier:InitiateVaultLock
	glacier:ListJobs
	glacier:ListMultipartUploads

서비스 접두사	작업
	glacier:ListParts
	glacier:ListProvisionedCapacity
	glacier:ListVaults
	glacier:PurchaseProvisionedCapacity
	glacier:SetDataRetrievalPolicy
	glacier:SetVaultAccessPolicy
	glacier:SetVaultNotifications
	glacier:UploadArchive
	glacier:UploadMultipartPart

서비스 접두사	작업
grafana	grafana:AssociateLicense grafana:CreateWorkspace grafana:CreateWorkspaceApiKey grafana:CreateWorkspaceServiceAccount grafana:CreateWorkspaceServiceAccountToken grafana:DeleteWorkspace grafana:DeleteWorkspaceApiKey grafana:DeleteWorkspaceServiceAccount grafana:DeleteWorkspaceServiceAccountToken grafana:DescribeWorkspace grafana:DescribeWorkspaceAuthentication grafana:DescribeWorkspaceConfiguration grafana:DisassociateLicense grafana:ListPermissions grafana:ListVersions grafana:ListWorkspaces grafana:ListWorkspaceServiceAccounts grafana:ListWorkspaceServiceAccountTokens grafana:UpdatePermissions grafana:UpdateWorkspace grafana:UpdateWorkspaceAuthentication

서비스 접두사	작업
	grafana:UpdateWorkspaceConfiguration

서비스 접두사	작업
greengrass	greengrass:AssociateRoleToGroup
	greengrass:AssociateServiceRoleToAccount
	greengrass:BatchAssociateClientDeviceWithCoreDevice
	greengrass:BatchDisassociateClientDeviceFromCoreDevice
	greengrass:CancelDeployment
	greengrass:CreateComponentVersion
	greengrass:CreateConnectorDefinition
	greengrass:CreateConnectorDefinitionVersion
	greengrass:CreateCoreDefinition
	greengrass:CreateCoreDefinitionVersion
	greengrass:CreateDeployment
	greengrass:CreateDeviceDefinition
	greengrass:CreateDeviceDefinitionVersion
	greengrass:CreateFunctionDefinition
	greengrass:CreateFunctionDefinitionVersion
	greengrass:CreateGroup
	greengrass:CreateGroupCertificateAuthority
	greengrass:CreateGroupVersion
	greengrass:CreateLoggerDefinition
	greengrass:CreateLoggerDefinitionVersion
	greengrass:CreateResourceDefinition

서비스 접두사	작업
	greengrass:CreateResourceDefinitionVersion
	greengrass:CreateSoftwareUpdateJob
	greengrass:CreateSubscriptionDefinition
	greengrass:CreateSubscriptionDefinitionVersion
	greengrass:DeleteComponent
	greengrass:DeleteConnectorDefinition
	greengrass:DeleteCoreDefinition
	greengrass:DeleteCoreDevice
	greengrass:DeleteDeployment
	greengrass:DeleteDeviceDefinition
	greengrass:DeleteFunctionDefinition
	greengrass:DeleteGroup
	greengrass:DeleteLoggerDefinition
	greengrass:DeleteResourceDefinition
	greengrass:DeleteSubscriptionDefinition
	greengrass:DescribeComponent
	greengrass:DisassociateRoleFromGroup
	greengrass:DisassociateServiceRoleFromAccount
	greengrass:GetAssociatedRole
	greengrass:GetBulkDeploymentStatus
	greengrass:GetComponent

서비스 접두사	작업
	greengrass:GetComponentVersionArtifact
	greengrass:GetConnectivityInfo
	greengrass:GetConnectorDefinition
	greengrass:GetConnectorDefinitionVersion
	greengrass:GetCoreDefinition
	greengrass:GetCoreDefinitionVersion
	greengrass:GetCoreDevice
	greengrass:GetDeployment
	greengrass:GetDeploymentStatus
	greengrass:GetDeviceDefinition
	greengrass:GetDeviceDefinitionVersion
	greengrass:GetFunctionDefinition
	greengrass:GetFunctionDefinitionVersion
	greengrass:GetGroup
	greengrass:GetGroupCertificateAuthority
	greengrass:GetGroupCertificateConfiguration
	greengrass:GetGroupVersion
	greengrass:GetLoggerDefinition
	greengrass:GetLoggerDefinitionVersion
	greengrass:GetResourceDefinition
	greengrass:GetResourceDefinitionVersion

서비스 접두사	작업
	greengrass:GetServiceRoleForAccount
	greengrass:GetSubscriptionDefinition
	greengrass:GetSubscriptionDefinitionVersion
	greengrass:GetThingRuntimeConfiguration
	greengrass:ListBulkDeploymentDetailedReports
	greengrass:ListBulkDeployments
	greengrass:ListClientDevicesAssociatedWithCoreDevice
	greengrass:ListComponents
	greengrass:ListComponentVersions
	greengrass:ListConnectorDefinitions
	greengrass:ListConnectorDefinitionVersions
	greengrass:ListCoreDefinitions
	greengrass:ListCoreDefinitionVersions
	greengrass:ListCoreDevices
	greengrass:ListDeployments
	greengrass:ListDeviceDefinitions
	greengrass:ListDeviceDefinitionVersions
	greengrass:ListEffectiveDeployments
	greengrass:ListFunctionDefinitions
	greengrass:ListFunctionDefinitionVersions
	greengrass:ListGroupCertificateAuthorities

서비스 접두사	작업
	greengrass:ListGroups
	greengrass:ListGroupVersions
	greengrass:ListInstalledComponents
	greengrass:ListLoggerDefinitions
	greengrass:ListLoggerDefinitionVersions
	greengrass:ListResourceDefinitions
	greengrass:ListResourceDefinitionVersions
	greengrass:ListSubscriptionDefinitions
	greengrass:ListSubscriptionDefinitionVersions
	greengrass:ResetDeployments
	greengrass:StartBulkDeployment
	greengrass:StopBulkDeployment
	greengrass:UpdateConnectivityInfo
	greengrass:UpdateConnectorDefinition
	greengrass:UpdateCoreDefinition
	greengrass:UpdateDeviceDefinition
	greengrass:UpdateFunctionDefinition
	greengrass:UpdateGroup
	greengrass:UpdateGroupCertificateConfiguration
	greengrass:UpdateLoggerDefinition
	greengrass:UpdateResourceDefinition

서비스 접두사	작업
	greengrass:UpdateSubscriptionDefinition greengrass:UpdateThingRuntimeConfiguration

서비스 접두사	작업
groundstation	groundstation:CancelContact groundstation:CreateConfig groundstation:CreateDataflowEndpointGroup groundstation:CreateEphemeris groundstation:CreateMissionProfile groundstation:DeleteConfig groundstation:DeleteDataflowEndpointGroup groundstation:DeleteEphemeris groundstation:DeleteMissionProfile groundstation:DescribeContact groundstation:DescribeEphemeris groundstation:GetConfig groundstation:GetDataflowEndpointGroup groundstation:GetMinuteUsage groundstation:GetMissionProfile groundstation:GetSatellite groundstation:ListConfigs groundstation:ListContacts groundstation:ListDataflowEndpointGroups groundstation:ListEphemerides groundstation:ListGroundStations

서비스 접두사	작업
	groundstation:ListMissionProfiles groundstation:ListSatellites groundstation:RegisterAgent groundstation:ReserveContact groundstation:UpdateAgentStatus groundstation:UpdateConfig groundstation:UpdateEphemeris groundstation:UpdateMissionProfile

서비스 접두사	작업
guardduty	guardduty:AcceptAdministratorInvitation
	guardduty:AcceptInvitation
	guardduty:ArchiveFindings
	guardduty:CreateDetector
	guardduty:CreateFilter
	guardduty:CreateIPSet
	guardduty:CreateMalwareProtectionPlan
	guardduty:CreateMembers
	guardduty:CreatePublishingDestination
	guardduty:CreateSampleFindings
	guardduty:CreateThreatIntelSet
	guardduty:DeclineInvitations
	guardduty:DeleteDetector
	guardduty:DeleteFilter
	guardduty:DeleteInvitations
	guardduty:DeleteIPSet
	guardduty:DeleteMalwareProtectionPlan
	guardduty:DeleteMembers
	guardduty:DeletePublishingDestination
	guardduty:DeleteThreatIntelSet
	guardduty:DescribeMalwareScans

서비스 접두사	작업
	guardduty:DescribeOrganizationConfiguration
	guardduty:DescribePublishingDestination
	guardduty:DisableOrganizationAdminAccount
	guardduty:DisassociateFromAdministratorAccount
	guardduty:DisassociateFromMasterAccount
	guardduty:DisassociateMembers
	guardduty:EnableOrganizationAdminAccount
	guardduty:GetAdministratorAccount
	guardduty:GetCoverageStatistics
	guardduty:GetDetector
	guardduty:GetFilter
	guardduty:GetFindings
	guardduty:GetFindingsStatistics
	guardduty:GetInvitationsCount
	guardduty:GetIPSet
	guardduty:GetMalwareProtectionPlan
	guardduty:GetMalwareScanSettings
	guardduty:GetMasterAccount
	guardduty:GetMemberDetectors
	guardduty:GetMembers
	guardduty:GetOrganizationStatistics

서비스 접두사	작업
	guardduty:GetRemainingFreeTrialDays
	guardduty:GetThreatIntelSet
	guardduty:GetUsageStatistics
	guardduty:InviteMembers
	guardduty:ListCoverage
	guardduty:ListDetectors
	guardduty:ListFilters
	guardduty:ListFindings
	guardduty:ListInvitations
	guardduty:ListIPSets
	guardduty:ListMalwareProtectionPlans
	guardduty:ListMembers
	guardduty:ListOrganizationAdminAccounts
	guardduty:ListPublishingDestinations
	guardduty:ListThreatIntelSets
	guardduty:SendSecurityTelemetry
	guardduty:StartMalwareScan
	guardduty:StartMonitoringMembers
	guardduty:StopMonitoringMembers
	guardduty:UnarchiveFindings
	guardduty:UpdateDetector

서비스 접두사	작업
	guardduty:UpdateFilter guardduty:UpdateFindingsFeedback guardduty:UpdateIPSet guardduty:UpdateMalwareProtectionPlan guardduty:UpdateMalwareScanSettings guardduty:UpdateMemberDetectors guardduty:UpdateOrganizationConfiguration guardduty:UpdatePublishingDestination guardduty:UpdateThreatIntelSet

서비스 접두사	작업
healthlake	healthlake:CreateFHIRDatastore healthlake:CreateResource healthlake:DeleteFHIRDatastore healthlake:DeleteResource healthlake:DescribeFHIRDatastore healthlake:DescribeFHIRExportJob healthlake:DescribeFHIRImportJob healthlake:GetCapabilities healthlake:ListFHIRDatastores healthlake:ListFHIRExportJobs healthlake:ListFHIRImportJobs healthlake:ReadResource healthlake:SearchEverything healthlake:SearchWithGet healthlake:SearchWithPost healthlake:StartFHIRExportJob healthlake:StartFHIRImportJob healthlake:UpdateResource

서비스 접두사	작업
honeycode	honeycode:BatchCreateTableRows
	honeycode:BatchDeleteTableRows
	honeycode:BatchUpdateTableRows
	honeycode:BatchUpsertTableRows
	honeycode:DescribeTableDataImportJob
	honeycode:GetScreenData
	honeycode:InvokeScreenAutomation
	honeycode:ListTableColumns
	honeycode:ListTableRows
	honeycode:ListTables
	honeycode:QueryTableRows
	honeycode:StartTableDataImportJob

서비스 접두사	작업
iam	iam:AddClientIDToOpenIDConnectProvider iam:AddRoleToInstanceProfile iam:AddUserToGroup iam:AttachGroupPolicy iam:AttachRolePolicy iam:AttachUserPolicy iam:ChangePassword iam:CreateAccessKey iam:CreateAccountAlias iam:CreateGroup iam:CreateInstanceProfile iam:CreateLoginProfile iam:CreateOpenIDConnectProvider iam:CreatePolicy iam:CreatePolicyVersion iam:CreateRole iam:CreateSAMLProvider iam:CreateServiceLinkedRole iam:CreateServiceSpecificCredential iam:CreateUser iam:CreateVirtualMFADevice

서비스 접두사	작업
	iam:DeactivateMFADevice iam:DeleteAccessKey iam:DeleteAccountAlias iam:DeleteAccountPasswordPolicy iam:DeleteCloudFrontPublicKey iam:DeleteGroup iam:DeleteGroupPolicy iam:DeleteInstanceProfile iam:DeleteLoginProfile iam:DeleteOpenIDConnectProvider iam:DeletePolicy iam:DeletePolicyVersion iam:DeleteRole iam:DeleteRolePermissionsBoundary iam:DeleteRolePolicy iam:DeleteSAMLProvider iam:DeleteServerCertificate iam:DeleteServiceLinkedRole iam:DeleteServiceSpecificCredential iam:DeleteSigningCertificate iam:DeleteSSHPublicKey

서비스 접두사	작업
	iam:DeleteUser iam:DeleteUserPermissionsBoundary iam:DeleteUserPolicy iam:DeleteVirtualMFADevice iam:DetachGroupPolicy iam:DetachRolePolicy iam:DetachUserPolicy iam:EnableMFADevice iam:GenerateCredentialReport iam:GenerateOrganizationsAccessReport iam:GenerateServiceLastAccessedDetails iam:GetAccessKeyLastUsed iam:GetAccountAuthorizationDetails iam:GetAccountEmailAddress iam:GetAccountName iam:GetAccountPasswordPolicy iam:GetAccountSummary iam:GetCloudFrontPublicKey iam:GetContextKeysForCustomPolicy iam:GetContextKeysForPrincipalPolicy iam:GetCredentialReport

서비스 접두사	작업
	iam:GetGroup iam:GetGroupPolicy iam:GetInstanceProfile iam:GetLoginProfile iam:GetMFADevice iam:GetOpenIDConnectProvider iam:GetOrganizationsAccessReport iam:GetPolicy iam:GetPolicyVersion iam:GetRole iam:GetRolePolicy iam:GetSAMLProvider iam:GetServerCertificate iam:GetServiceLastAccessedDetails iam:GetServiceLastAccessedDetailsWithEntities iam:GetServiceLinkedRoleDeletionStatus iam:GetSSHPublicKey iam:GetUser iam:GetUserPolicy iam:ListAccessKeys iam:ListAccountAliases

서비스 접두사	작업
	iam:ListAttachedGroupPolicies iam:ListAttachedRolePolicies iam:ListAttachedUserPolicies iam:ListCloudFrontPublicKeys iam:ListEntitiesForPolicy iam:ListGroupPolicies iam:ListGroups iam:ListGroupsForUser iam:ListInstanceProfiles iam:ListInstanceProfilesForRole iam:ListMFADevices iam:ListOpenIDConnectProviders iam:ListPolicies iam:ListPoliciesGrantingServiceAccess iam:ListPolicyVersions iam:ListRolePolicies iam:ListRoles iam:ListSAMLProviders iam:ListServerCertificates iam:ListServiceSpecificCredentials iam:ListSigningCertificates

서비스 접두사	작업
	iam:ListSSHPublicKeys iam:ListSTSRegionalEndpointsStatus iam:ListUserPolicies iam:ListUsers iam:ListVirtualMFADevices iam:PutGroupPolicy iam:PutRolePermissionsBoundary iam:PutRolePolicy iam:PutUserPermissionsBoundary iam:PutUserPolicy iam:RemoveClientIDFromOpenIDConnectProvider iam:RemoveRoleFromInstanceProfile iam:RemoveUserFromGroup iam:ResetServiceSpecificCredential iam:ResyncMFADevice iam:SetDefaultPolicyVersion iam:SetSecurityTokenServicePreferences iam:SetSTSRegionalEndpointStatus iam:SimulateCustomPolicy iam:SimulatePrincipalPolicy iam:UpdateAccessKey

서비스 접두사	작업
	iam:UpdateAccountEmailAddress iam:UpdateAccountName iam:UpdateAccountPasswordPolicy iam:UpdateAssumeRolePolicy iam:UpdateCloudFrontPublicKey iam:UpdateGroup iam:UpdateLoginProfile iam:UpdateOpenIDConnectProviderThumbprint iam:UpdateRole iam:UpdateRoleDescription iam:UpdateSAMLProvider iam:UpdateServerCertificate iam:UpdateServiceSpecificCredential iam:UpdateSigningCertificate iam:UpdateSSHPublicKey iam:UpdateUser iam:UploadCloudFrontPublicKey iam:UploadServerCertificate iam:UploadSigningCertificate iam:UploadSSHPublicKey

서비스 접두사	작업
identitystore	identitystore:CreateGroup identitystore:CreateGroupMembership identitystore:CreateUser identitystore:DeleteGroup identitystore:DeleteGroupMembership identitystore:DeleteUser identitystore:DescribeGroup identitystore:DescribeGroupMembership identitystore:DescribeUser identitystore:GetGroupId identitystore:GetGroupMembershipId identitystore:GetUserId identitystore:IsMemberInGroups identitystore:ListGroupMemberships identitystore:ListGroupMembershipsForMember identitystore:ListGroups identitystore:ListUsers identitystore:UpdateGroup identitystore:UpdateUser

서비스 접두사	작업
imagebuilder	imagebuilder:CancelImageCreation imagebuilder:CancelLifecycleExecution imagebuilder:CreateComponent imagebuilder:CreateContainerRecipe imagebuilder:CreateDistributionConfiguration imagebuilder:CreateImage imagebuilder:CreateImagePipeline imagebuilder:CreateImageRecipe imagebuilder:CreateInfrastructureConfiguration imagebuilder:CreateLifecyclePolicy imagebuilder:CreateWorkflow imagebuilder:DeleteComponent imagebuilder:DeleteContainerRecipe imagebuilder:DeleteDistributionConfiguration imagebuilder:DeleteImage imagebuilder:DeleteImagePipeline imagebuilder:DeleteImageRecipe imagebuilder:DeleteInfrastructureConfiguration imagebuilder:DeleteLifecyclePolicy imagebuilder:DeleteWorkflow imagebuilder:GetComponentPolicy

서비스 접두사	작업
	imagebuilder:GetContainerRecipePolicy imagebuilder:GetImagePolicy imagebuilder:GetImageRecipePolicy imagebuilder:GetLifecycleExecution imagebuilder:GetLifecyclePolicy imagebuilder:GetWorkflowExecution imagebuilder:GetWorkflowStepExecution imagebuilder:ImportComponent imagebuilder:ImportVmImage imagebuilder:ListComponentBuildVersions imagebuilder:ListComponents imagebuilder:ListContainerRecipes imagebuilder:ListDistributionConfigurations imagebuilder:ListImageBuildVersions imagebuilder:ListImagePackages imagebuilder:ListImagePipelineImages imagebuilder:ListImagePipelines imagebuilder:ListImageRecipes imagebuilder:ListImages imagebuilder:ListImageScanFindingAggregations imagebuilder:ListImageScanFindings

서비스 접두사	작업
	imagebuilder:ListInfrastructureConfigurations imagebuilder:ListLifecycleExecutionResources imagebuilder:ListLifecycleExecutions imagebuilder:ListLifecyclePolicies imagebuilder:ListWaitingWorkflowSteps imagebuilder:ListWorkflowExecutions imagebuilder:ListWorkflows imagebuilder:ListWorkflowStepExecutions imagebuilder:PutComponentPolicy imagebuilder:PutContainerRecipePolicy imagebuilder:PutImagePolicy imagebuilder:PutImageRecipePolicy imagebuilder:SendWorkflowStepAction imagebuilder:StartImagePipelineExecution imagebuilder:StartResourceStateUpdate imagebuilder:UpdateDistributionConfiguration imagebuilder:UpdateImagePipeline imagebuilder:UpdateInfrastructureConfiguration

서비스 접두사	작업
inspector	inspector:AddAttributesToFindings
	inspector:CreateAssessmentTarget
	inspector:CreateAssessmentTemplate
	inspector:CreateExclusionsPreview
	inspector:CreateResourceGroup
	inspector:DeleteAssessmentRun
	inspector:DeleteAssessmentTarget
	inspector:DeleteAssessmentTemplate
	inspector:DescribeAssessmentRuns
	inspector:DescribeAssessmentTargets
	inspector:DescribeAssessmentTemplates
	inspector:DescribeCrossAccountAccessRole
	inspector:DescribeExclusions
	inspector:DescribeFindings
	inspector:DescribeResourceGroups
	inspector:DescribeRulesPackages
	inspector:GetAssessmentReport
	inspector:GetExclusionsPreview
	inspector:GetTelemetryMetadata
	inspector:ListAssessmentRunAgents
	inspector:ListAssessmentRuns

서비스 접두사	작업
	inspector:ListAssessmentTargets
	inspector:ListAssessmentTemplates
	inspector:ListEventSubscriptions
	inspector:ListExclusions
	inspector:ListFindings
	inspector:ListRulesPackages
	inspector:PreviewAgents
	inspector:RegisterCrossAccountAccessRole
	inspector:RemoveAttributesFromFindings
	inspector:StartAssessmentRun
	inspector:StopAssessmentRun
	inspector:SubscribeToEvent
	inspector:UnsubscribeFromEvent
	inspector:UpdateAssessmentTarget

서비스 접두사	작업
inspector2	inspector2:AssociateMember inspector2:BatchGetAccountStatus inspector2:BatchGetCodeSnippet inspector2:BatchGetFindingDetails inspector2:BatchGetFreeTrialInfo inspector2:BatchGetMemberEc2DeepInspectionStatus inspector2:BatchUpdateMemberEc2DeepInspectionStatus inspector2:CancelFindingsReport inspector2:CancelSbomExport inspector2:CreateCisScanConfiguration inspector2:CreateFilter inspector2:CreateFindingsReport inspector2:CreateSbomExport inspector2:DeleteCisScanConfiguration inspector2:DeleteFilter inspector2:DescribeOrganizationConfiguration inspector2:Disable inspector2:DisableDelegatedAdminAccount inspector2:DisassociateMember inspector2:Enable inspector2:EnableDelegatedAdminAccount

서비스 접두사	작업
	inspector2:GetCisScanReport inspector2:GetCisScanResultDetails inspector2:GetConfiguration inspector2:GetDelegatedAdminAccount inspector2:GetEc2DeepInspectionConfiguration inspector2:GetEncryptionKey inspector2:GetFindingsReportStatus inspector2:GetMember inspector2:GetSbomExport inspector2:ListAccountPermissions inspector2:ListCisScanConfigurations inspector2:ListCisScanResultsAggregatedByChecks inspector2:ListCisScanResultsAggregatedByTargetResource inspector2:ListCisScans inspector2:ListCoverage inspector2:ListCoverageStatistics inspector2:ListDelegatedAdminAccounts inspector2:ListFilters inspector2:ListFindingAggregations inspector2:ListFindings inspector2:ListMembers

서비스 접두사	작업
	inspector2:ListUsageTotals inspector2:ResetEncryptionKey inspector2:SearchVulnerabilities inspector2:SendCisSessionHealth inspector2:SendCisSessionTelemetry inspector2:StartCisSession inspector2:StopCisSession inspector2:UpdateCisScanConfiguration inspector2:UpdateConfiguration inspector2:UpdateEc2DeepInspectionConfiguration inspector2:UpdateEncryptionKey inspector2:UpdateFilter inspector2:UpdateOrganizationConfiguration inspector2:UpdateOrgEc2DeepInspectionConfiguration

서비스 접두사	작업
iot	iot:AcceptCertificateTransfer
	iot:AddThingToBillingGroup
	iot:AddThingToThingGroup
	iot:AssociateTargetsWithJob
	iot:AttachPolicy
	iot:AttachPrincipalPolicy
	iot:AttachSecurityProfile
	iot:AttachThingPrincipal
	iot:CancelAuditMitigationActionsTask
	iot:CancelAuditTask
	iot:CancelCertificateTransfer
	iot:CancelDetectMitigationActionsTask
	iot:CancelJob
	iot:CancelJobExecution
	iot:ClearDefaultAuthorizer
	iot:ConfirmTopicRuleDestination
	iot:CreateAuditSuppression
	iot:CreateAuthorizer
	iot:CreateBillingGroup
	iot:CreateCertificateFromCsr
	iot:CreateCertificateProvider

서비스 접두사	작업
	iot:CreateCustomMetric
	iot:CreateDimension
	iot:CreateDomainConfiguration
	iot:CreateDynamicThingGroup
	iot:CreateFleetMetric
	iot:CreateJob
	iot:CreateJobTemplate
	iot:CreateKeysAndCertificate
	iot:CreateMitigationAction
	iot:CreateOTAUpdate
	iot:CreatePackage
	iot:CreatePackageVersion
	iot:CreatePolicy
	iot:CreatePolicyVersion
	iot:CreateProvisioningClaim
	iot:CreateProvisioningTemplate
	iot:CreateProvisioningTemplateVersion
	iot:CreateRoleAlias
	iot:CreateScheduledAudit
	iot:CreateSecurityProfile
	iot:CreateStream

서비스 접두사	작업
	iot:CreateThing
	iot:CreateThingGroup
	iot:CreateThingType
	iot:CreateTopicRule
	iot:CreateTopicRuleDestination
	iot:DeleteAccountAuditConfiguration
	iot:DeleteAuditSuppression
	iot:DeleteAuthorizer
	iot:DeleteBillingGroup
	iot:DeleteCACertificate
	iot:DeleteCertificate
	iot:DeleteCertificateProvider
	iot:DeleteCustomMetric
	iot:DeleteDimension
	iot:DeleteDomainConfiguration
	iot:DeleteDynamicThingGroup
	iot:DeleteFleetMetric
	iot:DeleteJob
	iot:DeleteJobExecution
	iot:DeleteJobTemplate
	iot:DeleteMitigationAction

서비스 접두사	작업
	iot:DeleteOTAUpdate
	iot:DeletePackage
	iot:DeletePackageVersion
	iot:DeletePolicy
	iot:DeletePolicyVersion
	iot:DeleteProvisioningTemplate
	iot:DeleteProvisioningTemplateVersion
	iot:DeleteRegistrationCode
	iot:DeleteRoleAlias
	iot:DeleteScheduledAudit
	iot:DeleteSecurityProfile
	iot:DeleteStream
	iot:DeleteThing
	iot:DeleteThingGroup
	iot:DeleteThingType
	iot:DeleteTopicRule
	iot:DeleteTopicRuleDestination
	iot:DeleteV2LoggingLevel
	iot:DeprecateThingType
	iot:DescribeAccountAuditConfiguration
	iot:DescribeAuditFinding

서비스 접두사	작업
	iot:DescribeAuditMitigationActionsTask
	iot:DescribeAuditSuppression
	iot:DescribeAuditTask
	iot:DescribeAuthorizer
	iot:DescribeBillingGroup
	iot:DescribeCACertificate
	iot:DescribeCertificate
	iot:DescribeCertificateProvider
	iot:DescribeCustomMetric
	iot:DescribeDefaultAuthorizer
	iot:DescribeDetectMitigationActionsTask
	iot:DescribeDimension
	iot:DescribeDomainConfiguration
	iot:DescribeEndpoint
	iot:DescribeEventConfigurations
	iot:DescribeFleetMetric
	iot:DescribeIndex
	iot:DescribeJob
	iot:DescribeJobExecution
	iot:DescribeJobTemplate
	iot:DescribeManagedJobTemplate

서비스 접두사	작업
	iot:DescribeMitigationAction
	iot:DescribeProvisioningTemplate
	iot:DescribeProvisioningTemplateVersion
	iot:DescribeRoleAlias
	iot:DescribeScheduledAudit
	iot:DescribeSecurityProfile
	iot:DescribeStream
	iot:DescribeThing
	iot:DescribeThingGroup
	iot:DescribeThingRegistrationTask
	iot:DescribeThingType
	iot:DetachPolicy
	iot:DetachPrincipalPolicy
	iot:DetachSecurityProfile
	iot:DetachThingPrincipal
	iot:DisableTopicRule
	iot:EnableTopicRule
	iot:GetBehaviorModelTrainingSummaries
	iot:GetBucketsAggregation
	iot:GetCardinality
	iot:GetEffectivePolicies

서비스 접두사	작업
	iot:GetJobDocument
	iot:GetLoggingOptions
	iot:GetOTAUpdate
	iot:GetPackage
	iot:GetPackageConfiguration
	iot:GetPackageVersion
	iot:GetPercentiles
	iot:GetPolicy
	iot:GetPolicyVersion
	iot:GetRegistrationCode
	iot:GetStatistics
	iot:GetTopicRule
	iot:GetTopicRuleDestination
	iot:GetV2LoggingOptions
	iot:ListActiveViolations
	iot:ListAttachedPolicies
	iot:ListAuditFindings
	iot:ListAuditMitigationActionsExecutions
	iot:ListAuditMitigationActionsTasks
	iot:ListAuditSuppressions
	iot:ListAuditTasks

서비스 접두사	작업
	iot:ListAuthorizers
	iot:ListBillingGroups
	iot:ListCACertificates
	iot:ListCertificateProviders
	iot:ListCertificates
	iot:ListCertificatesByCA
	iot:ListCustomMetrics
	iot:ListDetectMitigationActionsExecutions
	iot:ListDetectMitigationActionsTasks
	iot:ListDimensions
	iot:ListDomainConfigurations
	iot:ListFleetMetrics
	iot:ListIndices
	iot:ListJobExecutionsForJob
	iot:ListJobExecutionsForThing
	iot:ListJobs
	iot:ListJobTemplates
	iot:ListManagedJobTemplates
	iot:ListMetricValues
	iot:ListMitigationActions
	iot:ListOTAUpdates

서비스 접두사	작업
	iot:ListOutgoingCertificates
	iot:ListPackages
	iot:ListPackageVersions
	iot:ListPolicies
	iot:ListPolicyPrincipals
	iot:ListPolicyVersions
	iot:ListPrincipalPolicies
	iot:ListPrincipalThings
	iot:ListProvisioningTemplates
	iot:ListProvisioningTemplateVersions
	iot:ListRelatedResourcesForAuditFinding
	iot:ListRoleAliases
	iot:ListScheduledAudits
	iot:ListSecurityProfiles
	iot:ListSecurityProfilesForTarget
	iot:ListStreams
	iot:ListTargetsForPolicy
	iot:ListTargetsForSecurityProfile
	iot:ListThingGroups
	iot:ListThingGroupsForThing
	iot:ListThingPrincipals

서비스 접두사	작업
	iot:ListThingRegistrationTaskReports
	iot:ListThingRegistrationTasks
	iot:ListThings
	iot:ListThingsInBillingGroup
	iot:ListThingsInThingGroup
	iot:ListThingTypes
	iot:ListTopicRuleDestinations
	iot:ListTopicRules
	iot:ListV2LoggingLevels
	iot:ListViolationEvents
	iot:PutVerificationStateOnViolation
	iot:RegisterCACertificate
	iot:RegisterCertificate
	iot:RegisterCertificateWithoutCA
	iot:RegisterThing
	iot:RejectCertificateTransfer
	iot:RemoveThingFromBillingGroup
	iot:RemoveThingFromThingGroup
	iot:ReplaceTopicRule
	iot:SearchIndex
	iot:SetDefaultAuthorizer

서비스 접두사	작업
	iot:SetDefaultPolicyVersion
	iot:SetLoggingOptions
	iot:SetV2LoggingLevel
	iot:SetV2LoggingOptions
	iot:StartAuditMitigationActionsTask
	iot:StartDetectMitigationActionsTask
	iot:StartOnDemandAuditTask
	iot:StartThingRegistrationTask
	iot:StopThingRegistrationTask
	iot:TestAuthorization
	iot:TestInvokeAuthorizer
	iot:TransferCertificate
	iot:UpdateAccountAuditConfiguration
	iot:UpdateAuditSuppression
	iot:UpdateAuthorizer
	iot:UpdateBillingGroup
	iot:UpdateCACertificate
	iot:UpdateCertificate
	iot:UpdateCertificateProvider
	iot:UpdateCustomMetric
	iot:UpdateDimension

서비스 접두사	작업
	iot:UpdateDomainConfiguration
	iot:UpdateDynamicThingGroup
	iot:UpdateEventConfigurations
	iot:UpdateFleetMetric
	iot:UpdateIndexingConfiguration
	iot:UpdateJob
	iot:UpdateMitigationAction
	iot:UpdatePackage
	iot:UpdatePackageConfiguration
	iot:UpdatePackageVersion
	iot:UpdateProvisioningTemplate
	iot:UpdateRoleAlias
	iot:UpdateScheduledAudit
	iot:UpdateSecurityProfile
	iot:UpdateStream
	iot:UpdateThing
	iot:UpdateThingGroup
	iot:UpdateThingGroupsForThing
	iot:UpdateTopicRuleDestination
	iot:ValidateSecurityProfileBehaviors

서비스 접두사	작업
iotanalytics	iotanalytics:CancelPipelineReprocessing iotanalytics:CreateChannel iotanalytics:CreateDataset iotanalytics:CreateDatasetContent iotanalytics:CreateDatastore iotanalytics:CreatePipeline iotanalytics:DeleteChannel iotanalytics:DeleteDataset iotanalytics:DeleteDatasetContent iotanalytics:DeleteDatastore iotanalytics:DeletePipeline iotanalytics:DescribeChannel iotanalytics:DescribeDataset iotanalytics:DescribeDatastore iotanalytics:DescribeLoggingOptions iotanalytics:DescribePipeline iotanalytics:GetDatasetContent iotanalytics:ListChannels iotanalytics:ListDatasetContents iotanalytics:ListDatasets iotanalytics:ListDatastores

서비스 접두사	작업
	iotanalytics:ListPipelines iotanalytics:PutLoggingOptions iotanalytics:RunPipelineActivity iotanalytics:SampleChannelData iotanalytics:StartPipelineReprocessing iotanalytics:UpdateChannel iotanalytics:UpdateDataset iotanalytics:UpdateDatastore iotanalytics:UpdatePipeline
iotdeviceadvisor	iotdeviceadvisor:CreateSuiteDefinition iotdeviceadvisor:DeleteSuiteDefinition iotdeviceadvisor:GetEndpoint iotdeviceadvisor:GetSuiteDefinition iotdeviceadvisor:GetSuiteRun iotdeviceadvisor:GetSuiteRunReport iotdeviceadvisor:ListSuiteDefinitions iotdeviceadvisor:ListSuiteRuns iotdeviceadvisor:StartSuiteRun iotdeviceadvisor:StopSuiteRun iotdeviceadvisor:UpdateSuiteDefinition

서비스 접두사	작업
iotevents	iotevents:BatchAcknowledgeAlarm iotevents:BatchDeleteDetector iotevents:BatchDisableAlarm iotevents:BatchEnableAlarm iotevents:BatchResetAlarm iotevents:BatchSnoozeAlarm iotevents:BatchUpdateDetector iotevents:CreateAlarmModel iotevents:CreateDetectorModel iotevents:CreateInput iotevents:DeleteAlarmModel iotevents:DeleteDetectorModel iotevents:DeleteInput iotevents:DescribeAlarm iotevents:DescribeAlarmModel iotevents:DescribeDetector iotevents:DescribeDetectorModel iotevents:DescribeDetectorModelAnalysis iotevents:DescribeInput iotevents:DescribeLoggingOptions iotevents:GetDetectorModelAnalysisResults

서비스 접두사	작업
	iotevents:ListAlarmModels iotevents:ListAlarmModelVersions iotevents:ListAlarms iotevents:ListDetectorModels iotevents:ListDetectorModelVersions iotevents:ListDetectors iotevents:ListInputRoutings iotevents:ListInputs iotevents:PutLoggingOptions iotevents:StartDetectorModelAnalysis iotevents:UpdateAlarmModel iotevents:UpdateDetectorModel iotevents:UpdateInput
iotfleethub	iotfleethub:CreateApplication iotfleethub:DeleteApplication iotfleethub:DescribeApplication iotfleethub:ListApplications iotfleethub:UpdateApplication

서비스 접두사	작업
iotsitewise	iotsitewise:AssociateAssets iotsitewise:AssociateTimeSeriesToAssetProperty iotsitewise:BatchAssociateProjectAssets iotsitewise:BatchDisassociateProjectAssets iotsitewise:BatchGetAssetPropertyValue iotsitewise:BatchGetAssetPropertyValueHistory iotsitewise:BatchPutAssetPropertyValue iotsitewise:CreateAccessPolicy iotsitewise:CreateAsset iotsitewise:CreateAssetModel iotsitewise:CreateAssetModelCompositeModel iotsitewise:CreateBulkImportJob iotsitewise:CreateDashboard iotsitewise:CreateGateway iotsitewise:CreatePortal iotsitewise:CreateProject iotsitewise:DeleteAccessPolicy iotsitewise:DeleteAsset iotsitewise:DeleteAssetModel iotsitewise:DeleteAssetModelCompositeModel iotsitewise:DeleteDashboard

서비스 접두사	작업
	iotsitewise:DeleteGateway iotsitewise:DeletePortal iotsitewise:DeleteProject iotsitewise:DeleteTimeSeries iotsitewise:DescribeAccessPolicy iotsitewise:DescribeAsset iotsitewise:DescribeAssetCompositeModel iotsitewise:DescribeAssetModel iotsitewise:DescribeAssetModelCompositeModel iotsitewise:DescribeAssetProperty iotsitewise:DescribeBulkImportJob iotsitewise:DescribeDashboard iotsitewise:DescribeDefaultEncryptionConfiguration iotsitewise:DescribeGateway iotsitewise:DescribeGatewayCapabilityConfiguration iotsitewise:DescribeLoggingOptions iotsitewise:DescribePortal iotsitewise:DescribeProject iotsitewise:DescribeStorageConfiguration iotsitewise:DescribeTimeSeries iotsitewise:DisassociateAssets

서비스 접두사	작업
	iotsitewise:DisassociateTimeSeriesFromAssetProperty iotsitewise:ExecuteAction iotsitewise:ExecuteQuery iotsitewise:ListAccessPolicies iotsitewise:ListActions iotsitewise:ListAssetModelCompositeModels iotsitewise:ListAssetModelProperties iotsitewise:ListAssetModels iotsitewise:ListAssetProperties iotsitewise:ListAssetRelationships iotsitewise:ListAssets iotsitewise:ListAssociatedAssets iotsitewise:ListBulkImportJobs iotsitewise:ListCompositionRelationships iotsitewise:ListDashboards iotsitewise:ListGateways iotsitewise:ListPortals iotsitewise:ListProjectAssets iotsitewise:ListProjects iotsitewise:ListTimeSeries iotsitewise:PutDefaultEncryptionConfiguration

서비스 접두사	작업
	iotsitewise:PutLoggingOptions iotsitewise:PutStorageConfiguration iotsitewise:UpdateAccessPolicy iotsitewise:UpdateAsset iotsitewise:UpdateAssetModel iotsitewise:UpdateAssetModelCompositeModel iotsitewise:UpdateAssetProperty iotsitewise:UpdateDashboard iotsitewise:UpdateGateway iotsitewise:UpdateGatewayCapabilityConfiguration iotsitewise:UpdatePortal iotsitewise:UpdateProject

서비스 접두사	작업
iottwinmaker	iottwinmaker:CancelMetadataTransferJob
	iottwinmaker:CreateComponentType
	iottwinmaker:CreateEntity
	iottwinmaker:CreateMetadataTransferJob
	iottwinmaker:CreateScene
	iottwinmaker:CreateSyncJob
	iottwinmaker:CreateWorkspace
	iottwinmaker:DeleteComponentType
	iottwinmaker:DeleteEntity
	iottwinmaker:DeleteScene
	iottwinmaker:DeleteSyncJob
	iottwinmaker:DeleteWorkspace
	iottwinmaker:ExecuteQuery
	iottwinmaker:GetMetadataTransferJob
	iottwinmaker:GetPricingPlan
	iottwinmaker:GetScene
	iottwinmaker:GetSyncJob
	iottwinmaker:ListComponents
	iottwinmaker:ListComponentTypes
	iottwinmaker:ListEntities
	iottwinmaker:ListMetadataTransferJobs

서비스 접두사	작업
	iottwinmaker:ListProperties
	iottwinmaker:ListScenes
	iottwinmaker:ListSyncJobs
	iottwinmaker:ListSyncResources
	iottwinmaker:ListWorkspaces
	iottwinmaker:UpdateComponentType
	iottwinmaker:UpdateEntity
	iottwinmaker:UpdatePricingPlan
	iottwinmaker:UpdateScene
	iottwinmaker:UpdateWorkspace

서비스 접두사	작업
iotwireless	iotwireless:AssociateAwsAccountWithPartnerAccount iotwireless:AssociateMulticastGroupWithFuotaTask iotwireless:AssociateWirelessDeviceWithFuotaTask iotwireless:AssociateWirelessDeviceWithMulticastGroup iotwireless:AssociateWirelessDeviceWithThing iotwireless:AssociateWirelessGatewayWithCertificate iotwireless:AssociateWirelessGatewayWithThing iotwireless:CancelMulticastGroupSession iotwireless:CreateDestination iotwireless:CreateDeviceProfile iotwireless:CreateFuotaTask iotwireless:CreateMulticastGroup iotwireless:CreateNetworkAnalyzerConfiguration iotwireless:CreateServiceProfile iotwireless:CreateWirelessDevice iotwireless:CreateWirelessGateway iotwireless:CreateWirelessGatewayTask iotwireless:CreateWirelessGatewayTaskDefinition iotwireless:DeleteDestination iotwireless:DeleteDeviceProfile iotwireless:DeleteFuotaTask

서비스 접두사	작업
	iotwireless:DeleteMulticastGroup iotwireless:DeleteNetworkAnalyzerConfiguration iotwireless:DeleteQueuedMessages iotwireless:DeleteServiceProfile iotwireless:DeleteWirelessDevice iotwireless:DeleteWirelessDeviceImportTask iotwireless:DeleteWirelessGateway iotwireless:DeleteWirelessGatewayTask iotwireless:DeleteWirelessGatewayTaskDefinition iotwireless:DeregisterWirelessDevice iotwireless:DisassociateAwsAccountFromPartnerAccount iotwireless:DisassociateMulticastGroupFromFuotaTask iotwireless:DisassociateWirelessDeviceFromFuotaTask iotwireless:DisassociateWirelessDeviceFromMulticastGroup iotwireless:DisassociateWirelessDeviceFromThing iotwireless:DisassociateWirelessGatewayFromCertificate iotwireless:DisassociateWirelessGatewayFromThing iotwireless:GetDestination iotwireless:GetDeviceProfile iotwireless:GetEventConfigurationByResourceTypes iotwireless:GetFuotaTask

서비스 접두사	작업
	iotwireless:GetLogLevelsByResourceTypes iotwireless:GetMetricConfiguration iotwireless:GetMetrics iotwireless:GetMulticastGroup iotwireless:GetMulticastGroupSession iotwireless:GetNetworkAnalyzerConfiguration iotwireless:GetPartnerAccount iotwireless:GetPosition iotwireless:GetPositionConfiguration iotwireless:GetPositionEstimate iotwireless:GetResourceEventConfiguration iotwireless:GetResourceLogLevel iotwireless:GetResourcePosition iotwireless:GetServiceEndpoint iotwireless:GetServiceProfile iotwireless:GetWirelessDevice iotwireless:GetWirelessDeviceImportTask iotwireless:GetWirelessDeviceStatistics iotwireless:GetWirelessGateway iotwireless:GetWirelessGatewayCertificate iotwireless:GetWirelessGatewayFirmwareInformation

서비스 접두사	작업
	iotwireless:GetWirelessGatewayStatistics iotwireless:GetWirelessGatewayTask iotwireless:GetWirelessGatewayTaskDefinition iotwireless:ListDestinations iotwireless:ListDeviceProfiles iotwireless:ListDevicesForWirelessDeviceImportTask iotwireless:ListEventConfigurations iotwireless:ListFuotaTasks iotwireless:ListMulticastGroups iotwireless:ListMulticastGroupsByFuotaTask iotwireless:ListNetworkAnalyzerConfigurations iotwireless:ListPartnerAccounts iotwireless:ListPositionConfigurations iotwireless:ListQueuedMessages iotwireless:ListServiceProfiles iotwireless:ListWirelessDeviceImportTasks iotwireless:ListWirelessDevices iotwireless:ListWirelessGateways iotwireless:ListWirelessGatewayTaskDefinitions iotwireless:PutPositionConfiguration iotwireless:PutResourceLogLevel

서비스 접두사	작업
	iotwireless:ResetAllResourceLogLevels iotwireless:ResetResourceLogLevel iotwireless:SendDataToMulticastGroup iotwireless:SendDataToWirelessDevice iotwireless:StartBulkAssociateWirelessDeviceWithMulticastGroup iotwireless:StartBulkDisassociateWirelessDeviceFromMulticastGroup iotwireless:StartFuotaTask iotwireless:StartMulticastGroupSession iotwireless:StartNetworkAnalyzerStream iotwireless:StartSingleWirelessDeviceImportTask iotwireless:StartWirelessDeviceImportTask iotwireless:TestWirelessDevice iotwireless:UpdateDestination iotwireless:UpdateEventConfigurationByResourceTypes iotwireless:UpdateFuotaTask iotwireless:UpdateLogLevelsByResourceTypes iotwireless:UpdateMetricConfiguration iotwireless:UpdateMulticastGroup iotwireless:UpdateNetworkAnalyzerConfiguration iotwireless:UpdatePartnerAccount

서비스 접두사	작업
	iotwireless:UpdatePosition iotwireless:UpdateResourceEventConfiguration iotwireless:UpdateResourcePosition iotwireless:UpdateWirelessDevice iotwireless:UpdateWirelessDeviceImportTask iotwireless:UpdateWirelessGateway

서비스 접두사	작업
ivs	ivs:BatchGetChannel
	ivs:BatchGetStreamKey
	ivs:BatchStartViewerSessionRevocation
	ivs:CreateChannel
	ivs:CreateEncoderConfiguration
	ivs:CreateParticipantToken
	ivs:CreatePlaybackRestrictionPolicy
	ivs:CreateRecordingConfiguration
	ivs:CreateStorageConfiguration
	ivs:CreateStreamKey
	ivs:DeleteChannel
	ivs:DeleteEncoderConfiguration
	ivs:DeletePlaybackKeyPair
	ivs:DeletePlaybackRestrictionPolicy
	ivs:DeleteRecordingConfiguration
	ivs:DeleteStorageConfiguration
	ivs:DeleteStreamKey
	ivs:DisconnectParticipant
	ivs:GetChannel
	ivs:GetComposition
	ivs:GetEncoderConfiguration

서비스 접두사	작업
	ivs:GetParticipant
	ivs:GetPlaybackKeyPair
	ivs:GetPlaybackRestrictionPolicy
	ivs:GetRecordingConfiguration
	ivs:GetStorageConfiguration
	ivs:GetStream
	ivs:GetStreamKey
	ivs:GetStreamSession
	ivs:ImportPlaybackKeyPair
	ivs:ListChannels
	ivs:ListCompositions
	ivs:ListEncoderConfigurations
	ivs:ListParticipantEvents
	ivs:ListParticipants
	ivs:ListPlaybackKeyPairs
	ivs:ListPlaybackRestrictionPolicies
	ivs:ListRecordingConfigurations
	ivs:ListStorageConfigurations
	ivs:ListStreamKeys
	ivs:ListStreams
	ivs:ListStreamSessions

서비스 접두사	작업
	ivs:PutMetadata ivs:StartComposition ivs:StartViewerSessionRevocation ivs:StopComposition ivs:StopStream ivs:UpdateChannel ivs:UpdatePlaybackRestrictionPolicy
ivschat	ivschat:CreateChatToken ivschat:CreateLoggingConfiguration ivschat:CreateRoom ivschat:DeleteLoggingConfiguration ivschat:DeleteMessage ivschat:DeleteRoom ivschat:DisconnectUser ivschat:GetLoggingConfiguration ivschat:GetRoom ivschat:ListLoggingConfigurations ivschat:ListRooms ivschat:SendEvent ivschat:UpdateLoggingConfiguration ivschat:UpdateRoom

서비스 접두사	작업
kafka	kafka:BatchAssociateScramSecret
	kafka:BatchDisassociateScramSecret
	kafka:CreateCluster
	kafka:CreateClusterV2
	kafka:CreateConfiguration
	kafka:CreateReplicator
	kafka:CreateVpcConnection
	kafka:DeleteCluster
	kafka:DeleteClusterPolicy
	kafka:DeleteConfiguration
	kafka:DeleteReplicator
	kafka:DeleteVpcConnection
	kafka:DescribeCluster
	kafka:DescribeClusterOperation
	kafka:DescribeClusterOperationV2
	kafka:DescribeClusterV2
	kafka:DescribeConfiguration
	kafka:DescribeConfigurationRevision
	kafka:DescribeVpcConnection
	kafka:GetBootstrapBrokers
	kafka:GetClusterPolicy

서비스 접두사	작업
	kafka:GetCompatibleKafkaVersions
	kafka:ListClientVpcConnections
	kafka:ListClusterOperations
	kafka:ListClusterOperationsV2
	kafka:ListClusters
	kafka:ListClustersV2
	kafka:ListConfigurationRevisions
	kafka:ListConfigurations
	kafka:ListKafkaVersions
	kafka:ListNodes
	kafka:ListReplicators
	kafka:ListScramSecrets
	kafka:ListVpcConnections
	kafka:PutClusterPolicy
	kafka:RebootBroker
	kafka:RejectClientVpcConnection
	kafka:UpdateBrokerCount
	kafka:UpdateBrokerStorage
	kafka:UpdateBrokerType
	kafka:UpdateClusterConfiguration
	kafka:UpdateClusterKafkaVersion

서비스 접두사	작업
	kafka:UpdateConfiguration kafka:UpdateConnectivity kafka:UpdateMonitoring kafka:UpdateReplicationInfo kafka:UpdateSecurity kafka:UpdateStorage
kafkaconnect	kafkaconnect:CreateConnector kafkaconnect:CreateCustomPlugin kafkaconnect:CreateWorkerConfiguration kafkaconnect:DeleteConnector kafkaconnect:DeleteCustomPlugin kafkaconnect:DeleteWorkerConfiguration kafkaconnect:DescribeConnector kafkaconnect:DescribeCustomPlugin kafkaconnect:DescribeWorkerConfiguration kafkaconnect:ListConnectors kafkaconnect:ListCustomPlugins kafkaconnect:ListWorkerConfigurations kafkaconnect:UpdateConnector

서비스 접두사	작업
kendra	kendra:AssociateEntitiesToExperience
	kendra:AssociatePersonasToEntities
	kendra:BatchDeleteDocument
	kendra:BatchDeleteFeaturedResultsSet
	kendra:BatchGetDocumentStatus
	kendra:BatchPutDocument
	kendra:ClearQuerySuggestions
	kendra:CreateAccessControlConfiguration
	kendra:CreateDataSource
	kendra:CreateExperience
	kendra:CreateFaq
	kendra:CreateFeaturedResultsSet
	kendra:CreateIndex
	kendra:CreateQuerySuggestionsBlockList
	kendra:CreateThesaurus
	kendra:DeleteDataSource
	kendra:DeleteExperience
	kendra:DeleteFaq
	kendra:DeleteIndex
	kendra:DeletePrincipalMapping
	kendra:DeleteQuerySuggestionsBlockList

서비스 접두사	작업
	kendra:DeleteThesaurus
	kendra:DescribeAccessControlConfiguration
	kendra:DescribeDataSource
	kendra:DescribeExperience
	kendra:DescribeFaq
	kendra:DescribeFeaturedResultsSet
	kendra:DescribeIndex
	kendra:DescribePrincipalMapping
	kendra:DescribeQuerySuggestionsBlockList
	kendra:DescribeQuerySuggestionsConfig
	kendra:DescribeThesaurus
	kendra:DisassociateEntitiesFromExperience
	kendra:DisassociatePersonasFromEntities
	kendra:GetQuerySuggestions
	kendra:GetSnapshots
	kendra:ListAccessControlConfigurations
	kendra:ListDataSources
	kendra:ListDataSourceSyncJobs
	kendra:ListEntityPersonas
	kendra:ListExperienceEntities
	kendra:ListExperiences

서비스 접두사	작업
	kendra:ListFaqs
	kendra:ListFeaturedResultsSets
	kendra:ListGroupsOlderThanOrderingId
	kendra:ListIndices
	kendra:ListQuerySuggestionsBlockLists
	kendra:ListThesauri
	kendra:PutPrincipalMapping
	kendra:Query
	kendra:Retrieve
	kendra:StartDataSourceSyncJob
	kendra:StopDataSourceSyncJob
	kendra:SubmitFeedback
	kendra:UpdateDataSource
	kendra:UpdateExperience
	kendra:UpdateFeaturedResultsSet
	kendra:UpdateIndex
	kendra:UpdateQuerySuggestionsBlockList
	kendra:UpdateQuerySuggestionsConfig
	kendra:UpdateThesaurus

서비스 접두사	작업
kinesis	kinesis:CreateStream kinesis:DecreaseStreamRetentionPeriod kinesis:DeleteStream kinesis:DeregisterStreamConsumer kinesis:DescribeLimits kinesis:DescribeStream kinesis:DescribeStreamConsumer kinesis:DescribeStreamSummary kinesis:DisableEnhancedMonitoring kinesis:EnableEnhancedMonitoring kinesis:IncreaseStreamRetentionPeriod kinesis:ListShards kinesis:ListStreamConsumers kinesis:ListStreams kinesis:MergeShards kinesis:RegisterStreamConsumer kinesis:SplitShard kinesis:StartStreamEncryption kinesis:StopStreamEncryption kinesis:UpdateShardCount kinesis:UpdateStreamMode

서비스 접두사	작업
kinesisanalytics	kinesisanalytics:AddApplicationCloudWatchLoggingOption
	kinesisanalytics:AddApplicationInput
	kinesisanalytics:AddApplicationInputProcessingConfiguration
	kinesisanalytics:AddApplicationOutput
	kinesisanalytics:AddApplicationReferenceDataSource
	kinesisanalytics:AddApplicationVpcConfiguration
	kinesisanalytics:CreateApplication
	kinesisanalytics:CreateApplicationPresignedUrl
	kinesisanalytics:CreateApplicationSnapshot
	kinesisanalytics:DeleteApplication
	kinesisanalytics:DeleteApplicationCloudWatchLoggingOption
	kinesisanalytics:DeleteApplicationInputProcessingConfiguration
	kinesisanalytics:DeleteApplicationOutput
	kinesisanalytics:DeleteApplicationReferenceDataSource
	kinesisanalytics:DeleteApplicationSnapshot
	kinesisanalytics:DeleteApplicationVpcConfiguration
	kinesisanalytics:DescribeApplication
	kinesisanalytics:DescribeApplicationSnapshot
	kinesisanalytics:DescribeApplicationVersion
	kinesisanalytics:DiscoverInputSchema
	kinesisanalytics:ListApplications

서비스 접두사	작업
	kinesisanalytics:ListApplicationSnapshots kinesisanalytics:ListApplicationVersions kinesisanalytics:RollbackApplication kinesisanalytics:StartApplication kinesisanalytics:StopApplication kinesisanalytics:UpdateApplication kinesisanalytics:UpdateApplicationMaintenanceConfiguration

서비스 접두사	작업
kms	kms:CancelKeyDeletion kms:ConnectCustomKeyStore kms:CreateAlias kms:CreateCustomKeyStore kms:CreateGrant kms:CreateKey kms:Decrypt kms:DeleteAlias kms:DeleteCustomKeyStore kms:DeleteImportedKeyMaterial kms:DeriveSharedSecret kms:DescribeCustomKeyStores kms:DescribeKey kms:DisableKey kms:DisableKeyRotation kms:DisconnectCustomKeyStore kms:EnableKey kms:EnableKeyRotation kms:Encrypt kms:GenerateDataKey kms:GenerateDataKeyPair

서비스 접두사	작업
	kms:GenerateDataKeyPairWithoutPlaintext kms:GenerateDataKeyWithoutPlaintext kms:GenerateMac kms:GenerateRandom kms:GetKeyPolicy kms:GetKeyRotationStatus kms:GetParametersForImport kms:GetPublicKey kms:ImportKeyMaterial kms:ListAliases kms:ListGrants kms:ListKeyPolicies kms:ListKeyRotations kms:ListKeys kms:ListRetirableGrants kms:ReplicateKey kms:RetireGrant kms:RevokeGrant kms:RotateKeyOnDemand kms:ScheduleKeyDeletion kms:Sign

서비스 접두사	작업
	kms:UpdateAlias kms:UpdateCustomKeyStore kms:UpdateKeyDescription kms:UpdatePrimaryRegion kms:Verify kms:VerifyMac

서비스 접두사	작업
lambda	lambda:AddLayerVersionPermission
	lambda:AddLayerVersionPermission
	lambda:AddPermission
	lambda:AddPermission
	lambda:AddPermission
	lambda:CreateAlias
	lambda:CreateAlias
	lambda:CreateCodeSigningConfig
	lambda:CreateEventSourceMapping
	lambda:CreateEventSourceMapping
	lambda:CreateFunction
	lambda:CreateFunction
	lambda:CreateFunctionUrlConfig
	lambda:DeleteAlias
	lambda:DeleteAlias
	lambda:DeleteCodeSigningConfig
	lambda:DeleteEventSourceMapping
	lambda:DeleteEventSourceMapping
	lambda:DeleteFunction
	lambda:DeleteFunction
	lambda:DeleteFunctionCodeSigningConfig

서비스 접두사	작업
	lambda:DeleteFunctionConcurrency
	lambda:DeleteFunctionConcurrency
	lambda:DeleteFunctionEventInvokeConfig
	lambda:DeleteFunctionUrlConfig
	lambda:DeleteLayerVersion
	lambda:DeleteLayerVersion
	lambda:DeleteProvisionedConcurrencyConfig
	lambda:GetAccountSettings
	lambda:GetAccountSettings
	lambda:GetAlias
	lambda:GetAlias
	lambda:GetCodeSigningConfig
	lambda:GetEventSourceMapping
	lambda:GetEventSourceMapping
	lambda:GetFunction
	lambda:GetFunction
	lambda:GetFunction
	lambda:GetFunctionCodeSigningConfig
	lambda:GetFunctionConcurrency
	lambda:GetFunctionConfiguration
	lambda:GetFunctionConfiguration

서비스 접두사	작업
	lambda:GetFunctionConfiguration
	lambda:GetFunctionEventInvokeConfig
	lambda:GetFunctionUrlConfig
	lambda:GetLayerVersion
	lambda:GetLayerVersion
	lambda:GetLayerVersion
	lambda:GetLayerVersion
	lambda:GetLayerVersionPolicy
	lambda:GetLayerVersionPolicy
	lambda:GetPolicy
	lambda:GetPolicy
	lambda:GetPolicy
	lambda:GetProvisionedConcurrencyConfig
	lambda:GetRuntimeManagementConfig
	lambda:ListAliases
	lambda:ListAliases
	lambda:ListCodeSigningConfigs
	lambda:ListEventSourceMappings
	lambda:ListEventSourceMappings
	lambda:ListFunctionEventInvokeConfigs
	lambda:ListFunctions

서비스 접두사	작업
	lambda:ListFunctions
	lambda:ListFunctionsByCodeSigningConfig
	lambda:ListFunctionUrlConfigs
	lambda:ListLayers
	lambda:ListLayers
	lambda:ListLayerVersions
	lambda:ListLayerVersions
	lambda:ListProvisionedConcurrencyConfigs
	lambda:ListVersionsByFunction
	lambda:ListVersionsByFunction
	lambda:PublishLayerVersion
	lambda:PublishLayerVersion
	lambda:PublishVersion
	lambda:PublishVersion
	lambda:PutFunctionCodeSigningConfig
	lambda:PutFunctionConcurrency
	lambda:PutFunctionConcurrency
	lambda:PutFunctionEventInvokeConfig
	lambda:PutProvisionedConcurrencyConfig
	lambda:PutRuntimeManagementConfig
	lambda:RemoveLayerVersionPermission

서비스 접두사	작업
	lambda:RemoveLayerVersionPermission
	lambda:RemovePermission
	lambda:RemovePermission
	lambda:RemovePermission
	lambda:UpdateAlias
	lambda:UpdateAlias
	lambda:UpdateCodeSigningConfig
	lambda:UpdateEventSourceMapping
	lambda:UpdateEventSourceMapping
	lambda:UpdateFunctionCode
	lambda:UpdateFunctionCode
	lambda:UpdateFunctionCode
	lambda:UpdateFunctionConfiguration
	lambda:UpdateFunctionConfiguration
	lambda:UpdateFunctionConfiguration
	lambda:UpdateFunctionEventInvokeConfig
	lambda:UpdateFunctionUrlConfig

서비스 접두사	작업
lex	lex:BatchCreateCustomVocabularyItem lex:BatchDeleteCustomVocabularyItem lex:BatchUpdateCustomVocabularyItem lex:BuildBotLocale lex:CreateBotAlias lex:CreateBotReplica lex:CreateBotVersion lex:CreateExport lex:CreateIntentVersion lex:CreateResourcePolicy lex:CreateSlotTypeVersion lex:CreateTestSetDiscrepancyReport lex:CreateUploadUrl lex:DeleteBot lex:DeleteBotChannelAssociation lex:DeleteBotReplica lex:DeleteExport lex:DeleteImport lex:DeleteIntentVersion lex:DeleteResourcePolicy lex:DeleteSlotTypeVersion

서비스 접두사	작업
	lex:DeleteTestSet lex:DeleteUtterances lex:DescribeBotAlias lex:DescribeBotRecommendation lex:DescribeBotReplica lex:DescribeBotResourceGeneration lex:DescribeBotVersion lex:DescribeCustomVocabularyMetadata lex:DescribeExport lex:DescribeImport lex:DescribeResourcePolicy lex:DescribeTestExecution lex:DescribeTestSet lex:DescribeTestSetDiscrepancyReport lex:DescribeTestSetGeneration lex:GenerateBotElement lex:GetBot lex:GetBotAlias lex:GetBotAliases lex:GetBotChannelAssociation lex:GetBotChannelAssociations

서비스 접두사	작업
	lex:GetBots lex:GetBotVersions lex:GetBuiltinIntent lex:GetBuiltinIntents lex:GetBuiltinSlotTypes lex:GetExport lex:GetImport lex:GetIntent lex:GetIntents lex:GetIntentVersions lex:GetMigration lex:GetMigrations lex:GetSlotType lex:GetSlotTypes lex:GetSlotTypeVersions lex:GetTestExecutionArtifactsUrl lex:GetUtterancesView lex:ListBotAliases lex:ListBotAliasReplicas lex:ListBotRecommendations lex:ListBotReplicas

서비스 접두사	작업
	lex:ListBotResourceGenerations lex:ListBots lex:ListBotVersionReplicas lex:ListBotVersions lex:ListBuiltInIntents lex:ListBuiltInSlotTypes lex:ListCustomVocabularyItems lex:ListExports lex:ListImports lex:ListIntentMetrics lex:ListIntentPaths lex:ListRecommendedIntents lex:ListSessionAnalyticsData lex:ListSessionMetrics lex:ListTestExecutionResultItems lex:ListTestExecutions lex:ListTestSets lex:PutBot lex:PutBotAlias lex:PutIntent lex:PutSlotType

서비스 접두사	작업
	lex:SearchAssociatedTranscripts lex:StartBotRecommendation lex:StartImport lex:StartMigration lex:StartTestExecution lex:StartTestSetGeneration lex:StopBotRecommendation lex:UpdateBotAlias lex:UpdateBotRecommendation lex:UpdateExport lex:UpdateResourcePolicy
license-manager-linux-subscriptions	license-manager-linux-subscriptions:GetServiceSettings license-manager-linux-subscriptions:ListLinuxSubscriptionInstances license-manager-linux-subscriptions:ListLinuxSubscriptions license-manager-linux-subscriptions:UpdateServiceSettings

서비스 접두사	작업
lightsail	lightsail:AllocateStaticIp lightsail:AttachCertificateToDistribution lightsail:AttachDisk lightsail:AttachInstancesToLoadBalancer lightsail:AttachLoadBalancerTlsCertificate lightsail:AttachStaticIp lightsail:CloseInstancePublicPorts lightsail:CopySnapshot lightsail:CreateBucket lightsail:CreateBucketAccessKey lightsail:CreateCertificate lightsail:CreateCloudFormationStack lightsail:CreateContactMethod lightsail:CreateContainerService lightsail:CreateContainerServiceDeployment lightsail:CreateContainerServiceRegistryLogin lightsail:CreateDisk lightsail:CreateDiskFromSnapshot lightsail:CreateDiskSnapshot lightsail:CreateDistribution lightsail:CreateDomain

서비스 접두사	작업
	lightsail:CreateGUISessionAccessDetails lightsail:CreateInstances lightsail:CreateInstancesFromSnapshot lightsail:CreateInstanceSnapshot lightsail:CreateKeyPair lightsail:CreateLoadBalancer lightsail:CreateLoadBalancerTlsCertificate lightsail:CreateRelationalDatabase lightsail:CreateRelationalDatabaseFromSnapshot lightsail:CreateRelationalDatabaseSnapshot lightsail:DeleteAlarm lightsail:DeleteAutoSnapshot lightsail:DeleteBucket lightsail:DeleteBucketAccessKey lightsail:DeleteCertificate lightsail:DeleteContactMethod lightsail:DeleteContainerImage lightsail:DeleteContainerService lightsail:DeleteDisk lightsail:DeleteDiskSnapshot lightsail:DeleteDistribution

서비스 접두사	작업
	lightsail:DeleteDomain lightsail:DeleteDomainEntry lightsail:DeleteInstance lightsail:DeleteInstanceSnapshot lightsail:DeleteKeyPair lightsail:DeleteKnownHostKeys lightsail:DeleteLoadBalancer lightsail:DeleteLoadBalancerTlsCertificate lightsail:DeleteRelationalDatabase lightsail:DeleteRelationalDatabaseSnapshot lightsail:DetachCertificateFromDistribution lightsail:DetachDisk lightsail:DetachInstancesFromLoadBalancer lightsail:DetachStaticIp lightsail:DisableAddOn lightsail:DownloadDefaultKeyPair lightsail:EnableAddOn lightsail:ExportSnapshot lightsail:GetActiveNames lightsail:GetAlarms lightsail:GetAutoSnapshots

서비스 접두사	작업
	lightsail:GetBlueprints lightsail:GetBucketAccessKeys lightsail:GetBucketBundles lightsail:GetBucketMetricData lightsail:GetBuckets lightsail:GetBundles lightsail:GetCertificates lightsail:GetCloudFormationStackRecords lightsail:GetContactMethods lightsail:GetContainerAPIMetadata lightsail:GetContainerImages lightsail:GetContainerLog lightsail:GetContainerServiceDeployments lightsail:GetContainerServiceMetricData lightsail:GetContainerServicePowers lightsail:GetContainerServices lightsail:GetCostEstimate lightsail:GetDisk lightsail:GetDisks lightsail:GetDiskSnapshot lightsail:GetDiskSnapshots

서비스 접두사	작업
	lightsail:GetDistributionBundles lightsail:GetDistributionLatestCacheReset lightsail:GetDistributionMetricData lightsail:GetDistributions lightsail:GetDomain lightsail:GetExportSnapshotRecords lightsail:GetInstance lightsail:GetInstanceMetricData lightsail:GetInstancePortStates lightsail:GetInstances lightsail:GetInstanceSnapshot lightsail:GetInstanceSnapshots lightsail:GetInstanceState lightsail:GetKeyPair lightsail:GetKeyPairs lightsail:GetLoadBalancer lightsail:GetLoadBalancerMetricData lightsail:GetLoadBalancers lightsail:GetLoadBalancerTlsCertificates lightsail:GetLoadBalancerTlsPolicies lightsail:GetOperation

서비스 접두사	작업
	lightsail:GetOperations lightsail:GetOperationsForResource lightsail:GetRegions lightsail:GetRelationalDatabase lightsail:GetRelationalDatabaseBlueprints lightsail:GetRelationalDatabaseBundles lightsail:GetRelationalDatabaseEvents lightsail:GetRelationalDatabaseLogEvents lightsail:GetRelationalDatabaseLogStreams lightsail:GetRelationalDatabaseMasterUserPassword lightsail:GetRelationalDatabaseMetricData lightsail:GetRelationalDatabaseParameters lightsail:GetRelationalDatabases lightsail:GetRelationalDatabaseSnapshot lightsail:GetRelationalDatabaseSnapshots lightsail:GetSetupHistory lightsail:GetStaticIp lightsail:GetStaticIps lightsail:ImportKeyPair lightsail:IsVpcPeered lightsail:OpenInstancePublicPorts

서비스 접두사	작업
	lightsail:PeerVpc lightsail:PutAlarm lightsail:PutInstancePublicPorts lightsail:RebootInstance lightsail:RebootRelationalDatabase lightsail:RegisterContainerImage lightsail:ReleaseStaticIp lightsail:ResetDistributionCache lightsail:SendContactMethodVerification lightsail:SetIpAddressType lightsail:SetResourceAccessForBucket lightsail:SetupInstanceHttps lightsail:StartGUISession lightsail:StartInstance lightsail:StartRelationalDatabase lightsail:StopGUISession lightsail:StopInstance lightsail:StopRelationalDatabase lightsail:TestAlarm lightsail:UnpeerVpc lightsail:UpdateBucket

서비스 접두사	작업
	lightsail:UpdateBucketBundle lightsail:UpdateContainerService lightsail:UpdateDistribution lightsail:UpdateDistributionBundle lightsail:UpdateDomainEntry lightsail:UpdateInstanceMetadataOptions lightsail:UpdateLoadBalancerAttribute lightsail:UpdateRelationalDatabase lightsail:UpdateRelationalDatabaseParameters

서비스 접두사	작업
로그	logs:AssociateKmsKey
	logs:CancelExportTask
	logs:CreateExportTask
	logs:CreateLogAnomalyDetector
	logs:CreateLogGroup
	logs:CreateLogStream
	logs:DeleteDataProtectionPolicy
	logs:DeleteDelivery
	logs:DeleteDeliveryDestination
	logs:DeleteDeliveryDestinationPolicy
	logs:DeleteDeliverySource
	logs:DeleteDestination
	logs:DeleteLogGroup
	logs:DeleteLogStream
	logs:DeleteMetricFilter
	logs:DeleteQueryDefinition
	logs:DeleteResourcePolicy
	logs:DeleteRetentionPolicy
	logs:DeleteSubscriptionFilter
	logs:DescribeAccountPolicies
	logs:DescribeDeliveries

서비스 접두사	작업
	logs:DescribeDeliveryDestinations
	logs:DescribeDeliverySources
	logs:DescribeDestinations
	logs:DescribeExportTasks
	logs:DescribeLogGroups
	logs:DescribeLogStreams
	logs:DescribeMetricFilters
	logs:DescribeQueries
	logs:DescribeQueryDefinitions
	logs:DescribeResourcePolicies
	logs:DescribeSubscriptionFilters
	logs:DisassociateKmsKey
	logs:GetDataProtectionPolicy
	logs:GetDelivery
	logs:GetDeliveryDestination
	logs:GetDeliveryDestinationPolicy
	logs:GetDeliverySource
	logs:GetLogGroupFields
	logs:GetLogRecord
	logs:GetQueryResults
	logs:ListAnomalies

서비스 접두사	작업
	logs:ListLogAnomalyDetectors
	logs:PutDataProtectionPolicy
	logs:PutDeliveryDestination
	logs:PutDeliveryDestinationPolicy
	logs:PutDeliverySource
	logs:PutDestination
	logs:PutDestinationPolicy
	logs:PutMetricFilter
	logs:PutQueryDefinition
	logs:PutResourcePolicy
	logs:PutRetentionPolicy
	logs:PutSubscriptionFilter
	logs:StartLiveTail
	logs:StartQuery
	logs:StopQuery
	logs:TestMetricFilter

서비스 접두사	작업
lookoutequipment	lookoutequipment:CreateDataset lookoutequipment:CreateInferenceScheduler lookoutequipment:CreateLabel lookoutequipment:CreateLabelGroup lookoutequipment:CreateModel lookoutequipment:DeleteDataset lookoutequipment:DeleteInferenceScheduler lookoutequipment:DeleteLabel lookoutequipment:DeleteLabelGroup lookoutequipment:DeleteModel lookoutequipment:DeleteResourcePolicy lookoutequipment:DeleteRetrainingScheduler lookoutequipment:DescribeDataIngestionJob lookoutequipment:DescribeDataset lookoutequipment:DescribeInferenceScheduler lookoutequipment:Describelabel lookoutequipment:DescribeLabelGroup lookoutequipment:DescribeModel lookoutequipment:DescribeModelVersion lookoutequipment:DescribeResourcePolicy lookoutequipment:DescribeRetrainingScheduler

서비스 접두사	작업
	lookoutequipment:ImportDataset lookoutequipment:ImportModelVersion lookoutequipment:ListDataIngestionJobs lookoutequipment:ListDatasets lookoutequipment:ListInferenceEvents lookoutequipment:ListInferenceExecutions lookoutequipment:ListInferenceSchedulers lookoutequipment:ListLabelGroups lookoutequipment:ListLabels lookoutequipment:ListModels lookoutequipment:ListModelVersions lookoutequipment:ListRetrainingSchedulers lookoutequipment:ListSensorStatistics lookoutequipment:PutResourcePolicy lookoutequipment:StartDataIngestionJob lookoutequipment:StartInferenceScheduler lookoutequipment:StartRetrainingScheduler lookoutequipment:StopInferenceScheduler lookoutequipment:StopRetrainingScheduler lookoutequipment:UpdateActiveModelVersion lookoutequipment:UpdateInferenceScheduler

서비스 접두사	작업
	lookoutequipment:UpdateLabelGroup lookoutequipment:UpdateModel lookoutequipment:UpdateRetrainingScheduler

서비스 접두사	작업
lookoutmetrics	lookoutmetrics:ActivateAnomalyDetector
	lookoutmetrics:BackTestAnomalyDetector
	lookoutmetrics:CreateAlert
	lookoutmetrics:CreateAnomalyDetector
	lookoutmetrics:CreateMetricSet
	lookoutmetrics:DeactivateAnomalyDetector
	lookoutmetrics:DeleteAlert
	lookoutmetrics:DeleteAnomalyDetector
	lookoutmetrics:DescribeAlert
	lookoutmetrics:DescribeAnomalyDetectionExecutions
	lookoutmetrics:DescribeAnomalyDetector
	lookoutmetrics:DescribeMetricSet
	lookoutmetrics:DetectMetricSetConfig
	lookoutmetrics:GetAnomalyGroup
	lookoutmetrics:GetDataQualityMetrics
	lookoutmetrics:GetFeedback
	lookoutmetrics:GetSampleData
	lookoutmetrics:ListAlerts
	lookoutmetrics:ListAnomalyDetectors
	lookoutmetrics:ListAnomalyGroupRelatedMetrics
	lookoutmetrics:ListAnomalyGroupSummaries

서비스 접두사	작업
	lookoutmetrics:ListAnomalyGroupTimeSeries lookoutmetrics:ListMetricSets lookoutmetrics:PutFeedback lookoutmetrics:UpdateAlert lookoutmetrics:UpdateAnomalyDetector lookoutmetrics:UpdateMetricSet

서비스 접두사	작업
lookoutvision	lookoutvision:CreateDataset lookoutvision:CreateModel lookoutvision:CreateProject lookoutvision:DeleteDataset lookoutvision:DeleteModel lookoutvision:DeleteProject lookoutvision:DescribeDataset lookoutvision:DescribeModel lookoutvision:DescribeModelPackagingJob lookoutvision:DescribeProject lookoutvision:DetectAnomalies lookoutvision:ListDatasetEntries lookoutvision:ListModelPackagingJobs lookoutvision:ListModels lookoutvision:ListProjects lookoutvision:StartModel lookoutvision:StartModelPackagingJob lookoutvision:StopModel lookoutvision:UpdateDatasetEntries

서비스 접두사	작업
m2	m2:CancelBatchJobExecution
	m2:CreateApplication
	m2:CreateDataSetImportTask
	m2:CreateDeployment
	m2:CreateEnvironment
	m2:DeleteApplication
	m2:DeleteApplicationFromEnvironment
	m2:DeleteEnvironment
	m2:GetApplication
	m2:GetApplicationVersion
	m2:GetBatchJobExecution
	m2:GetDataSetDetails
	m2:GetDataSetImportTask
	m2:GetDeployment
	m2:GetEnvironment
	m2:GetSignedBluinsightsUrl
	m2:ListApplications
	m2:ListApplicationVersions
	m2:ListBatchJobDefinitions
	m2:ListBatchJobExecutions
	m2:ListBatchJobRestartPoints

서비스 접두사	작업
	m2:ListDataSetImportHistory
	m2:ListDataSets
	m2:ListDeployments
	m2:ListEngineVersions
	m2:ListEnvironments
	m2:StartApplication
	m2:StartBatchJob
	m2:StopApplication
	m2:UpdateApplication
	m2:UpdateEnvironment

서비스 접두사	작업
managedblockchain	managedblockchain:CreateAccessor managedblockchain:CreateMember managedblockchain:CreateNetwork managedblockchain:CreateNode managedblockchain:CreateProposal managedblockchain:DeleteAccessor managedblockchain:DeleteMember managedblockchain:DeleteNode managedblockchain:GetAccessor managedblockchain:GetMember managedblockchain:GetNetwork managedblockchain:GetNode managedblockchain:GetProposal managedblockchain:InvokeRpcPolygonMainnet managedblockchain:InvokeRpcPolygonMumbaiTestnet managedblockchain:ListAccessors managedblockchain:ListInvitations managedblockchain:ListMembers managedblockchain:ListNetworks managedblockchain:ListNodes managedblockchain:ListProposals

서비스 접두사	작업
	managedblockchain:ListProposalVotes managedblockchain:RejectInvitation managedblockchain:UpdateMember managedblockchain:UpdateNode managedblockchain:VoteOnProposal

서비스 접두사	작업
mediaconnect	mediaconnect:AddBridgeOutputs
	mediaconnect:AddBridgeSources
	mediaconnect:AddFlowMediaStreams
	mediaconnect:AddFlowOutputs
	mediaconnect:AddFlowSources
	mediaconnect:AddFlowVpcInterfaces
	mediaconnect:CreateBridge
	mediaconnect:CreateFlow
	mediaconnect:CreateGateway
	mediaconnect:DeleteBridge
	mediaconnect:DeleteFlow
	mediaconnect:DeleteGateway
	mediaconnect:DeregisterGatewayInstance
	mediaconnect:DescribeBridge
	mediaconnect:DescribeFlow
	mediaconnect:DescribeFlowSourceMetadata
	mediaconnect:DescribeGateway
	mediaconnect:DescribeGatewayInstance
	mediaconnect:DescribeOffering
	mediaconnect:DescribeReservation
	mediaconnect:GrantFlowEntitlements

서비스 접두사	작업
	mediaconnect:ListBridges
	mediaconnect:ListEntitlements
	mediaconnect:ListFlows
	mediaconnect:ListGatewayInstances
	mediaconnect:ListGateways
	mediaconnect:ListOfferings
	mediaconnect:ListReservations
	mediaconnect:PurchaseOffering
	mediaconnect:RemoveBridgeOutput
	mediaconnect:RemoveBridgeSource
	mediaconnect:RemoveFlowMediaStream
	mediaconnect:RemoveFlowOutput
	mediaconnect:RemoveFlowSource
	mediaconnect:RemoveFlowVpcInterface
	mediaconnect:RevokeFlowEntitlement
	mediaconnect:StartFlow
	mediaconnect:StopFlow
	mediaconnect:UpdateBridge
	mediaconnect:UpdateBridgeOutput
	mediaconnect:UpdateBridgeSource
	mediaconnect:UpdateBridgeState

서비스 접두사	작업
	mediaconnect:UpdateFlow mediaconnect:UpdateFlowEntitlement mediaconnect:UpdateFlowMediaStream mediaconnect:UpdateFlowOutput mediaconnect:UpdateFlowSource mediaconnect:UpdateGatewayInstance

서비스 접두사	작업
mediaconvert	mediaconvert:AssociateCertificate mediaconvert:CancelJob mediaconvert:CreateJob mediaconvert:CreateJobTemplate mediaconvert:CreatePreset mediaconvert:CreateQueue mediaconvert:DeleteJobTemplate mediaconvert:DeletePolicy mediaconvert:DeletePreset mediaconvert:DeleteQueue mediaconvert:DescribeEndpoints mediaconvert:DisassociateCertificate mediaconvert:GetJob mediaconvert:GetJobTemplate mediaconvert:GetPolicy mediaconvert:GetPreset mediaconvert:GetQueue mediaconvert:ListJobs mediaconvert:ListJobTemplates mediaconvert:ListPresets mediaconvert:ListQueues

서비스 접두사	작업
	mediaconvert:PutPolicy mediaconvert:UpdateJobTemplate mediaconvert:UpdatePreset mediaconvert:UpdateQueue

서비스 접두사	작업
medialive	medialive:AcceptInputDeviceTransfer
	medialive:BatchDelete
	medialive:BatchStart
	medialive:BatchStop
	medialive:BatchUpdateSchedule
	medialive:CancelInputDeviceTransfer
	medialive:ClaimDevice
	medialive:CreateChannel
	medialive:CreateCloudWatchAlarmTemplate
	medialive:CreateCloudWatchAlarmTemplateGroup
	medialive:CreateEventBridgeRuleTemplate
	medialive:CreateEventBridgeRuleTemplateGroup
	medialive:CreateInput
	medialive:CreateInputSecurityGroup
	medialive:CreateMultiplex
	medialive:CreateMultiplexProgram
	medialive:CreatePartnerInput
	medialive:CreateSignalMap
	medialive:DeleteChannel
	medialive:DeleteCloudWatchAlarmTemplate
	medialive:DeleteCloudWatchAlarmTemplateGroup

서비스 접두사	작업
	medialive:DeleteEventBridgeRuleTemplate
	medialive:DeleteEventBridgeRuleTemplateGroup
	medialive:DeleteInput
	medialive:DeleteInputSecurityGroup
	medialive:DeleteMultiplex
	medialive:DeleteMultiplexProgram
	medialive:DeleteReservation
	medialive:DeleteSchedule
	medialive:DeleteSignalMap
	medialive:DescribeAccountConfiguration
	medialive:DescribeChannel
	medialive:DescribeInput
	medialive:DescribeInputDevice
	medialive:DescribeInputDeviceThumbnail
	medialive:DescribeInputSecurityGroup
	medialive:DescribeMultiplex
	medialive:DescribeMultiplexProgram
	medialive:DescribeOffering
	medialive:DescribeReservation
	medialive:DescribeSchedule
	medialive:DescribeThumbnails

서비스 접두사	작업
	medialive:GetCloudWatchAlarmTemplate
	medialive:GetCloudWatchAlarmTemplateGroup
	medialive:GetEventBridgeRuleTemplate
	medialive:GetEventBridgeRuleTemplateGroup
	medialive:GetSignalMap
	medialive:ListChannels
	medialive:ListCloudWatchAlarmTemplateGroups
	medialive:ListCloudWatchAlarmTemplates
	medialive:ListEventBridgeRuleTemplateGroups
	medialive:ListEventBridgeRuleTemplates
	medialive:ListInputDevices
	medialive:ListInputDeviceTransfers
	medialive:ListInputs
	medialive:ListInputSecurityGroups
	medialive:ListMultiplexes
	medialive:ListMultiplexPrograms
	medialive:ListOfferings
	medialive:ListReservations
	medialive:ListSignalMaps
	medialive:PurchaseOffering
	medialive:RebootInputDevice

서비스 접두사	작업
	medialive:RejectInputDeviceTransfer
	medialive:RestartChannelPipelines
	medialive:StartChannel
	medialive:StartDeleteMonitorDeployment
	medialive:StartInputDevice
	medialive:StartInputDeviceMaintenanceWindow
	medialive:StartMonitorDeployment
	medialive:StartMultiplex
	medialive:StartUpdateSignalMap
	medialive:StopChannel
	medialive:StopInputDevice
	medialive:StopMultiplex
	medialive:TransferInputDevice
	medialive:UpdateAccountConfiguration
	medialive:UpdateChannel
	medialive:UpdateChannelClass
	medialive:UpdateCloudWatchAlarmTemplate
	medialive:UpdateCloudWatchAlarmTemplateGroup
	medialive:UpdateEventBridgeRuleTemplate
	medialive:UpdateEventBridgeRuleTemplateGroup
	medialive:UpdateInput

서비스 접두사	작업
	medialive:UpdateInputDevice medialive:UpdateInputSecurityGroup medialive:UpdateMultiplex medialive:UpdateMultiplexProgram medialive:UpdateReservation

서비스 접두사	작업
mediastore	mediastore:CreateContainer mediastore:DeleteContainer mediastore:DeleteContainerPolicy mediastore:DeleteCorsPolicy mediastore:DeleteLifecyclePolicy mediastore:DeleteMetricPolicy mediastore:DescribeContainer mediastore:GetContainerPolicy mediastore:GetCorsPolicy mediastore:GetLifecyclePolicy mediastore:GetMetricPolicy mediastore:ListContainers mediastore:PutContainerPolicy mediastore:PutCorsPolicy mediastore:PutLifecyclePolicy mediastore:PutMetricPolicy mediastore:StartAccessLogging mediastore:StopAccessLogging

서비스 접두사	작업
mediatailor	mediatailor:ConfigureLogsForPlaybackConfiguration
	mediatailor:CreateChannel
	mediatailor:CreateLiveSource
	mediatailor:CreatePrefetchSchedule
	mediatailor:CreateProgram
	mediatailor:CreateSourceLocation
	mediatailor:CreateVodSource
	mediatailor:DeleteChannel
	mediatailor:DeleteChannelPolicy
	mediatailor:DeleteLiveSource
	mediatailor:DeletePlaybackConfiguration
	mediatailor:DeletePrefetchSchedule
	mediatailor:DeleteProgram
	mediatailor:DeleteSourceLocation
	mediatailor:DeleteVodSource
	mediatailor:DescribeChannel
	mediatailor:DescribeLiveSource
	mediatailor:DescribeProgram
	mediatailor:DescribeSourceLocation
	mediatailor:DescribeVodSource
	mediatailor:GetChannelPolicy

서비스 접두사	작업
	mediatailor:GetChannelSchedule
	mediatailor:GetPlaybackConfiguration
	mediatailor:GetPrefetchSchedule
	mediatailor:ListAlerts
	mediatailor:ListChannels
	mediatailor:ListLiveSources
	mediatailor:ListPlaybackConfigurations
	mediatailor:ListPrefetchSchedules
	mediatailor:ListSourceLocations
	mediatailor:ListVodSources
	mediatailor:PutChannelPolicy
	mediatailor:PutPlaybackConfiguration
	mediatailor:StartChannel
	mediatailor:StopChannel
	mediatailor:UpdateChannel
	mediatailor:UpdateLiveSource
	mediatailor:UpdateProgram
	mediatailor:UpdateSourceLocation
	mediatailor:UpdateVodSource

서비스 접두사	작업
memorydb	memorydb:BatchUpdateCluster memorydb:CopySnapshot memorydb:CreateAcl memorydb:CreateCluster memorydb:CreateParameterGroup memorydb:CreateSnapshot memorydb:CreateSubnetGroup memorydb:CreateUser memorydb:DeleteAcl memorydb:DeleteCluster memorydb:DeleteParameterGroup memorydb:DeleteSnapshot memorydb:DeleteSubnetGroup memorydb:DeleteUser memorydb:DescribeAcls memorydb:DescribeClusters memorydb:DescribeEngineVersions memorydb:DescribeEvents memorydb:DescribeParameterGroups memorydb:DescribeParameters memorydb:DescribeReservedNodes

서비스 접두사	작업
	memorydb:DescribeReservedNodesOfferings memorydb:DescribeServiceUpdates memorydb:DescribeSnapshots memorydb:DescribeSubnetGroups memorydb:DescribeUsers memorydb:FailoverShard memorydb:ListAllowedNodeTypeUpdates memorydb:PurchaseReservedNodesOffering memorydb:ResetParameterGroup memorydb:UpdateAcl memorydb:UpdateCluster memorydb:UpdateParameterGroup memorydb:UpdateSubnetGroup memorydb:UpdateUser

서비스 접두사	작업
mgh	mgh:AssociateCreatedArtifact
	mgh:AssociateDiscoveredResource
	mgh:CreateHomeRegionControl
	mgh:CreateProgressUpdateStream
	mgh:DeleteHomeRegionControl
	mgh:DeleteProgressUpdateStream
	mgh:DescribeApplicationState
	mgh:DescribeHomeRegionControls
	mgh:DescribeMigrationTask
	mgh:DisassociateCreatedArtifact
	mgh:DisassociateDiscoveredResource
	mgh:GetHomeRegion
	mgh:ImportMigrationTask
	mgh:ListApplicationStates
	mgh:ListCreatedArtifacts
	mgh:ListDiscoveredResources
	mgh:ListMigrationTasks
	mgh:ListProgressUpdateStreams
	mgh:NotifyApplicationState
	mgh:NotifyMigrationTaskState
	mgh:PutResourceAttributes

서비스 접두사	작업
mgn	mgn:ArchiveApplication mgn:ArchiveWave mgn:AssociateApplications mgn:AssociateSourceServers mgn:ChangeServerLifeCycleState mgn:CreateApplication mgn:CreateConnector mgn:CreateLaunchConfigurationTemplate mgn:CreateReplicationConfigurationTemplate mgn:CreateWave mgn:DeleteApplication mgn:DeleteConnector mgn:DeleteJob mgn:DeleteLaunchConfigurationTemplate mgn:DeleteReplicationConfigurationTemplate mgn:DeleteSourceServer mgn:DeleteVcenterClient mgn:DeleteWave mgn:DescribeJobLogItems mgn:DescribeJobs mgn:DescribeLaunchConfigurationTemplates

서비스 접두사	작업
	mgn:DescribeReplicationConfigurationTemplates mgn:DescribeVcenterClients mgn:DisassociateApplications mgn:DisassociateSourceServers mgn:DisconnectFromService mgn:FinalizeCutover mgn:GetReplicationConfiguration mgn:InitializeService mgn:ListConnectors mgn:ListExportErrors mgn:ListExports mgn:ListImportErrors mgn:ListImports mgn:ListManagedAccounts mgn:ListSourceServerActions mgn:ListTemplateActions mgn:MarkAsArchived mgn:PauseReplication mgn:PutSourceServerAction mgn:PutTemplateAction mgn:RemoveSourceServerAction

서비스 접두사	작업
	mgn:RemoveTemplateAction mgn:ResumeReplication mgn:RetryDataReplication mgn:StartCutover mgn:StartExport mgn:StartImport mgn:StartReplication mgn:StartTest mgn:StopReplication mgn:TerminateTargetInstances mgn:UnarchiveApplication mgn:UnarchiveWave mgn:UpdateApplication mgn:UpdateConnector mgn:UpdateLaunchConfigurationTemplate mgn:UpdateReplicationConfiguration mgn:UpdateReplicationConfigurationTemplate mgn:UpdateSourceServer mgn:UpdateSourceServerReplicationType mgn:UpdateWave

서비스 접두사	작업
migrationhub-strategy	migrationhub-strategy:GetAntiPattern
	migrationhub-strategy:GetApplicationComponentDetails
	migrationhub-strategy:GetApplicationComponentStrategies
	migrationhub-strategy:GetAssessment
	migrationhub-strategy:GetImportFileTask
	migrationhub-strategy:GetLatestAssessmentId
	migrationhub-strategy:GetMessage
	migrationhub-strategy:GetPortfolioPreferences
	migrationhub-strategy:GetPortfolioSummary
	migrationhub-strategy:GetRecommendationReportDetails
	migrationhub-strategy:GetServerDetails
	migrationhub-strategy:GetServerStrategies
	migrationhub-strategy:ListAnalyzableServers
	migrationhub-strategy:ListAntiPatterns
	migrationhub-strategy:ListApplicationComponents
	migrationhub-strategy:ListCollectors
	migrationhub-strategy:ListImportFileTask
	migrationhub-strategy:ListJarArtifacts
	migrationhub-strategy:ListServers
	migrationhub-strategy:PutLogData
	migrationhub-strategy:PutMetricData

서비스 접두사	작업
	migrationhub-strategy:PutPortfolioPreferences
	migrationhub-strategy:RegisterCollector
	migrationhub-strategy:SendMessage
	migrationhub-strategy:StartAssessment
	migrationhub-strategy:StartImportFileTask
	migrationhub-strategy:StartRecommendationReportGeneration
	migrationhub-strategy:StopAssessment
	migrationhub-strategy:UpdateApplicationComponentConfig
	migrationhub-strategy:UpdateCollectorConfiguration
	migrationhub-strategy:UpdateServerConfig

서비스 접두사	작업
mobiletargeting	mobiletargeting:CreateApp mobiletargeting:CreateCampaign mobiletargeting:CreateEmailTemplate mobiletargeting:CreateExportJob mobiletargeting:CreateImportJob mobiletargeting:CreateInAppTemplate mobiletargeting:CreateJourney mobiletargeting:CreatePushTemplate mobiletargeting:CreateRecommenderConfiguration mobiletargeting:CreateSegment mobiletargeting:CreateSmsTemplate mobiletargeting:CreateVoiceTemplate mobiletargeting:DeleteAdmChannel mobiletargeting:DeleteApnsChannel mobiletargeting:DeleteApnsSandboxChannel mobiletargeting:DeleteApnsVoipChannel mobiletargeting:DeleteApnsVoipSandboxChannel mobiletargeting:DeleteApp mobiletargeting:DeleteBaiduChannel mobiletargeting:DeleteCampaign mobiletargeting:DeleteEmailChannel

서비스 접두사	작업
	mobiletargeting:DeleteEmailTemplate mobiletargeting:DeleteEndpoint mobiletargeting:DeleteEventStream mobiletargeting:DeleteGcmChannel mobiletargeting:DeleteInAppTemplate mobiletargeting:DeleteJourney mobiletargeting:DeletePushTemplate mobiletargeting:DeleteRecommenderConfiguration mobiletargeting:DeleteSegment mobiletargeting:DeleteSmsChannel mobiletargeting:DeleteSmsTemplate mobiletargeting:DeleteUserEndpoints mobiletargeting:DeleteVoiceChannel mobiletargeting:DeleteVoiceTemplate mobiletargeting:GetAdmChannel mobiletargeting:GetApnsChannel mobiletargeting:GetApnsSandboxChannel mobiletargeting:GetApnsVoipChannel mobiletargeting:GetApnsVoipSandboxChannel mobiletargeting:GetApp mobiletargeting:GetApplicationDateRangeKpi

서비스 접두사	작업
	mobiletargeting:GetApplicationSettings mobiletargeting:GetApps mobiletargeting:GetBaiduChannel mobiletargeting:GetCampaign mobiletargeting:GetCampaignActivities mobiletargeting:GetCampaignDateRangeKpi mobiletargeting:GetCampaigns mobiletargeting:GetCampaignVersion mobiletargeting:GetCampaignVersions mobiletargeting:GetChannels mobiletargeting:GetEmailChannel mobiletargeting:GetEmailTemplate mobiletargeting:GetEndpoint mobiletargeting:GetEventStream mobiletargeting:GetExportJob mobiletargeting:GetExportJobs mobiletargeting:GetGcmChannel mobiletargeting:GetImportJob mobiletargeting:GetImportJobs mobiletargeting:GetInAppMessages mobiletargeting:GetInAppTemplate

서비스 접두사	작업
	mobiletargeting:GetJourney mobiletargeting:GetJourneyDateRangeKpi mobiletargeting:GetJourneyExecutionActivityMetrics mobiletargeting:GetJourneyExecutionMetrics mobiletargeting:GetJourneyRunExecutionActivityMetrics mobiletargeting:GetJourneyRunExecutionMetrics mobiletargeting:GetJourneyRuns mobiletargeting:GetPushTemplate mobiletargeting:GetRecommenderConfiguration mobiletargeting:GetRecommenderConfigurations mobiletargeting:GetSegment mobiletargeting:GetSegmentExportJobs mobiletargeting:GetSegmentImportJobs mobiletargeting:GetSegments mobiletargeting:GetSegmentVersion mobiletargeting:GetSegmentVersions mobiletargeting:GetSmsChannel mobiletargeting:GetSmsTemplate mobiletargeting:GetUserEndpoints mobiletargeting:GetVoiceChannel mobiletargeting:GetVoiceTemplate

서비스 접두사	작업
	mobiletargeting:ListJourneys mobiletargeting:ListTemplates mobiletargeting:ListTemplateVersions mobiletargeting:PhoneNumberValidate mobiletargeting:PutEventStream mobiletargeting:RemoveAttributes mobiletargeting:UpdateAdmChannel mobiletargeting:UpdateApnsChannel mobiletargeting:UpdateApnsSandboxChannel mobiletargeting:UpdateApnsVoipChannel mobiletargeting:UpdateApnsVoipSandboxChannel mobiletargeting:UpdateApplicationSettings mobiletargeting:UpdateBaiduChannel mobiletargeting:UpdateCampaign mobiletargeting:UpdateEmailChannel mobiletargeting:UpdateEmailTemplate mobiletargeting:UpdateEndpoint mobiletargeting:UpdateEndpointsBatch mobiletargeting:UpdateGcmChannel mobiletargeting:UpdateInAppTemplate mobiletargeting:UpdateJourney

서비스 접두사	작업
	mobiletargeting:UpdateJourneyState mobiletargeting:UpdatePushTemplate mobiletargeting:UpdateRecommenderConfiguration mobiletargeting:UpdateSegment mobiletargeting:UpdateSmsChannel mobiletargeting:UpdateSmsTemplate mobiletargeting:UpdateTemplateActiveVersion mobiletargeting:UpdateVoiceChannel mobiletargeting:UpdateVoiceTemplate mobiletargeting:VerifyOTPMessage

서비스 접두사	작업
mq	mq:CreateBroker mq:CreateConfiguration mq:CreateUser mq:DeleteBroker mq:DeleteUser mq:DescribeBroker mq:DescribeBrokerEngineTypes mq:DescribeBrokerInstanceOptions mq:DescribeConfiguration mq:DescribeConfigurationRevision mq:DescribeUser mq:ListBrokers mq:ListConfigurationRevisions mq:ListConfigurations mq:ListUsers mq:Promote mq:RebootBroker mq:UpdateBroker mq:UpdateConfiguration mq:UpdateUser

서비스 접두사	작업
networkmanager	networkmanager:AcceptAttachment networkmanager:AssociateConnectPeer networkmanager:AssociateCustomerGateway networkmanager:AssociateLink networkmanager:AssociateTransitGatewayConnectPeer networkmanager:CreateConnectAttachment networkmanager:CreateConnection networkmanager:CreateConnectPeer networkmanager:CreateCoreNetwork networkmanager:CreateDevice networkmanager:CreateGlobalNetwork networkmanager:CreateLink networkmanager:CreateSite networkmanager:CreateSiteToSiteVpnAttachment networkmanager:CreateTransitGatewayPeering networkmanager:CreateTransitGatewayRouteTableAttachment networkmanager:CreateVpcAttachment networkmanager:DeleteAttachment networkmanager:DeleteConnection networkmanager:DeleteConnectPeer networkmanager:DeleteCoreNetwork

서비스 접두사	작업
	networkmanager:DeleteCoreNetworkPolicyVersion networkmanager:DeleteDevice networkmanager:DeleteGlobalNetwork networkmanager:DeleteLink networkmanager:DeletePeering networkmanager:DeleteResourcePolicy networkmanager:DeleteSite networkmanager:DeregisterTransitGateway networkmanager:DescribeGlobalNetworks networkmanager:DisassociateConnectPeer networkmanager:DisassociateCustomerGateway networkmanager:DisassociateLink networkmanager:DisassociateTransitGatewayConnectPeer networkmanager:ExecuteCoreNetworkChangeSet networkmanager:GetConnectAttachment networkmanager:GetConnections networkmanager:GetConnectPeer networkmanager:GetConnectPeerAssociations networkmanager:GetCoreNetwork networkmanager:GetCoreNetworkChangeEvents networkmanager:GetCoreNetworkChangeSet

서비스 접두사	작업
	networkmanager:GetCoreNetworkPolicy networkmanager:GetCustomerGatewayAssociations networkmanager:GetDevices networkmanager:GetLinkAssociations networkmanager:GetLinks networkmanager:GetNetworkResourceCounts networkmanager:GetNetworkResourceRelationships networkmanager:GetNetworkResources networkmanager:GetNetworkRoutes networkmanager:GetNetworkTelemetry networkmanager:GetResourcePolicy networkmanager:GetRouteAnalysis networkmanager:GetSites networkmanager:GetSiteToSiteVpnAttachment networkmanager:GetTransitGatewayConnectPeerAssociations networkmanager:GetTransitGatewayPeering networkmanager:GetTransitGatewayRegistrations networkmanager:GetTransitGatewayRouteTableAttachment networkmanager:GetVpcAttachment networkmanager:ListAttachments networkmanager:ListConnectPeers

서비스 접두사	작업
	networkmanager:ListCoreNetworkPolicyVersions networkmanager:ListCoreNetworks networkmanager:ListOrganizationServiceAccessStatus networkmanager:ListPeerings networkmanager:PutCoreNetworkPolicy networkmanager:PutResourcePolicy networkmanager:RegisterTransitGateway networkmanager:RejectAttachment networkmanager:RestoreCoreNetworkPolicyVersion networkmanager:StartOrganizationServiceAccessUpdate networkmanager:StartRouteAnalysis networkmanager:UpdateConnection networkmanager:UpdateCoreNetwork networkmanager:UpdateDevice networkmanager:UpdateGlobalNetwork networkmanager:UpdateLink networkmanager:UpdateNetworkResourceMetadata networkmanager:UpdateSite networkmanager:UpdateVpcAttachment

서비스 접두사	작업
nimble	nimble:AcceptEulas
	nimble:CreateLaunchProfile
	nimble:CreateStreamingImage
	nimble:CreateStreamingSession
	nimble:CreateStreamingSessionStream
	nimble:CreateStudio
	nimble:CreateStudioComponent
	nimble:DeleteLaunchProfile
	nimble:DeleteLaunchProfileMember
	nimble:DeleteStreamingImage
	nimble:DeleteStreamingSession
	nimble:DeleteStudio
	nimble:DeleteStudioComponent
	nimble:DeleteStudioMember
	nimble:GetEula
	nimble:GetLaunchProfileDetails
	nimble:GetStreamingImage
	nimble:GetStreamingSession
	nimble:GetStreamingSessionBackup
	nimble:GetStreamingSessionStream
	nimble:GetStudio

서비스 접두사	작업
	nimble:GetStudioComponent
	nimble:GetStudioMember
	nimble:ListEulas
	nimble:ListLaunchProfileMembers
	nimble:ListLaunchProfiles
	nimble:ListStreamingImages
	nimble:ListStreamingSessionBackups
	nimble:ListStreamingSessions
	nimble:ListStudioComponents
	nimble:ListStudioMembers
	nimble:ListStudios
	nimble:PutLaunchProfileMembers
	nimble:PutStudioMembers
	nimble:StartStreamingSession
	nimble:StartStudioSSOConfigurationRepair
	nimble:StopStreamingSession
	nimble:UpdateLaunchProfile
	nimble:UpdateLaunchProfileMember
	nimble:UpdateStreamingImage
	nimble:UpdateStudio
	nimble:UpdateStudioComponent

서비스 접두사	작업
omics	omics:AbortMultipartReadSetUpload omics:BatchDeleteReadSet omics:CancelAnnotationImportJob omics:CancelRun omics:CancelVariantImportJob omics:CompleteMultipartReadSetUpload omics:CreateAnnotationStore omics:CreateMultipartReadSetUpload omics:CreateReferenceStore omics:CreateRunGroup omics:CreateSequenceStore omics:CreateVariantStore omics:CreateWorkflow omics:DeleteAnnotationStore omics:DeleteReference omics:DeleteReferenceStore omics:DeleteRun omics:DeleteRunGroup omics:DeleteSequenceStore omics:DeleteVariantStore omics:DeleteWorkflow

서비스 접두사	작업
	omics:GetAnnotationImportJob omics:GetAnnotationStore omics:GetReadSet omics:GetReadSetActivationJob omics:GetReadSetExportJob omics:GetReadSetImportJob omics:GetReadSetMetadata omics:GetReference omics:GetReferenceImportJob omics:GetReferenceMetadata omics:GetReferenceStore omics:GetRun omics:GetRunGroup omics:GetRunTask omics:GetSequenceStore omics:GetVariantImportJob omics:GetVariantStore omics:GetWorkflow omics:ListAnnotationImportJobs omics:ListAnnotationStores omics:ListMultipartReadSetUploads

서비스 접두사	작업
	omics:ListReadSetActivationJobs omics:ListReadSetExportJobs omics:ListReadSetImportJobs omics:ListReadSets omics:ListReadSetUploadParts omics:ListReferenceImportJobs omics:ListReferences omics:ListReferenceStores omics:ListRunGroups omics:ListRuns omics:ListRunTasks omics:ListSequenceStores omics:ListVariantImportJobs omics:ListVariantStores omics:ListWorkflows omics:StartAnnotationImportJob omics:StartReadSetActivationJob omics:StartReadSetExportJob omics:StartReadSetImportJob omics:StartReferenceImportJob omics:StartRun

서비스 접두사	작업
	omics:StartVariantImportJob omics:UpdateAnnotationStore omics:UpdateRunGroup omics:UpdateVariantStore omics:UpdateWorkflow omics:UploadReadSetPart

서비스 접두사	작업
opsworks	opsworks:AssignInstance
	opsworks:AssignVolume
	opsworks:AssociateElasticIp
	opsworks:AttachElasticLoadBalancer
	opsworks:CloneStack
	opsworks:CreateApp
	opsworks:CreateDeployment
	opsworks:CreateInstance
	opsworks:CreateLayer
	opsworks:CreateStack
	opsworks:CreateUserProfile
	opsworks:DeleteApp
	opsworks:DeleteInstance
	opsworks:DeleteLayer
	opsworks:DeleteStack
	opsworks:DeleteUserProfile
	opsworks:DeregisterEcsCluster
	opsworks:DeregisterElasticIp
	opsworks:DeregisterInstance
	opsworks:DeregisterRdsDbInstance
	opsworks:DeregisterVolume

서비스 접두사	작업
	opsworks:DescribeAgentVersions
	opsworks:DescribeApps
	opsworks:DescribeCommands
	opsworks:DescribeDeployments
	opsworks:DescribeEcsClusters
	opsworks:DescribeElasticIps
	opsworks:DescribeElasticLoadBalancers
	opsworks:DescribeInstances
	opsworks:DescribeLayers
	opsworks:DescribeLoadBasedAutoScaling
	opsworks:DescribeMyUserProfile
	opsworks:DescribeOperatingSystems
	opsworks:DescribePermissions
	opsworks:DescribeRaidArrays
	opsworks:DescribeRdsDbInstances
	opsworks:DescribeServiceErrors
	opsworks:DescribeStackProvisioningParameters
	opsworks:DescribeStacks
	opsworks:DescribeStackSummary
	opsworks:DescribeTimeBasedAutoScaling
	opsworks:DescribeUserProfiles

서비스 접두사	작업
	opsworks:DescribeVolumes
	opsworks:DetachElasticLoadBalancer
	opsworks:DisassociateElasticIp
	opsworks:GetHostnameSuggestion
	opsworks:GrantAccess
	opsworks:RebootInstance
	opsworks:RegisterEcsCluster
	opsworks:RegisterElasticIp
	opsworks:RegisterInstance
	opsworks:RegisterRdsDbInstance
	opsworks:RegisterVolume
	opsworks:SetLoadBasedAutoScaling
	opsworks:SetPermission
	opsworks:SetTimeBasedAutoScaling
	opsworks:StartInstance
	opsworks:StartStack
	opsworks:StopInstance
	opsworks:StopStack
	opsworks:UnassignInstance
	opsworks:UnassignVolume
	opsworks:UpdateApp

서비스 접두사	작업
	opsworks:UpdateElasticIp
	opsworks:UpdateInstance
	opsworks:UpdateLayer
	opsworks:UpdateMyUserProfile
	opsworks:UpdateRdsDbInstance
	opsworks:UpdateStack
	opsworks:UpdateUserProfile
	opsworks:UpdateVolume

서비스 접두사	작업
opsworks-cm	opsworks-cm:AssociateNode opsworks-cm:CreateBackup opsworks-cm:CreateServer opsworks-cm:DeleteBackup opsworks-cm:DeleteServer opsworks-cm:DescribeAccountAttributes opsworks-cm:DescribeBackups opsworks-cm:DescribeEvents opsworks-cm:DescribeNodeAssociationStatus opsworks-cm:DescribeServers opsworks-cm:DisassociateNode opsworks-cm:ExportServerEngineAttribute opsworks-cm:RestoreServer opsworks-cm:StartMaintenance opsworks-cm:UpdateServer opsworks-cm:UpdateServerEngineAttributes

서비스 접두사	작업
organizations	organizations:AcceptHandshake
	organizations:AttachPolicy
	organizations:CancelHandshake
	organizations:CloseAccount
	organizations:CreateAccount
	organizations:CreateGovCloudAccount
	organizations:CreateOrganization
	organizations:CreateOrganizationalUnit
	organizations:CreatePolicy
	organizations:DeclineHandshake
	organizations:DeleteOrganization
	organizations:DeleteOrganizationalUnit
	organizations:DeletePolicy
	organizations:DeleteResourcePolicy
	organizations:DeregisterDelegatedAdministrator
	organizations:DescribeAccount
	organizations:DescribeCreateAccountStatus
	organizations:DescribeEffectivePolicy
	organizations:DescribeHandshake
	organizations:DescribeOrganization
	organizations:DescribeOrganizationalUnit

서비스 접두사	작업
	organizations:DescribePolicy
	organizations:DescribeResourcePolicy
	organizations:DetachPolicy
	organizations:DisableAWSServiceAccess
	organizations:DisablePolicyType
	organizations:EnableAllFeatures
	organizations:EnableAWSServiceAccess
	organizations:EnablePolicyType
	organizations:InviteAccountToOrganization
	organizations:LeaveOrganization
	organizations:ListAccounts
	organizations:ListAccountsForParent
	organizations:ListAWSServiceAccessForOrganization
	organizations:ListChildren
	organizations:ListCreateAccountStatus
	organizations:ListDelegatedAdministrators
	organizations:ListDelegatedServicesForAccount
	organizations:ListHandshakesForAccount
	organizations:ListHandshakesForOrganization
	organizations:ListOrganizationalUnitsForParent
	organizations:ListParents

서비스 접두사	작업
	organizations:ListPolicies organizations:ListPoliciesForTarget organizations:ListRoots organizations:ListTargetsForPolicy organizations:MoveAccount organizations:PutResourcePolicy organizations:RegisterDelegatedAdministrator organizations:RemoveAccountFromOrganization organizations:UpdateOrganizationalUnit organizations:UpdatePolicy

서비스 접두사	작업
outposts	outposts:CancelCapacityTask outposts:CancelOrder outposts:CreateOrder outposts:CreateOutpost outposts:CreatePrivateConnectivityConfig outposts:CreateSite outposts:DeleteOutpost outposts:DeleteSite outposts:GetCapacityTask outposts:GetCatalogItem outposts:GetConnection outposts:GetOrder outposts:GetOutpost outposts:GetOutpostInstanceTypes outposts:GetOutpostSupportedInstanceTypes outposts:GetPrivateConnectivityConfig outposts:GetSite outposts:GetSiteAddress outposts:ListAssets outposts:ListCapacityTasks outposts:ListCatalogItems

서비스 접두사	작업
	outposts:ListOrders outposts:ListOutposts outposts:ListSites outposts:StartCapacityTask outposts:StartConnection outposts:UpdateOutpost outposts:UpdateSite outposts:UpdateSiteAddress outposts:UpdateSiteRackPhysicalProperties

서비스 접두사	작업
panorama	panorama:CreateApplicationInstance
	panorama:CreateJobForDevices
	panorama:CreateNodeFromTemplateJob
	panorama:CreatePackage
	panorama:CreatePackageImportJob
	panorama:DeleteDevice
	panorama:DeletePackage
	panorama:DeregisterPackageVersion
	panorama:DescribeApplicationInstance
	panorama:DescribeApplicationInstanceDetails
	panorama:DescribeDevice
	panorama:DescribeDeviceJob
	panorama:DescribeNode
	panorama:DescribeNodeFromTemplateJob
	panorama:DescribePackage
	panorama:DescribePackageImportJob
	panorama:DescribePackageVersion
	panorama:ListApplicationInstanceDependencies
	panorama:ListApplicationInstanceNodeInstances
	panorama:ListApplicationInstances
	panorama:ListDevices

서비스 접두사	작업
	panorama:ListDevicesJobs panorama:ListNodeFromTemplateJobs panorama:ListNodes panorama:ListPackageImportJobs panorama:ListPackages panorama:ProvisionDevice panorama:RegisterPackageVersion panorama:RemoveApplicationInstance panorama:SignalApplicationInstanceNodeInstances panorama:UpdateDeviceMetadata
pi	pi:CreatePerformanceAnalysisReport pi:DeletePerformanceAnalysisReport pi:DescribeDimensionKeys pi:GetDimensionKeyDetails pi:GetPerformanceAnalysisReport pi:GetResourceMetadata pi:GetResourceMetrics pi:ListAvailableResourceDimensions pi:ListAvailableResourceMetrics pi:ListPerformanceAnalysisReports

서비스 접두사	작업
pipes	pipes:CreatePipe pipes:DeletePipe pipes:DescribePipe pipes:ListPipes pipes:StartPipe pipes:StopPipe pipes:UpdatePipe
polly	polly:DeleteLexicon polly:DescribeVoices polly:GetLexicon polly:GetSpeechSynthesisTask polly:ListLexicons polly:ListSpeechSynthesisTasks polly:PutLexicon polly:StartSpeechSynthesisTask polly:SynthesizeSpeech

서비스 접두사	작업
profile	profile:AddProfileKey
	profile:CreateCalculatedAttributeDefinition
	profile:CreateDomain
	profile:CreateEventStream
	profile:CreateProfile
	profile:DeleteCalculatedAttributeDefinition
	profile:DeleteDomain
	profile:DeleteEventStream
	profile:DeleteIntegration
	profile:DeleteProfile
	profile:DeleteProfileKey
	profile:DeleteProfileObject
	profile:DeleteProfileObjectType
	profile:DeleteWorkflow
	profile:DetectProfileObjectType
	profile:GetAutoMergingPreview
	profile:GetCalculatedAttributeDefinition
	profile:GetCalculatedAttributeForProfile
	profile:GetDomain
	profile:GetEventStream
	profile:GetIdentityResolutionJob

서비스 접두사	작업
	profile:GetIntegration
	profile:GetMatches
	profile:GetProfileObjectType
	profile:GetProfileObjectTypeTemplate
	profile:GetSimilarProfiles
	profile:GetWorkflow
	profile:GetWorkflowSteps
	profile:ListAccountIntegrations
	profile:ListCalculatedAttributeDefinitions
	profile:ListCalculatedAttributesForProfile
	profile:ListDomains
	profile:ListEventStreams
	profile:ListIdentityResolutionJobs
	profile:ListIntegrations
	profile:ListProfileObjects
	profile:ListProfileObjectTypes
	profile:ListProfileObjectTypeTemplates
	profile:ListRuleBasedMatches
	profile:ListWorkflows
	profile:MergeProfiles
	profile:PutIntegration

서비스 접두사	작업
	profile:PutProfileObject profile:PutProfileObjectType profile:SearchProfiles profile:UpdateCalculatedAttributeDefinition profile:UpdateDomain profile:UpdateProfile

서비스 접두사	작업
qldb	qldb:CancelJournalKinesisStream qldb:CreateLedger qldb:DeleteLedger qldb:DescribeJournalKinesisStream qldb:DescribeJournalS3Export qldb:DescribeLedger qldb:ExportJournalToS3 qldb:GetBlock qldb:GetDigest qldb:GetRevision qldb:ListJournalKinesisStreamsForLedger qldb:ListJournalS3Exports qldb:ListJournalS3ExportsForLedger qldb:ListLedgers qldb:StreamJournalToKinesis qldb:UpdateLedger qldb:UpdateLedgerPermissionsMode

서비스 접두사	작업
ram	ram:AcceptResourceShareInvitation
	ram:AssociateResourceShare
	ram:AssociateResourceSharePermission
	ram:CreatePermission
	ram:CreatePermissionVersion
	ram:CreateResourceShare
	ram:DeletePermission
	ram:DeletePermissionVersion
	ram:DeleteResourceShare
	ram:DisassociateResourceShare
	ram:DisassociateResourceSharePermission
	ram:EnableSharingWithAwsOrganization
	ram:GetPermission
	ram:GetResourcePolicies
	ram:GetResourceShareAssociations
	ram:GetResourceShareInvitations
	ram:GetResourceShares
	ram:ListPendingInvitationResources
	ram:ListPermissionAssociations
	ram:ListPermissions
	ram:ListPermissionVersions

서비스 접두사	작업
	ram:ListPrincipals ram:ListReplacePermissionAssociationsWork ram:ListResources ram:ListResourceSharePermissions ram:ListResourceTypes ram:PromotePermissionCreatedFromPolicy ram:PromoteResourceShareCreatedFromPolicy ram:RejectResourceShareInvitation ram:ReplacePermissionAssociations ram:SetDefaultPermissionVersion ram:UpdateResourceShare
rbin	rbin:CreateRule rbin:DeleteRule rbin:GetRule rbin:ListRules rbin:LockRule rbin:UnlockRule rbin:UpdateRule

서비스 접두사	작업
RDS	rds:AddRoleToDBCluster
	rds:AddRoleToDBInstance
	rds:AddSourceIdentifierToSubscription
	rds:ApplyPendingMaintenanceAction
	rds:AuthorizeDBSecurityGroupIngress
	rds:BacktrackDBCluster
	rds:CancelExportTask
	rds:CopyDBClusterParameterGroup
	rds:CopyDBClusterSnapshot
	rds:CopyDBParameterGroup
	rds:CopyDBSnapshot
	rds:CopyOptionGroup
	rds:CreateCustomDBEngineVersion
	rds:CreateDBClusterParameterGroup
	rds:CreateDBClusterSnapshot
	rds:CreateDBParameterGroup
	rds:CreateDBProxy
	rds:CreateDBProxyEndpoint
	rds:CreateDBSecurityGroup
	rds:CreateDBSnapshot
	rds:CreateDBSubnetGroup

서비스 접두사	작업
	rds:CreateEventSubscription
	rds:CreateGlobalCluster
	rds:CreateOptionGroup
	rds:DeleteBlueGreenDeployment
	rds:DeleteDBClusterAutomatedBackup
	rds:DeleteDBClusterParameterGroup
	rds:DeleteDBClusterSnapshot
	rds:DeleteDBInstanceAutomatedBackup
	rds:DeleteDBParameterGroup
	rds:DeleteDBProxy
	rds:DeleteDBProxyEndpoint
	rds:DeleteDBSecurityGroup
	rds:DeleteDBSnapshot
	rds:DeleteDBSubnetGroup
	rds:DeleteEventSubscription
	rds:DeleteGlobalCluster
	rds:DeleteOptionGroup
	rds:DeregisterDBProxyTargets
	rds:DescribeAccountAttributes
	rds:DescribeBlueGreenDeployments
	rds:DescribeCertificates

서비스 접두사	작업
	rds:DescribeDBClusterAutomatedBackups
	rds:DescribeDBClusterBacktracks
	rds:DescribeDBClusterEndpoints
	rds:DescribeDBClusterParameterGroups
	rds:DescribeDBClusterParameters
	rds:DescribeDBClusters
	rds:DescribeDBClusterSnapshotAttributes
	rds:DescribeDBClusterSnapshots
	rds:DescribeDBEngineVersions
	rds:DescribeDBInstanceAutomatedBackups
	rds:DescribeDBInstances
	rds:DescribeDBLogFiles
	rds:DescribeDBParameterGroups
	rds:DescribeDBParameters
	rds:DescribeDBProxies
	rds:DescribeDBProxyEndpoints
	rds:DescribeDBProxyTargetGroups
	rds:DescribeDBProxyTargets
	rds:DescribeDBRecommendations
	rds:DescribeDBSecurityGroups
	rds:DescribeDBSnapshotAttributes

서비스 접두사	작업
	rds:DescribeDBSnapshots
	rds:DescribeDbSnapshotTenantDatabases
	rds:DescribeDBSubnetGroups
	rds:DescribeEngineDefaultClusterParameters
	rds:DescribeEngineDefaultParameters
	rds:DescribeEventCategories
	rds:DescribeEvents
	rds:DescribeEventSubscriptions
	rds:DescribeExportTasks
	rds:DescribeGlobalClusters
	rds:DescribeIntegrations
	rds:DescribeOptionGroupOptions
	rds:DescribeOptionGroups
	rds:DescribeOrderableDBInstanceOptions
	rds:DescribePendingMaintenanceActions
	rds:DescribeReservedDBInstances
	rds:DescribeReservedDBInstancesOfferings
	rds:DescribeSourceRegions
	rds:DescribeTenantDatabases
	rds:DescribeValidDBInstanceModifications
	rds:DownloadCompleteDBLogFile

서비스 접두사	작업
	rds:DownloadDBLogFilePortion
	rds:FailoverDBCluster
	rds:FailoverGlobalCluster
	rds:ModifyActivityStream
	rds:ModifyCertificates
	rds:ModifyCurrentDBClusterCapacity
	rds:ModifyDBClusterEndpoint
	rds:ModifyDBClusterParameterGroup
	rds:ModifyDBClusterSnapshotAttribute
	rds:ModifyDBParameterGroup
	rds:ModifyDBProxy
	rds:ModifyDBProxyEndpoint
	rds:ModifyDBProxyTargetGroup
	rds:ModifyDBRecommendation
	rds:ModifyDBSnapshot
	rds:ModifyDBSnapshotAttribute
	rds:ModifyDBSubnetGroup
	rds:ModifyEventSubscription
	rds:ModifyGlobalCluster
	rds:ModifyOptionGroup
	rds:ModifyTenantDatabase

서비스 접두사	작업
	rds:PurchaseReservedDBInstancesOffering
	rds:RebootDBCluster
	rds:RegisterDBProxyTargets
	rds:RemoveFromGlobalCluster
	rds:RemoveRoleFromDBCluster
	rds:RemoveRoleFromDBInstance
	rds:RemoveSourceIdentifierFromSubscription
	rds:ResetDBClusterParameterGroup
	rds:ResetDBParameterGroup
	rds:RestoreDBClusterFromS3
	rds:RestoreDBClusterFromSnapshot
	rds:RestoreDBClusterToPointInTime
	rds:RestoreDBInstanceFromDBSnapshot
	rds:RestoreDBInstanceFromS3
	rds:RestoreDBInstanceToPointInTime
	rds:RevokeDBSecurityGroupIngress
	rds:StartActivityStream
	rds:StartDBCluster
	rds:StartDBInstance
	rds:StartDBInstanceAutomatedBackupsReplication
	rds:StartExportTask

서비스 접두사	작업
	rds:StopActivityStream rds:StopDBCluster rds:StopDBInstance rds:StopDBInstanceAutomatedBackupsReplication rds:SwitchoverBlueGreenDeployment rds:SwitchoverGlobalCluster rds:SwitchoverReadReplica

서비스 접두사	작업
redshift	redshift:AcceptReservedNodeExchange redshift:AddPartner redshift:AssociateDataShareConsumer redshift:AuthorizeClusterSecurityGroupIngress redshift:AuthorizeDataShare redshift:AuthorizeEndpointAccess redshift:AuthorizeSnapshotAccess redshift:BatchDeleteClusterSnapshots redshift:BatchModifyClusterSnapshots redshift:CancelResize redshift:CopyClusterSnapshot redshift:CreateAuthenticationProfile redshift:CreateCluster redshift:CreateClusterParameterGroup redshift:CreateClusterSecurityGroup redshift:CreateClusterSnapshot redshift:CreateClusterSubnetGroup redshift:CreateCustomDomainAssociation redshift:CreateEndpointAccess redshift:CreateEventSubscription redshift:CreateHsmClientCertificate

서비스 접두사	작업
	redshift:CreateHsmConfiguration redshift:CreateRedshiftIdcApplication redshift:CreateScheduledAction redshift:CreateSnapshotCopyGrant redshift:CreateSnapshotSchedule redshift:CreateUsageLimit redshift:DeauthorizeDataShare redshift:DeleteAuthenticationProfile redshift:DeleteCluster redshift:DeleteClusterParameterGroup redshift:DeleteClusterSecurityGroup redshift:DeleteClusterSnapshot redshift:DeleteClusterSubnetGroup redshift:DeleteCustomDomainAssociation redshift:DeleteEndpointAccess redshift:DeleteEventSubscription redshift:DeleteHsmClientCertificate redshift:DeleteHsmConfiguration redshift:DeletePartner redshift:DeleteScheduledAction redshift:DeleteSnapshotCopyGrant

서비스 접두사	작업
	redshift:DeleteSnapshotSchedule redshift:DeleteUsageLimit redshift:DescribeAccountAttributes redshift:DescribeAuthenticationProfiles redshift:DescribeClusterDbRevisions redshift:DescribeClusterParameterGroups redshift:DescribeClusterParameters redshift:DescribeClusters redshift:DescribeClusterSecurityGroups redshift:DescribeClusterSnapshots redshift:DescribeClusterSubnetGroups redshift:DescribeClusterTracks redshift:DescribeClusterVersions redshift:DescribeCustomDomainAssociations redshift:DescribeDataShares redshift:DescribeDataSharesForConsumer redshift:DescribeDataSharesForProducer redshift:DescribeDefaultClusterParameters redshift:DescribeEndpointAccess redshift:DescribeEndpointAuthorization redshift:DescribeEventCategories

서비스 접두사	작업
	redshift:DescribeEvents redshift:DescribeEventSubscriptions redshift:DescribeHsmClientCertificates redshift:DescribeHsmConfigurations redshift:DescribeInboundIntegrations redshift:DescribeLoggingStatus redshift:DescribeNodeConfigurationOptions redshift:DescribeOrderableClusterOptions redshift:DescribePartners redshift:DescribeRedshiftIdcApplications redshift:DescribeReservedNodeExchangeStatus redshift:DescribeReservedNodeOfferings redshift:DescribeReservedNodes redshift:DescribeResize redshift:DescribeScheduledActions redshift:DescribeSnapshotCopyGrants redshift:DescribeSnapshotSchedules redshift:DescribeStorage redshift:DescribeTableRestoreStatus redshift:DescribeUsageLimits redshift:DisableLogging

서비스 접두사	작업
	redshift:DisableSnapshotCopy redshift:DisassociateDataShareConsumer redshift:EnableLogging redshift:EnableSnapshotCopy redshift:FailoverPrimaryCompute redshift:GetClusterCredentials redshift:GetClusterCredentialsWithIAM redshift:GetReservedNodeExchangeConfigurationOptions redshift:GetReservedNodeExchangeOfferings redshift:ListRecommendations redshift:ModifyAquaConfiguration redshift:ModifyAuthenticationProfile redshift:ModifyCluster redshift:ModifyClusterDbRevision redshift:ModifyClusterIamRoles redshift:ModifyClusterMaintenance redshift:ModifyClusterParameterGroup redshift:ModifyClusterSnapshot redshift:ModifyClusterSnapshotSchedule redshift:ModifyClusterSubnetGroup redshift:ModifyCustomDomainAssociation

서비스 접두사	작업
	redshift:ModifyEndpointAccess redshift:ModifyEventSubscription redshift:ModifyScheduledAction redshift:ModifySnapshotCopyRetentionPeriod redshift:ModifySnapshotSchedule redshift:ModifyUsageLimit redshift:PauseCluster redshift:PurchaseReservedNodeOffering redshift:RebootCluster redshift:RejectDataShare redshift:ResetClusterParameterGroup redshift:ResizeCluster redshift:RestoreFromClusterSnapshot redshift:RestoreTableFromClusterSnapshot redshift:ResumeCluster redshift:RevokeClusterSecurityGroupIngress redshift:RevokeEndpointAccess redshift:RevokeSnapshotAccess redshift:RotateEncryptionKey redshift:UpdatePartnerStatus

서비스 접두사	작업
redshift-data	redshift-data:BatchExecuteStatement
	redshift-data:CancelStatement
	redshift-data:DescribeStatement
	redshift-data:DescribeTable
	redshift-data:ExecuteStatement
	redshift-data:GetStatementResult
	redshift-data:ListDatabases
	redshift-data:ListSchemas
	redshift-data:ListStatements
	redshift-data:ListTables

서비스 접두사	작업
refactor-spaces	refactor-spaces:CreateApplication refactor-spaces:CreateEnvironment refactor-spaces:CreateRoute refactor-spaces:CreateService refactor-spaces:DeleteApplication refactor-spaces:DeleteEnvironment refactor-spaces:DeleteResourcePolicy refactor-spaces:DeleteRoute refactor-spaces:DeleteService refactor-spaces:GetApplication refactor-spaces:GetEnvironment refactor-spaces:GetResourcePolicy refactor-spaces:GetRoute refactor-spaces:GetService refactor-spaces:ListApplications refactor-spaces:ListEnvironments refactor-spaces:ListEnvironmentVpcs refactor-spaces:ListRoutes refactor-spaces:ListServices refactor-spaces:PutResourcePolicy refactor-spaces:UpdateRoute

서비스 접두사	작업
rekognition	rekognition:AssociateFaces
	rekognition:CompareFaces
	rekognition:CopyProjectVersion
	rekognition:CreateCollection
	rekognition:CreateDataset
	rekognition:CreateFaceLivenessSession
	rekognition:CreateProject
	rekognition:CreateProjectVersion
	rekognition:CreateStreamProcessor
	rekognition:CreateUser
	rekognition:DeleteCollection
	rekognition:DeleteDataset
	rekognition:DeleteFaces
	rekognition:DeleteProject
	rekognition:DeleteProjectPolicy
	rekognition:DeleteProjectVersion
	rekognition:DeleteStreamProcessor
	rekognition:DeleteUser
	rekognition:DescribeCollection
	rekognition:DescribeDataset
	rekognition:DescribeProjects

서비스 접두사	작업
	rekognition:DescribeProjectVersions
	rekognition:DescribeStreamProcessor
	rekognition:DetectCustomLabels
	rekognition:DetectFaces
	rekognition:DetectLabels
	rekognition:DetectModerationLabels
	rekognition:DetectProtectiveEquipment
	rekognition:DetectText
	rekognition:DisassociateFaces
	rekognition:DistributeDatasetEntries
	rekognition:GetCelebrityInfo
	rekognition:GetCelebrityRecognition
	rekognition:GetContentModeration
	rekognition:GetFaceDetection
	rekognition:GetFaceLivenessSessionResults
	rekognition:GetFaceSearch
	rekognition:GetLabelDetection
	rekognition:GetMediaAnalysisJob
	rekognition:GetPersonTracking
	rekognition:GetSegmentDetection
	rekognition:GetTextDetection

서비스 접두사	작업
	rekognition:IndexFaces
	rekognition:ListCollections
	rekognition:ListDatasetEntries
	rekognition:ListDatasetLabels
	rekognition:ListFaces
	rekognition:ListMediaAnalysisJobs
	rekognition:ListProjectPolicies
	rekognition:ListStreamProcessors
	rekognition:ListUsers
	rekognition:PutProjectPolicy
	rekognition:RecognizeCelebrities
	rekognition:SearchFaces
	rekognition:SearchFacesByImage
	rekognition:SearchUsers
	rekognition:SearchUsersByImage
	rekognition:StartCelebrityRecognition
	rekognition:StartContentModeration
	rekognition:StartFaceDetection
	rekognition:StartFaceLivenessSession
	rekognition:StartFaceSearch
	rekognition:StartLabelDetection

서비스 접두사	작업
	rekognition:StartMediaAnalysisJob
	rekognition:StartPersonTracking
	rekognition:StartProjectVersion
	rekognition:StartSegmentDetection
	rekognition:StartStreamProcessor
	rekognition:StartTextDetection
	rekognition:StopProjectVersion
	rekognition:StopStreamProcessor
	rekognition:UpdateDatasetEntries
	rekognition:UpdateStreamProcessor

서비스 접두사	작업
resiliencehub	resiliencehub:AddDraftAppVersionResourceMappings resiliencehub:BatchUpdateRecommendationStatus resiliencehub:CreateApp resiliencehub:CreateAppVersionAppComponent resiliencehub:CreateAppVersionResource resiliencehub:CreateRecommendationTemplate resiliencehub:CreateResiliencyPolicy resiliencehub:DeleteApp resiliencehub:DeleteAppAssessment resiliencehub:DeleteAppInputSource resiliencehub:DeleteAppVersionAppComponent resiliencehub:DeleteAppVersionResource resiliencehub:DeleteRecommendationTemplate resiliencehub:DeleteResiliencyPolicy resiliencehub:DescribeApp resiliencehub:DescribeAppAssessment resiliencehub:DescribeAppVersion resiliencehub:DescribeAppVersionAppComponent resiliencehub:DescribeAppVersionResource resiliencehub:DescribeAppVersionResourcesResolutionStatus resiliencehub:DescribeAppVersionTemplate

서비스 접두사	작업
	resiliencehub:DescribeDraftAppVersionResourcesImportStatus resiliencehub:DescribeResiliencyPolicy resiliencehub:ImportResourcesToDraftAppVersion resiliencehub:ListAlarmRecommendations resiliencehub:ListAppAssessmentComplianceDrifts resiliencehub:ListAppAssessmentResourceDrifts resiliencehub:ListAppAssessments resiliencehub:ListAppComponentCompliances resiliencehub:ListAppComponentRecommendations resiliencehub:ListAppInputSources resiliencehub:ListApps resiliencehub:ListAppVersionAppComponents resiliencehub:ListAppVersionResourceMappings resiliencehub:ListAppVersionResources resiliencehub:ListAppVersions resiliencehub:ListRecommendationTemplates resiliencehub:ListResiliencyPolicies resiliencehub:ListSopRecommendations resiliencehub:ListSuggestedResiliencyPolicies resiliencehub:ListTestRecommendations resiliencehub:ListUnsupportedAppVersionResources

서비스 접두사	작업
	resiliencehub:PublishAppVersion resiliencehub:PutDraftAppVersionTemplate resiliencehub:RemoveDraftAppVersionResourceMappings resiliencehub:ResolveAppVersionResources resiliencehub:StartAppAssessment resiliencehub:UpdateApp resiliencehub:UpdateAppVersion resiliencehub:UpdateAppVersionAppComponent resiliencehub:UpdateAppVersionResource resiliencehub:UpdateResiliencyPolicy

서비스 접두사	작업
resource-explorer-2	resource-explorer-2:AssociateDefaultView
	resource-explorer-2:BatchGetView
	resource-explorer-2:CreateIndex
	resource-explorer-2:CreateView
	resource-explorer-2:DeleteIndex
	resource-explorer-2:DeleteView
	resource-explorer-2:DisassociateDefaultView
	resource-explorer-2:GetAccountLevelServiceConfiguration
	resource-explorer-2:GetDefaultView
	resource-explorer-2:GetIndex
	resource-explorer-2:ListIndexes
	resource-explorer-2:ListIndexesForMembers
	resource-explorer-2:ListSupportedResourceTypes
	resource-explorer-2:ListViews
	resource-explorer-2:Search
	resource-explorer-2:UpdateIndexType
	resource-explorer-2:UpdateView

서비스 접두사	작업
resource-groups	resource-groups:CreateGroup resource-groups:DeleteGroup resource-groups:GetAccountSettings resource-groups:GetGroup resource-groups:GetGroupConfiguration resource-groups:GetGroupQuery resource-groups:GroupResources resource-groups:ListGroupResources resource-groups:ListGroups resource-groups:PutGroupConfiguration resource-groups:SearchResources resource-groups:UngroupResources resource-groups:UpdateAccountSettings resource-groups:UpdateGroup resource-groups:UpdateGroupQuery

서비스 접두사	작업
robomaker	robomaker:BatchDeleteWorlds
	robomaker:BatchDescribeSimulationJob
	robomaker:CancelDeploymentJob
	robomaker:CancelSimulationJob
	robomaker:CancelSimulationJobBatch
	robomaker:CancelWorldExportJob
	robomaker:CancelWorldGenerationJob
	robomaker:CreateDeploymentJob
	robomaker:CreateFleet
	robomaker:CreateRobot
	robomaker:CreateRobotApplication
	robomaker:CreateRobotApplicationVersion
	robomaker:CreateSimulationApplication
	robomaker:CreateSimulationApplicationVersion
	robomaker:CreateSimulationJob
	robomaker:CreateWorldExportJob
	robomaker:CreateWorldGenerationJob
	robomaker:CreateWorldTemplate
	robomaker:DeleteFleet
	robomaker:DeleteRobot
	robomaker:DeleteRobotApplication

서비스 접두사	작업
	robomaker:DeleteSimulationApplication
	robomaker:DeleteWorldTemplate
	robomaker:DeregisterRobot
	robomaker:DescribeDeploymentJob
	robomaker:DescribeFleet
	robomaker:DescribeRobot
	robomaker:DescribeRobotApplication
	robomaker:DescribeSimulationApplication
	robomaker:DescribeSimulationJob
	robomaker:DescribeSimulationJobBatch
	robomaker:DescribeWorld
	robomaker:DescribeWorldExportJob
	robomaker:DescribeWorldGenerationJob
	robomaker:DescribeWorldTemplate
	robomaker:GetWorldTemplateBody
	robomaker:ListDeploymentJobs
	robomaker:ListFleets
	robomaker:ListRobotApplications
	robomaker:ListRobots
	robomaker:ListSimulationApplications
	robomaker:ListSimulationJobBatches

서비스 접두사	작업
	robomaker:ListSimulationJobs
	robomaker:ListWorldExportJobs
	robomaker:ListWorldGenerationJobs
	robomaker:ListWorlds
	robomaker:ListWorldTemplates
	robomaker:RegisterRobot
	robomaker:RestartSimulationJob
	robomaker:StartSimulationJobBatch
	robomaker:SyncDeploymentJob
	robomaker:UpdateRobotApplication
	robomaker:UpdateSimulationApplication
	robomaker:UpdateWorldTemplate

서비스 접두사	작업
rolesanywhere	rolesanywhere:CreateProfile rolesanywhere:CreateTrustAnchor rolesanywhere:DeleteAttributeMapping rolesanywhere:DeleteCrl rolesanywhere:DeleteProfile rolesanywhere:DeleteTrustAnchor rolesanywhere:DisableCrl rolesanywhere:DisableProfile rolesanywhere:DisableTrustAnchor rolesanywhere:EnableCrl rolesanywhere:EnableProfile rolesanywhere:EnableTrustAnchor rolesanywhere:GetCrl rolesanywhere:GetProfile rolesanywhere:GetSubject rolesanywhere:GetTrustAnchor rolesanywhere:ImportCrl rolesanywhere:ListCrls rolesanywhere:ListProfiles rolesanywhere:ListSubjects rolesanywhere:ListTrustAnchors

서비스 접두사	작업
	rolesanywhere:PutAttributeMapping rolesanywhere:PutNotificationSettings rolesanywhere:ResetNotificationSettings rolesanywhere:UpdateCrl rolesanywhere:UpdateProfile rolesanywhere:UpdateTrustAnchor

서비스 접두사	작업
route53	route53:ActivateKeySigningKey
	route53:AssociateVPCWithHostedZone
	route53:ChangeCidrCollection
	route53:ChangeResourceRecordSets
	route53:CreateCidrCollection
	route53:CreateHealthCheck
	route53:CreateHostedZone
	route53:CreateKeySigningKey
	route53:CreateQueryLoggingConfig
	route53:CreateReusableDelegationSet
	route53:CreateTrafficPolicy
	route53:CreateTrafficPolicyInstance
	route53:CreateTrafficPolicyVersion
	route53:CreateVPCAssociationAuthorization
	route53:DeactivateKeySigningKey
	route53:DeleteCidrCollection
	route53:DeleteHealthCheck
	route53:DeleteHostedZone
	route53:DeleteKeySigningKey
	route53:DeleteQueryLoggingConfig
	route53:DeleteReusableDelegationSet

서비스 접두사	작업
	route53:DeleteTrafficPolicy
	route53:DeleteTrafficPolicyInstance
	route53:DeleteVPCAssociationAuthorization
	route53:DisableHostedZoneDNSSEC
	route53:DisassociateVPCFromHostedZone
	route53:EnableHostedZoneDNSSEC
	route53:GetAccountLimit
	route53:GetChange
	route53:GetCheckerIpRanges
	route53:GetDNSSEC
	route53:GetGeoLocation
	route53:GetHealthCheck
	route53:GetHealthCheckCount
	route53:GetHealthCheckLastFailureReason
	route53:GetHealthCheckStatus
	route53:GetHostedZone
	route53:GetHostedZoneCount
	route53:GetHostedZoneLimit
	route53:GetQueryLoggingConfig
	route53:GetReusableDelegationSet
	route53:GetReusableDelegationSetLimit

서비스 접두사	작업
	route53:GetTrafficPolicy
	route53:GetTrafficPolicyInstance
	route53:GetTrafficPolicyInstanceCount
	route53:ListCidrBlocks
	route53:ListCidrCollections
	route53:ListCidrLocations
	route53:ListGeoLocations
	route53:ListHealthChecks
	route53:ListHostedZones
	route53:ListHostedZonesByName
	route53:ListHostedZonesByVPC
	route53:ListQueryLoggingConfigs
	route53:ListResourceRecordSets
	route53:ListReusableDelegationSets
	route53:ListTrafficPolicies
	route53:ListTrafficPolicyInstances
	route53:ListTrafficPolicyInstancesByHostedZone
	route53:ListTrafficPolicyInstancesByPolicy
	route53:ListTrafficPolicyVersions
	route53:ListVPCAssociationAuthorizations
	route53:TestDNSAnswer

서비스 접두사	작업
	route53:UpdateHealthCheck route53:UpdateHostedZoneComment route53:UpdateTrafficPolicyComment route53:UpdateTrafficPolicyInstance

서비스 접두사	작업
route53-recovery-control-config	route53-recovery-control-config:CreateCluster route53-recovery-control-config:CreateControlPanel route53-recovery-control-config:CreateRoutingControl route53-recovery-control-config:CreateSafetyRule route53-recovery-control-config:DeleteCluster route53-recovery-control-config:DeleteControlPanel route53-recovery-control-config:DeleteRoutingControl route53-recovery-control-config:DeleteSafetyRule route53-recovery-control-config:DescribeCluster route53-recovery-control-config:DescribeControlPanel route53-recovery-control-config:DescribeRoutingControl route53-recovery-control-config:DescribeSafetyRule route53-recovery-control-config:GetResourcePolicy route53-recovery-control-config:ListAssociatedRoute53HealthChecks route53-recovery-control-config:ListClusters route53-recovery-control-config:ListControlPanels route53-recovery-control-config:ListRoutingControls route53-recovery-control-config:ListSafetyRules route53-recovery-control-config:UpdateControlPanel route53-recovery-control-config:UpdateRoutingControl

서비스 접두사	작업
	route53-recovery-control-config:UpdateSafetyRule

서비스 접두사	작업
route53-recovery-readiness	route53-recovery-readiness:CreateCell route53-recovery-readiness:CreateCrossAccountAuthorization route53-recovery-readiness:CreateReadinessCheck route53-recovery-readiness:CreateRecoveryGroup route53-recovery-readiness:CreateResourceSet route53-recovery-readiness:DeleteCell route53-recovery-readiness:DeleteCrossAccountAuthorization route53-recovery-readiness:DeleteReadinessCheck route53-recovery-readiness:DeleteRecoveryGroup route53-recovery-readiness:DeleteResourceSet route53-recovery-readiness:GetArchitectureRecommendations route53-recovery-readiness:GetCell route53-recovery-readiness:GetCellReadinessSummary route53-recovery-readiness:GetReadinessCheck route53-recovery-readiness:GetReadinessCheckResourceStatus route53-recovery-readiness:GetReadinessCheckStatus route53-recovery-readiness:GetRecoveryGroup route53-recovery-readiness:GetRecoveryGroupReadinessSummary route53-recovery-readiness:GetResourceSet route53-recovery-readiness:ListCells route53-recovery-readiness:ListCrossAccountAuthorizations

서비스 접두사	작업
	route53-recovery-readiness:ListReadinessChecks
	route53-recovery-readiness:ListRecoveryGroups
	route53-recovery-readiness:ListResourceSets
	route53-recovery-readiness:ListRules
	route53-recovery-readiness:UpdateCell
	route53-recovery-readiness:UpdateReadinessCheck
	route53-recovery-readiness:UpdateRecoveryGroup
	route53-recovery-readiness:UpdateResourceSet

서비스 접두사	작업
route53resolver	route53resolver:AssociateFirewallRuleGroup route53resolver:AssociateResolverEndpointIpAddress route53resolver:AssociateResolverQueryLogConfig route53resolver:AssociateResolverRule route53resolver:CreateFirewallDomainList route53resolver:CreateFirewallRule route53resolver:CreateFirewallRuleGroup route53resolver:CreateResolverEndpoint route53resolver:CreateResolverQueryLogConfig route53resolver:CreateResolverRule route53resolver:DeleteFirewallDomainList route53resolver:DeleteFirewallRule route53resolver:DeleteFirewallRuleGroup route53resolver:DeleteOutpostResolver route53resolver:DeleteResolverEndpoint route53resolver:DeleteResolverQueryLogConfig route53resolver:DeleteResolverRule route53resolver:DisassociateFirewallRuleGroup route53resolver:DisassociateResolverEndpointIpAddress route53resolver:DisassociateResolverQueryLogConfig route53resolver:DisassociateResolverRule

서비스 접두사	작업
	route53resolver:GetFirewallConfig route53resolver:GetFirewallDomainList route53resolver:GetFirewallRuleGroup route53resolver:GetFirewallRuleGroupAssociation route53resolver:GetFirewallRuleGroupPolicy route53resolver:GetOutpostResolver route53resolver:GetResolverConfig route53resolver:GetResolverDnssecConfig route53resolver:GetResolverEndpoint route53resolver:GetResolverQueryLogConfig route53resolver:GetResolverQueryLogConfigAssociation route53resolver:GetResolverQueryLogConfigPolicy route53resolver:GetResolverRule route53resolver:GetResolverRuleAssociation route53resolver:GetResolverRulePolicy route53resolver:ImportFirewallDomains route53resolver:ListFirewallConfigs route53resolver:ListFirewallDomainLists route53resolver:ListFirewallDomains route53resolver:ListFirewallRuleGroupAssociations route53resolver:ListFirewallRuleGroups

서비스 접두사	작업
	route53resolver:ListFirewallRules route53resolver:ListOutpostResolvers route53resolver:ListResolverConfigs route53resolver:ListResolverDnssecConfigs route53resolver:ListResolverEndpointIpAddresses route53resolver:ListResolverEndpoints route53resolver:ListResolverQueryLogConfigAssociations route53resolver:ListResolverQueryLogConfigs route53resolver:ListResolverRuleAssociations route53resolver:ListResolverRules route53resolver:PutFirewallRuleGroupPolicy route53resolver:PutResolverQueryLogConfigPolicy route53resolver:UpdateFirewallConfig route53resolver:UpdateFirewallDomains route53resolver:UpdateFirewallRule route53resolver:UpdateFirewallRuleGroupAssociation route53resolver:UpdateOutpostResolver route53resolver:UpdateResolverConfig route53resolver:UpdateResolverDnssecConfig route53resolver:UpdateResolverEndpoint route53resolver:UpdateResolverRule

서비스 접두사	작업
rum	rum:BatchCreateRumMetricDefinitions rum:BatchDeleteRumMetricDefinitions rum:BatchGetRumMetricDefinitions rum:CreateAppMonitor rum:DeleteAppMonitor rum:DeleteRumMetricsDestination rum:GetAppMonitor rum:GetAppMonitorData rum:ListAppMonitors rum:ListRumMetricsDestinations rum:PutRumMetricsDestination rum:UpdateAppMonitor rum:UpdateRumMetricDefinition

서비스 접두사	작업
s3	s3:AssociateAccessGrantsIdentityCenter s3:CreateAccessGrant s3:CreateAccessGrantsInstance s3:CreateAccessGrantsLocation s3:CreateAccessPoint s3:CreateAccessPointForObjectLambda s3:CreateBucket s3:CreateJob s3:CreateMultiRegionAccessPoint s3:DeleteAccessGrant s3:DeleteAccessGrantsInstance s3:DeleteAccessGrantsInstanceResourcePolicy s3:DeleteAccessGrantsLocation s3:DeleteAccessPoint s3:DeleteAccessPointForObjectLambda s3:DeleteAccessPointPolicy s3:DeleteAccessPointPolicyForObjectLambda s3:PutAccountPublicAccessBlock s3:DeleteBucket s3:PutAnalyticsConfiguration s3:PutBucketCORS

서비스 접두사	작업
	s3:PutEncryptionConfiguration s3:PutIntelligentTieringConfiguration s3:PutInventoryConfiguration s3:PutLifecycleConfiguration s3:PutMetricsConfiguration s3:PutBucketOwnershipControls s3:DeleteBucketPolicy s3:PutBucketPublicAccessBlock s3:PutReplicationConfiguration s3:DeleteBucketWebsite s3:DeleteMultiRegionAccessPoint s3:DeleteStorageLensConfiguration s3:DescribeJob s3:DescribeMultiRegionAccessPointOperation s3:DissociateAccessGrantsIdentityCenter s3:GetAccelerateConfiguration s3:GetAccessGrant s3:GetAccessGrantsInstance s3:GetAccessGrantsInstanceForPrefix s3:GetAccessGrantsInstanceResourcePolicy s3:GetAccessGrantsLocation

서비스 접두사	작업
	s3:GetAccessPoint s3:GetAccessPointConfigurationForObjectLambda s3:GetAccessPointForObjectLambda s3:GetAccessPointPolicy s3:GetAccessPointPolicyForObjectLambda s3:GetAccessPointPolicyStatus s3:GetAccessPointPolicyStatusForObjectLambda s3:GetAccountPublicAccessBlock s3:GetBucketAcl s3:GetAnalyticsConfiguration s3:GetBucketCORS s3:GetEncryptionConfiguration s3:GetIntelligentTieringConfiguration s3:GetInventoryConfiguration s3:GetLifecycleConfiguration s3:GetBucketLocation s3:GetBucketLogging s3:GetMetricsConfiguration s3:GetBucketNotification s3:GetBucketObjectLockConfiguration s3:GetBucketOwnershipControls

서비스 접두사	작업
	s3:GetBucketPolicy s3:GetBucketPolicyStatus s3:GetBucketPublicAccessBlock s3:GetReplicationConfiguration s3:GetBucketRequestPayment s3:GetBucketVersioning s3:GetBucketWebsite s3:GetDataAccess s3:GetMultiRegionAccessPoint s3:GetMultiRegionAccessPointPolicy s3:GetMultiRegionAccessPointPolicyStatus s3:GetMultiRegionAccessPointRoutes s3:GetObjectAttributes s3:GetStorageLensConfiguration s3:GetStorageLensDashboard s3:ListAccessGrants s3:ListAccessGrantsInstances s3:ListAccessGrantsLocations s3:ListAccessPoints s3:ListAccessPointsForObjectLambda s3:ListAllMyBuckets

서비스 접두사	작업
	s3:ListJobs s3:ListBucketMultipartUploads s3:ListMultiRegionAccessPoints s3:ListStorageLensConfigurations s3:PutAccelerateConfiguration s3:PutAccessGrantsInstanceResourcePolicy s3:PutAccessPointConfigurationForObjectLambda s3:PutAccessPointPolicy s3:PutAccessPointPolicyForObjectLambda s3:PutAccountPublicAccessBlock s3:PutBucketAcl s3:PutAnalyticsConfiguration s3:PutBucketCORS s3:PutEncryptionConfiguration s3:PutIntelligentTieringConfiguration s3:PutInventoryConfiguration s3:PutLifecycleConfiguration s3:PutBucketLogging s3:PutMetricsConfiguration s3:PutBucketNotification s3:PutBucketObjectLockConfiguration

서비스 접두사	작업
	s3:PutBucketOwnershipControls s3:PutBucketPolicy s3:PutBucketPublicAccessBlock s3:PutReplicationConfiguration s3:PutBucketRequestPayment s3:PutBucketVersioning s3:PutBucketWebsite s3:PutMultiRegionAccessPointPolicy s3:PutStorageLensConfiguration s3:SubmitMultiRegionAccessPointRoutes s3:UpdateAccessGrantsLocation s3:UpdateJobPriority s3:UpdateJobStatus
s3-outposts	s3-outposts:CreateEndpoint s3-outposts:DeleteEndpoint s3-outposts:ListEndpoints s3-outposts:ListOutpostsWithS3 s3-outposts:ListSharedEndpoints

서비스 접두사	작업
sagemaker-geospatial	sagemaker-geospatial:DeleteEarthObservationJob sagemaker-geospatial:DeleteVectorEnrichmentJob sagemaker-geospatial:ExportEarthObservationJob sagemaker-geospatial:ExportVectorEnrichmentJob sagemaker-geospatial:GetEarthObservationJob sagemaker-geospatial:GetRasterDataCollection sagemaker-geospatial:GetTile sagemaker-geospatial:GetVectorEnrichmentJob sagemaker-geospatial:ListEarthObservationJobs sagemaker-geospatial:ListRasterDataCollections sagemaker-geospatial:ListVectorEnrichmentJobs sagemaker-geospatial:SearchRasterDataCollection sagemaker-geospatial:StartEarthObservationJob sagemaker-geospatial:StartVectorEnrichmentJob sagemaker-geospatial:StopEarthObservationJob sagemaker-geospatial:StopVectorEnrichmentJob

서비스 접두사	작업
savingsplans	savingsplans:CreateSavingsPlan savingsplans:DeleteQueuedSavingsPlan savingsplans:DescribeSavingsPlanRates savingsplans:DescribeSavingsPlans savingsplans:DescribeSavingsPlansOfferingRates savingsplans:DescribeSavingsPlansOfferings savingsplans:ReturnSavingsPlan

서비스 접두사	작업
schemas	schemas:CreateDiscoverer schemas:CreateRegistry schemas:CreateSchema schemas:DeleteDiscoverer schemas:DeleteRegistry schemas:DeleteResourcePolicy schemas:DeleteSchema schemas:DeleteSchemaVersion schemas:DescribeCodeBinding schemas:DescribeDiscoverer schemas:DescribeRegistry schemas:DescribeSchema schemas:ExportSchema schemas:GetCodeBindingSource schemas:GetDiscoveredSchema schemas:GetResourcePolicy schemas:ListDiscoverers schemas:ListRegistries schemas:ListSchemas schemas:ListSchemaVersions schemas:PutCodeBinding

서비스 접두사	작업
	schemas:PutResourcePolicy schemas:SearchSchemas schemas:StartDiscoverer schemas:StopDiscoverer schemas:UpdateDiscoverer schemas:UpdateRegistry schemas:UpdateSchema
sdb	sdb:CreateDomain sdb:DeleteDomain sdb:DomainMetadata sdb:ListDomains

서비스 접두사	작업
secretsmanager	secretsmanager:CancelRotateSecret secretsmanager:CreateSecret secretsmanager:DeleteResourcePolicy secretsmanager:DeleteSecret secretsmanager:DescribeSecret secretsmanager:GetRandomPassword secretsmanager:GetResourcePolicy secretsmanager:GetSecretValue secretsmanager:ListSecrets secretsmanager:ListSecretVersionIds secretsmanager:PutResourcePolicy secretsmanager:PutSecretValue secretsmanager:RemoveRegionsFromReplication secretsmanager:ReplicateSecretToRegions secretsmanager:RestoreSecret secretsmanager:RotateSecret secretsmanager:StopReplicationToReplica secretsmanager:UpdateSecret secretsmanager:ValidateResourcePolicy

서비스 접두사	작업
securityhub	securityhub:AcceptAdministratorInvitation
	securityhub:AcceptInvitation
	securityhub:BatchDeleteAutomationRules
	securityhub:BatchDisableStandards
	securityhub:BatchEnableStandards
	securityhub:BatchGetAutomationRules
	securityhub:BatchGetConfigurationPolicyAssociations
	securityhub:BatchGetSecurityControls
	securityhub:BatchGetStandardsControlAssociations
	securityhub:BatchImportFindings
	securityhub:BatchUpdateAutomationRules
	securityhub:BatchUpdateFindings
	securityhub:BatchUpdateStandardsControlAssociations
	securityhub:CreateActionTarget
	securityhub:CreateAutomationRule
	securityhub:CreateConfigurationPolicy
	securityhub:CreateFindingAggregator
	securityhub:CreateInsight
	securityhub:CreateMembers
	securityhub:DeclineInvitations
	securityhub:DeleteActionTarget

서비스 접두사	작업
	securityhub:DeleteConfigurationPolicy
	securityhub:DeleteFindingAggregator
	securityhub:DeleteInsight
	securityhub:DeleteInvitations
	securityhub:DeleteMembers
	securityhub:DescribeActionTargets
	securityhub:DescribeHub
	securityhub:DescribeOrganizationConfiguration
	securityhub:DescribeProducts
	securityhub:DescribeStandards
	securityhub:DisableImportFindingsForProduct
	securityhub:DisableOrganizationAdminAccount
	securityhub:DisableSecurityHub
	securityhub:DisassociateFromAdministratorAccount
	securityhub:DisassociateFromMasterAccount
	securityhub:DisassociateMembers
	securityhub:EnableImportFindingsForProduct
	securityhub:EnableOrganizationAdminAccount
	securityhub:EnableSecurityHub
	securityhub:GetAdministratorAccount
	securityhub:GetConfigurationPolicy

서비스 접두사	작업
	securityhub:GetConfigurationPolicyAssociation
	securityhub:GetEnabledStandards
	securityhub:GetFindingAggregator
	securityhub:GetFindingHistory
	securityhub:GetFindings
	securityhub:GetInsightResults
	securityhub:GetInsights
	securityhub:GetInvitationsCount
	securityhub:GetMasterAccount
	securityhub:GetMembers
	securityhub:GetSecurityControlDefinition
	securityhub:InviteMembers
	securityhub:ListAutomationRules
	securityhub:ListConfigurationPolicies
	securityhub:ListConfigurationPolicyAssociations
	securityhub:ListEnabledProductsForImport
	securityhub:ListFindingAggregators
	securityhub:ListInvitations
	securityhub:ListMembers
	securityhub:ListOrganizationAdminAccounts
	securityhub:ListSecurityControlDefinitions

서비스 접두사	작업
	securityhub:ListStandardsControlAssociations
	securityhub:StartConfigurationPolicyAssociation
	securityhub:StartConfigurationPolicyDisassociation
	securityhub:UpdateActionTarget
	securityhub:UpdateConfigurationPolicy
	securityhub:UpdateFindingAggregator
	securityhub:UpdateFindings
	securityhub:UpdateInsight
	securityhub:UpdateOrganizationConfiguration
	securityhub:UpdateSecurityControl
	securityhub:UpdateSecurityHubConfiguration

서비스 접두사	작업
securitylake	securitylake:CreateAwsLogSource securitylake:CreateCustomLogSource securitylake:CreateDataLakeExceptionSubscription securitylake:CreateDataLakeOrganizationConfiguration securitylake:CreateSubscriber securitylake:CreateSubscriberNotification securitylake:DeleteAwsLogSource securitylake:DeleteCustomLogSource securitylake:DeleteDataLakeExceptionSubscription securitylake:DeleteDataLakeOrganizationConfiguration securitylake:DeleteSubscriber securitylake:DeleteSubscriberNotification securitylake:DeregisterDataLakeDelegatedAdministrator securitylake:GetDataLakeExceptionSubscription securitylake:GetDataLakeOrganizationConfiguration securitylake:GetDataLakeSources securitylake:GetSubscriber securitylake:ListDataLakes securitylake:ListLogSources securitylake:ListSubscribers securitylake:RegisterDataLakeDelegatedAdministrator

서비스 접두사	작업
	securitylake:UpdateDataLakeExceptionSubscription securitylake:UpdateSubscriber securitylake:UpdateSubscriberNotification
serverlessrepo	serverlessrepo:CreateApplication serverlessrepo:CreateApplicationVersion serverlessrepo:CreateCloudFormationChangeSet serverlessrepo:CreateCloudFormationTemplate serverlessrepo:DeleteApplication serverlessrepo:GetApplication serverlessrepo:GetApplicationPolicy serverlessrepo:GetCloudFormationTemplate serverlessrepo:ListApplicationDependencies serverlessrepo:ListApplications serverlessrepo:ListApplicationVersions serverlessrepo:PutApplicationPolicy serverlessrepo:UnshareApplication serverlessrepo:UpdateApplication

서비스 접두사	작업
servicecatalog	servicecatalog:AcceptPortfolioShare servicecatalog:AssociateBudgetWithResource servicecatalog:AssociatePrincipalWithPortfolio servicecatalog:AssociateProductWithPortfolio servicecatalog:AssociateServiceActionWithProvisioningArtifact servicecatalog:BatchAssociateServiceActionWithProvisioningArtifact servicecatalog:BatchDisassociateServiceActionFromProvisioningArtifact servicecatalog:CopyProduct servicecatalog:CreateConstraint servicecatalog:CreatePortfolio servicecatalog:CreatePortfolioShare servicecatalog:CreateProduct servicecatalog:CreateProvisionedProductPlan servicecatalog:CreateProvisioningArtifact servicecatalog:CreateServiceAction servicecatalog:DeleteConstraint servicecatalog:DeletePortfolio servicecatalog:DeletePortfolioShare servicecatalog:DeleteProduct servicecatalog:DeleteProvisionedProductPlan

서비스 접두사	작업
	servicecatalog:DeleteProvisioningArtifact servicecatalog:DeleteServiceAction servicecatalog:DescribeConstraint servicecatalog:DescribeCopyProductStatus servicecatalog:DescribePortfolio servicecatalog:DescribePortfolioShares servicecatalog:DescribePortfolioShareStatus servicecatalog:DescribeProduct servicecatalog:DescribeProductAsAdmin servicecatalog:DescribeProductView servicecatalog:DescribeProvisionedProductPlan servicecatalog:DescribeProvisioningArtifact servicecatalog:DescribeProvisioningParameters servicecatalog:DescribeRecord servicecatalog:DescribeServiceAction servicecatalog:DescribeServiceActionExecutionParameters servicecatalog:DisableAWSOrganizationsAccess servicecatalog:DisassociateBudgetFromResource servicecatalog:DisassociatePrincipalFromPortfolio servicecatalog:DisassociateProductFromPortfolio servicecatalog:DisassociateServiceActionFromProvisioningArtifact

서비스 접두사	작업
	servicecatalog:EnableAWSOrganizationsAccess servicecatalog:ExecuteProvisionedProductPlan servicecatalog:ExecuteProvisionedProductServiceAction servicecatalog:GetAWSOrganizationsAccessStatus servicecatalog:GetProvisionedProductOutputs servicecatalog:ImportAsProvisionedProduct servicecatalog:ListAcceptedPortfolioShares servicecatalog:ListBudgetsForResource servicecatalog:ListConstraintsForPortfolio servicecatalog:ListLaunchPaths servicecatalog:ListOrganizationPortfolioAccess servicecatalog:ListPortfolioAccess servicecatalog:ListPortfolios servicecatalog:ListPortfoliosForProduct servicecatalog:ListPrincipalsForPortfolio servicecatalog:ListProvisionedProductPlans servicecatalog:ListProvisioningArtifacts servicecatalog:ListProvisioningArtifactsForServiceAction servicecatalog:ListRecordHistory servicecatalog:ListServiceActions servicecatalog:ListServiceActionsForProvisioningArtifact

서비스 접두사	작업
	servicecatalog:ListStackInstancesForProvisionedProduct servicecatalog:NotifyProvisionProductEngineWorkflowResult servicecatalog:NotifyTerminateProvisionedProductEngine WorkflowResult servicecatalog:NotifyUpdateProvisionedProductEngineWor kflowResult servicecatalog:ProvisionProduct servicecatalog:RejectPortfolioShare servicecatalog:ScanProvisionedProducts servicecatalog:SearchProducts servicecatalog:SearchProductsAsAdmin servicecatalog:SearchProvisionedProducts servicecatalog:TerminateProvisionedProduct servicecatalog:UpdateConstraint servicecatalog:UpdatePortfolio servicecatalog:UpdatePortfolioShare servicecatalog:UpdateProduct servicecatalog:UpdateProvisionedProduct servicecatalog:UpdateProvisionedProductProperties servicecatalog:UpdateProvisioningArtifact servicecatalog:UpdateServiceAction

서비스 접두사	작업
servicediscovery	servicediscovery:CreateHttpNamespace
	servicediscovery:CreatePrivateDnsNamespace
	servicediscovery:CreatePublicDnsNamespace
	servicediscovery:CreateService
	servicediscovery:DeleteNamespace
	servicediscovery:DeleteService
	servicediscovery:DeregisterInstance
	servicediscovery:GetInstance
	servicediscovery:GetInstancesHealthStatus
	servicediscovery:GetNamespace
	servicediscovery:GetOperation
	servicediscovery:GetService
	servicediscovery:ListInstances
	servicediscovery:ListNamespaces
	servicediscovery:ListOperations
	servicediscovery:ListServices
	servicediscovery:RegisterInstance
	servicediscovery:UpdateHttpNamespace
	servicediscovery:UpdateInstanceCustomHealthStatus
	servicediscovery:UpdatePrivateDnsNamespace
	servicediscovery:UpdatePublicDnsNamespace

서비스 접두사	작업
	servicediscovery:UpdateService
servicequotas	servicequotas:AssociateServiceQuotaTemplate servicequotas:DeleteServiceQuotaIncreaseRequestFromTemplate servicequotas:DisassociateServiceQuotaTemplate servicequotas:GetAssociationForServiceQuotaTemplate servicequotas:GetAWSDefaultServiceQuota servicequotas:GetRequestedServiceQuotaChange servicequotas:GetServiceQuota servicequotas:GetServiceQuotaIncreaseRequestFromTemplate servicequotas:ListAWSDefaultServiceQuotas servicequotas:ListRequestedServiceQuotaChangeHistory servicequotas:ListRequestedServiceQuotaChangeHistoryByQuota servicequotas:ListServiceQuotaIncreaseRequestsInTemplate servicequotas:ListServiceQuotas servicequotas:ListServices servicequotas:PutServiceQuotaIncreaseRequestIntoTemplate servicequotas:RequestServiceQuotaIncrease

서비스 접두사	작업
ses	ses:BatchGetMetricData
	ses:CloneReceiptRuleSet
	ses:CreateConfigurationSet
	ses:CreateConfigurationSetEventDestination
	ses:CreateConfigurationSetTrackingOptions
	ses:CreateContact
	ses:CreateContactList
	ses:CreateCustomVerificationEmailTemplate
	ses:CreateDedicatedIpPool
	ses:CreateDeliverabilityTestReport
	ses:CreateEmailIdentity
	ses:CreateEmailIdentityPolicy
	ses:CreateEmailTemplate
	ses:CreateImportJob
	ses:CreateReceiptFilter
	ses:CreateReceiptRule
	ses:CreateReceiptRuleSet
	ses:CreateTemplate
	ses:DeleteConfigurationSet
	ses:DeleteConfigurationSetEventDestination
	ses:DeleteConfigurationSetTrackingOptions

서비스 접두사	작업
	ses:DeleteContact
	ses:DeleteContactList
	ses:DeleteCustomVerificationEmailTemplate
	ses:DeleteDedicatedIpPool
	ses:DeleteEmailIdentity
	ses:DeleteEmailIdentityPolicy
	ses:DeleteEmailTemplate
	ses:DeleteIdentity
	ses:DeleteIdentityPolicy
	ses:DeleteReceiptFilter
	ses:DeleteReceiptRule
	ses:DeleteReceiptRuleSet
	ses:DeleteSuppressedDestination
	ses:DeleteTemplate
	ses:DeleteVerifiedEmailAddress
	ses:DescribeActiveReceiptRuleSet
	ses:DescribeConfigurationSet
	ses:DescribeReceiptRule
	ses:DescribeReceiptRuleSet
	ses:GetAccount
	ses:GetAccountSendingEnabled

서비스 접두사	작업
	ses:GetBlacklistReports
	ses:GetConfigurationSet
	ses:GetConfigurationSetEventDestinations
	ses:GetContact
	ses:GetContactList
	ses:GetCustomVerificationEmailTemplate
	ses:GetDedicatedIp
	ses:GetDedicatedIpPool
	ses:GetDedicatedIps
	ses:GetDeliverabilityDashboardOptions
	ses:GetDeliverabilityTestReport
	ses:GetDomainDeliverabilityCampaign
	ses:GetDomainStatisticsReport
	ses:GetEmailIdentity
	ses:GetEmailIdentityPolicies
	ses:GetEmailTemplate
	ses:GetIdentityDkimAttributes
	ses:GetIdentityMailFromDomainAttributes
	ses:GetIdentityNotificationAttributes
	ses:GetIdentityPolicies
	ses:GetIdentityVerificationAttributes

서비스 접두사	작업
	ses:GetImportJob
	ses:GetMessageInsights
	ses:GetSendQuota
	ses:GetSendStatistics
	ses:GetSuppressedDestination
	ses:GetTemplate
	ses:ListConfigurationSets
	ses:ListContactLists
	ses:ListContacts
	ses:ListCustomVerificationEmailTemplates
	ses:ListDedicatedIpPools
	ses:ListDeliverabilityTestReports
	ses:ListDomainDeliverabilityCampaigns
	ses:ListEmailIdentities
	ses:ListEmailTemplates
	ses:ListExportJobs
	ses:ListIdentities
	ses:ListIdentityPolicies
	ses:ListImportJobs
	ses:ListReceiptFilters
	ses:ListReceiptRuleSets

서비스 접두사	작업
	ses:ListRecommendations
	ses:ListSuppressedDestinations
	ses:ListTemplates
	ses:ListVerifiedEmailAddresses
	ses:PutAccountDedicatedIpWarmupAttributes
	ses:PutAccountDetails
	ses:PutAccountSendingAttributes
	ses:PutAccountSuppressionAttributes
	ses:PutAccountVdmAttributes
	ses:PutConfigurationSetDeliveryOptions
	ses:PutConfigurationSetReputationOptions
	ses:PutConfigurationSetSendingOptions
	ses:PutConfigurationSetSuppressionOptions
	ses:PutConfigurationSetTrackingOptions
	ses:PutConfigurationSetVdmOptions
	ses:PutDedicatedIpInPool
	ses:PutDedicatedIpPoolScalingAttributes
	ses:PutDedicatedIpWarmupAttributes
	ses:PutDeliverabilityDashboardOption
	ses:PutEmailIdentityConfigurationSetAttributes
	ses:PutEmailIdentityDkimAttributes

서비스 접두사	작업
	ses:PutEmailIdentityDkimSigningAttributes
	ses:PutEmailIdentityFeedbackAttributes
	ses:PutEmailIdentityMailFromAttributes
	ses:PutIdentityPolicy
	ses:PutSuppressedDestination
	ses:ReorderReceiptRuleSet
	ses:SendBounce
	ses:SendCustomVerificationEmail
	ses:SetActiveReceiptRuleSet
	ses:SetIdentityDkimEnabled
	ses:SetIdentityFeedbackForwardingEnabled
	ses:SetIdentityHeadersInNotificationsEnabled
	ses:SetIdentityMailFromDomain
	ses:SetIdentityNotificationTopic
	ses:SetReceiptRulePosition
	ses:TestRenderEmailTemplate
	ses:TestRenderTemplate
	ses:UpdateAccountSendingEnabled
	ses:UpdateConfigurationSetEventDestination
	ses:UpdateConfigurationSetReputationMetricsEnabled
	ses:UpdateConfigurationSetSendingEnabled

서비스 접두사	작업
	ses:UpdateConfigurationSetTrackingOptions
	ses:UpdateContact
	ses:UpdateContactList
	ses:UpdateCustomVerificationEmailTemplate
	ses:UpdateEmailIdentityPolicy
	ses:UpdateEmailTemplate
	ses:UpdateReceiptRule
	ses:UpdateTemplate
	ses:VerifyDomainDkim
	ses:VerifyDomainIdentity
	ses:VerifyEmailAddress
	ses:VerifyEmailIdentity

서비스 접두사	작업
shield	shield:AssociateDRTLogBucket shield:AssociateHealthCheck shield:AssociateProactiveEngagementDetails shield:CreateProtection shield:CreateProtectionGroup shield:CreateSubscription shield:DeleteProtection shield:DeleteProtectionGroup shield:DeleteSubscription shield:DescribeAttack shield:DescribeAttackStatistics shield:DescribeDRTAccess shield:DescribeEmergencyContactSettings shield:DescribeProtection shield:DescribeProtectionGroup shield:DescribeSubscription shield:DisableApplicationLayerAutomaticResponse shield:DisableProactiveEngagement shield:DisassociateDRTLogBucket shield:DisassociateDRTRole shield:DisassociateHealthCheck

서비스 접두사	작업
	shield:EnableApplicationLayerAutomaticResponse shield:EnableProactiveEngagement shield:GetSubscriptionState shield:ListAttacks shield:ListProtectionGroups shield:ListProtections shield:ListResourcesInProtectionGroup shield:UpdateApplicationLayerAutomaticResponse shield:UpdateEmergencyContactSettings shield:UpdateProtectionGroup shield:UpdateSubscription

서비스 접두사	작업
signer	signer:AddProfilePermission
	signer:CancelSigningProfile
	signer:DescribeSigningJob
	signer:GetRevocationStatus
	signer:GetSigningPlatform
	signer:GetSigningProfile
	signer:ListProfilePermissions
	signer:ListSigningJobs
	signer:ListSigningPlatforms
	signer:ListSigningProfiles
	signer:PutSigningProfile
	signer:RemoveProfilePermission
	signer:RevokeSignature
	signer:RevokeSigningProfile
	signer:SignPayload
	signer:StartSigningJob

서비스 접두사	작업
simspaceweaver	simspaceweaver:CreateSnapshot simspaceweaver:DeleteApp simspaceweaver:DeleteSimulation simspaceweaver:DescribeApp simspaceweaver:DescribeSimulation simspaceweaver:ListApps simspaceweaver:ListSimulations simspaceweaver:StartApp simspaceweaver:StartClock simspaceweaver:StartSimulation simspaceweaver:StopApp simspaceweaver:StopClock simspaceweaver:StopSimulation

서비스 접두사	작업
sms	sms:CreateApp
	sms:CreateReplicationJob
	sms:DeleteApp
	sms:DeleteAppLaunchConfiguration
	sms:DeleteAppReplicationConfiguration
	sms:DeleteAppValidationConfiguration
	sms:DeleteReplicationJob
	sms:DeleteServerCatalog
	sms:DisassociateConnector
	sms:GenerateChangeSet
	sms:GenerateTemplate
	sms:GetApp
	sms:GetAppLaunchConfiguration
	sms:GetAppReplicationConfiguration
	sms:GetAppValidationConfiguration
	sms:GetAppValidationOutput
	sms:GetConnectors
	sms:GetReplicationJobs
	sms:GetReplicationRuns
	sms:GetServers
	sms:ImportAppCatalog

서비스 접두사	작업
	sms:ImportServerCatalog
	sms:LaunchApp
	sms:ListApps
	sms:NotifyAppValidationOutput
	sms:PutAppLaunchConfiguration
	sms:PutAppReplicationConfiguration
	sms:PutAppValidationConfiguration
	sms:StartAppReplication
	sms:StartOnDemandAppReplication
	sms:StartOnDemandReplicationRun
	sms:StopAppReplication
	sms:TerminateApp
	sms:UpdateApp
	sms:UpdateReplicationJob

서비스 접두사	작업
sms-voice	sms-voice:AssociateProtectConfiguration sms-voice:CreateConfigurationSet sms-voice:CreateConfigurationSetEventDestination sms-voice:CreateEventDestination sms-voice:CreateOptOutList sms-voice:CreatePool sms-voice:CreateProtectConfiguration sms-voice:CreateRegistration sms-voice:CreateRegistrationAssociation sms-voice:CreateRegistrationAttachment sms-voice:CreateRegistrationVersion sms-voice:CreateVerifiedDestinationNumber sms-voice:DeleteAccountDefaultProtectConfiguration sms-voice:DeleteConfigurationSet sms-voice:DeleteConfigurationSetEventDestination sms-voice:DeleteDefaultMessageType sms-voice:DeleteDefaultSenderId sms-voice:DeleteEventDestination sms-voice:DeleteKeyword sms-voice:DeleteMediaMessageSpendLimitOverride sms-voice:DeleteOptedOutNumber

서비스 접두사	작업
	sms-voice:DeleteOptOutList sms-voice:DeletePool sms-voice:DeleteProtectConfiguration sms-voice:DeleteRegistration sms-voice:DeleteRegistrationAttachment sms-voice:DeleteTextMessageSpendLimitOverride sms-voice:DeleteVerifiedDestinationNumber sms-voice:DeleteVoiceMessageSpendLimitOverride sms-voice:DescribeAccountAttributes sms-voice:DescribeAccountLimits sms-voice:DescribeConfigurationSets sms-voice:DescribeKeywords sms-voice:DescribeOptedOutNumbers sms-voice:DescribeOptOutLists sms-voice:DescribePhoneNumbers sms-voice:DescribePools sms-voice:DescribeProtectConfigurations sms-voice:DescribeRegistrationAttachments sms-voice:DescribeRegistrationFieldDefinitions sms-voice:DescribeRegistrationFieldValues sms-voice:DescribeRegistrations

서비스 접두사	작업
	sms-voice:DescribeRegistrationSectionDefinitions sms-voice:DescribeRegistrationTypeDefinitions sms-voice:DescribeRegistrationVersions sms-voice:DescribeSenderIds sms-voice:DescribeSpendLimits sms-voice:DescribeVerifiedDestinationNumbers sms-voice:DisassociateOriginationIdentity sms-voice:DisassociateProtectConfiguration sms-voice:DiscardRegistrationVersion sms-voice:GetConfigurationSetEventDestinations sms-voice:GetProtectConfigurationCountryRuleSet sms-voice:ListConfigurationSets sms-voice:ListPoolOriginationIdentities sms-voice:ListRegistrationAssociations sms-voice:PutKeyword sms-voice:PutOptedOutNumber sms-voice:ReleasePhoneNumber sms-voice:ReleaseSenderId sms-voice:RequestPhoneNumber sms-voice:RequestSenderId sms-voice:SendDestinationNumberVerificationCode

서비스 접두사	작업
	sms-voice:SetAccountDefaultProtectConfiguration sms-voice:SetDefaultMessageType sms-voice:SetDefaultSenderId sms-voice:SetMediaMessageSpendLimitOverride sms-voice:SetTextMessageSpendLimitOverride sms-voice:SetVoiceMessageSpendLimitOverride sms-voice:SubmitRegistrationVersion sms-voice:UpdateConfigurationSetEventDestination sms-voice:UpdateEventDestination sms-voice:UpdatePhoneNumber sms-voice:UpdatePool sms-voice:UpdateProtectConfiguration sms-voice:UpdateProtectConfigurationCountryRuleSet sms-voice:UpdateSenderId

서비스 접두사	작업
snowball	snowball:CancelCluster
	snowball:CancelJob
	snowball:CreateAddress
	snowball:CreateCluster
	snowball:CreateJob
	snowball:CreateLongTermPricing
	snowball:CreateReturnShippingLabel
	snowball:DescribeAddress
	snowball:DescribeAddresses
	snowball:DescribeCluster
	snowball:DescribeJob
	snowball:DescribeReturnShippingLabel
	snowball:GetJobManifest
	snowball:GetJobUnlockCode
	snowball:GetSnowballUsage
	snowball:GetSoftwareUpdates
	snowball:ListClusterJobs
	snowball:ListClusters
	snowball:ListCompatibleImages
	snowball:ListJobs
	snowball:ListLongTermPricing

서비스 접두사	작업
	snowball:ListPickupLocations snowball:ListServiceVersions snowball:UpdateCluster snowball:UpdateJob snowball:UpdateJobShipmentState snowball:UpdateLongTermPricing
sqs	sqs:AddPermission sqs:CancelMessageMoveTask sqs:CreateQueue sqs:DeleteQueue sqs:PurgeQueue sqs:RemovePermission sqs:SetQueueAttributes

서비스 접두사	작업
ssm	ssm:AssociateOpsItemRelatedItem
	ssm:CancelCommand
	ssm:CancelMaintenanceWindowExecution
	ssm:CreateActivation
	ssm:CreateAssociation
	ssm:CreateAssociationBatch
	ssm:CreateDocument
	ssm:CreateMaintenanceWindow
	ssm:CreateOpsItem
	ssm:CreateOpsMetadata
	ssm:CreatePatchBaseline
	ssm:CreateResourceDataSync
	ssm:DeleteActivation
	ssm:DeleteAssociation
	ssm:DeleteDocument
	ssm:DeleteInventory
	ssm:DeleteMaintenanceWindow
	ssm:DeleteOpsItem
	ssm:DeleteOpsMetadata
	ssm:DeleteParameter
	ssm:DeleteParameters

서비스 접두사	작업
	ssm:DeletePatchBaseline
	ssm:DeleteResourceDataSync
	ssm:DeleteResourcePolicy
	ssm:DeregisterManagedInstance
	ssm:DeregisterPatchBaselineForPatchGroup
	ssm:DeregisterTargetFromMaintenanceWindow
	ssm:DeregisterTaskFromMaintenanceWindow
	ssm:DescribeActivations
	ssm:DescribeAssociation
	ssm:DescribeAssociationExecutions
	ssm:DescribeAssociationExecutionTargets
	ssm:DescribeAutomationExecutions
	ssm:DescribeAutomationStepExecutions
	ssm:DescribeAvailablePatches
	ssm:DescribeDocument
	ssm:DescribeDocumentParameters
	ssm:DescribeDocumentPermission
	ssm:DescribeEffectiveInstanceAssociations
	ssm:DescribeEffectivePatchesForPatchBaseline
	ssm:DescribeInstanceAssociationsStatus
	ssm:DescribeInstanceInformation

서비스 접두사	작업
	ssm:DescribeInstancePatches
	ssm:DescribeInstancePatchStates
	ssm:DescribeInstancePatchStatesForPatchGroup
	ssm:DescribeInstanceProperties
	ssm:DescribeInventoryDeletions
	ssm:DescribeMaintenanceWindowExecutions
	ssm:DescribeMaintenanceWindowExecutionTaskInvocations
	ssm:DescribeMaintenanceWindowExecutionTasks
	ssm:DescribeMaintenanceWindows
	ssm:DescribeMaintenanceWindowSchedule
	ssm:DescribeMaintenanceWindowsForTarget
	ssm:DescribeMaintenanceWindowTargets
	ssm:DescribeMaintenanceWindowTasks
	ssm:DescribeOpsItems
	ssm:DescribeParameters
	ssm:DescribePatchBaselines
	ssm:DescribePatchGroups
	ssm:DescribePatchGroupState
	ssm:DescribePatchProperties
	ssm:DescribeSessions
	ssm:DisassociateOpsItemRelatedItem

서비스 접두사	작업
	ssm:GetAutomationExecution
	ssm:GetCalendarState
	ssm:GetCommandInvocation
	ssm:GetConnectionStatus
	ssm:GetDefaultPatchBaseline
	ssm:GetDeployablePatchSnapshotForInstance
	ssm:GetDocument
	ssm:GetInventory
	ssm:GetInventorySchema
	ssm:GetMaintenanceWindow
	ssm:GetMaintenanceWindowExecution
	ssm:GetMaintenanceWindowExecutionTask
	ssm:GetMaintenanceWindowExecutionTaskInvocation
	ssm:GetMaintenanceWindowTask
	ssm:GetOpsItem
	ssm:GetOpsMetadata
	ssm:GetOpsSummary
	ssm:GetParameter
	ssm:GetParameterHistory
	ssm:GetParameters
	ssm:GetParametersByPath

서비스 접두사	작업
	ssm:GetPatchBaseline
	ssm:GetPatchBaselineForPatchGroup
	ssm:GetResourcePolicies
	ssm:GetServiceSetting
	ssm:LabelParameterVersion
	ssm:ListAssociations
	ssm:ListAssociationVersions
	ssm:ListCommandInvocations
	ssm:ListCommands
	ssm:ListComplianceItems
	ssm:ListComplianceSummaries
	ssm:ListDocumentMetadataHistory
	ssm:ListDocuments
	ssm:ListDocumentVersions
	ssm:ListInstanceAssociations
	ssm:ListInventoryEntries
	ssm:ListOpsItemEvents
	ssm:ListOpsItemRelatedItems
	ssm:ListOpsMetadata
	ssm:ListResourceComplianceSummaries
	ssm:ListResourceDataSync

서비스 접두사	작업
	ssm:ModifyDocumentPermission
	ssm:PutComplianceItems
	ssm:PutInventory
	ssm:PutParameter
	ssm:PutResourcePolicy
	ssm:RegisterDefaultPatchBaseline
	ssm:RegisterManagedInstance
	ssm:RegisterPatchBaselineForPatchGroup
	ssm:RegisterTargetWithMaintenanceWindow
	ssm:RegisterTaskWithMaintenanceWindow
	ssm:ResetServiceSetting
	ssm:ResumeSession
	ssm:SendAutomationSignal
	ssm:SendCommand
	ssm:StartAssociationsOnce
	ssm:StartAutomationExecution
	ssm:StartChangeRequestExecution
	ssm:StartSession
	ssm:StopAutomationExecution
	ssm:TerminateSession
	ssm:UnlabelParameterVersion

서비스 접두사	작업
	ssm:UpdateAssociation
	ssm:UpdateAssociationStatus
	ssm:UpdateDocument
	ssm:UpdateDocumentDefaultVersion
	ssm:UpdateDocumentMetadata
	ssm:UpdateInstanceInformation
	ssm:UpdateMaintenanceWindow
	ssm:UpdateMaintenanceWindowTarget
	ssm:UpdateMaintenanceWindowTask
	ssm:UpdateManagedInstanceRole
	ssm:UpdateOpsItem
	ssm:UpdateOpsMetadata
	ssm:UpdatePatchBaseline
	ssm:UpdateResourceDataSync
	ssm:UpdateServiceSetting

서비스 접두사	작업
ssm-incidents	ssm-incidents:BatchGetIncidentFindings ssm-incidents:CreateReplicationSet ssm-incidents:CreateResponsePlan ssm-incidents:CreateTimelineEvent ssm-incidents:DeleteIncidentRecord ssm-incidents:DeleteReplicationSet ssm-incidents:DeleteResourcePolicy ssm-incidents:DeleteResponsePlan ssm-incidents:DeleteTimelineEvent ssm-incidents:GetIncidentRecord ssm-incidents:GetReplicationSet ssm-incidents:GetResourcePolicies ssm-incidents:GetResponsePlan ssm-incidents:GetTimelineEvent ssm-incidents:ListIncidentFindings ssm-incidents:ListIncidentRecords ssm-incidents:ListRelatedItems ssm-incidents:ListReplicationSets ssm-incidents:ListResponsePlans ssm-incidents:ListTimelineEvents ssm-incidents:PutResourcePolicy

서비스 접두사	작업
	ssm-incidents:StartIncident ssm-incidents:UpdateDeletionProtection ssm-incidents:UpdateIncidentRecord ssm-incidents:UpdateRelatedItems ssm-incidents:UpdateReplicationSet ssm-incidents:UpdateResponsePlan ssm-incidents:UpdateTimelineEvent

서비스 접두사	작업
ssm-sap	ssm-sap:BackupDatabase
	ssm-sap:DeleteResourcePermission
	ssm-sap:DeregisterApplication
	ssm-sap:GetApplication
	ssm-sap:GetComponent
	ssm-sap:GetDatabase
	ssm-sap:GetOperation
	ssm-sap:GetResourcePermission
	ssm-sap:ListApplications
	ssm-sap:ListComponents
	ssm-sap:ListDatabases
	ssm-sap:ListOperationEvents
	ssm-sap:ListOperations
	ssm-sap:PutResourcePermission
	ssm-sap:RegisterApplication
	ssm-sap:RestoreDatabase
	ssm-sap:StartApplication
	ssm-sap:StartApplicationRefresh
	ssm-sap:StopApplication
	ssm-sap:UpdateApplicationSettings
	ssm-sap:UpdateHANABackupSettings

서비스 접두사	작업
states	states:CreateActivity states:CreateStateMachine states:CreateStateMachineAlias states:DeleteActivity states:DeleteStateMachine states:DeleteStateMachineAlias states:DeleteStateMachineVersion states:DescribeActivity states:DescribeExecution states:DescribeMapRun states:DescribeStateMachine states:DescribeStateMachineAlias states:DescribeStateMachineForExecution states:GetExecutionHistory states:ListActivities states:ListExecutions states:ListMapRuns states:ListStateMachineAliases states:ListStateMachines states:ListStateMachineVersions states:SendTaskFailure

서비스 접두사	작업
	states:SendTaskHeartbeat states:SendTaskSuccess states:StartExecution states:StopExecution states:UpdateMapRun states:UpdateStateMachine states:UpdateStateMachineAlias states:ValidateStateMachineDefinition
sts	sts:AssumeRole sts:AssumeRoleWithSAML sts:AssumeRoleWithWebIdentity sts:DecodeAuthorizationMessage sts:GetAccessKeyInfo sts:GetCallerIdentity sts:GetFederationToken sts:GetSessionToken

서비스 접두사	작업
swf	swf:DeleteActivityType swf:DeleteWorkflowType swf:DeprecateActivityType swf:DeprecateDomain swf:DeprecateWorkflowType swf:DescribeActivityType swf:DescribeDomain swf:DescribeWorkflowType swf:ListActivityTypes swf:ListDomains swf:ListWorkflowTypes swf:RegisterActivityType swf:RegisterDomain swf:RegisterWorkflowType swf:UndeprecateActivityType swf:UndeprecateDomain swf:UndeprecateWorkflowType

서비스 접두사	작업
synthetics	synthetics:AssociateResource
	synthetics:CreateCanary
	synthetics:CreateGroup
	synthetics:DeleteCanary
	synthetics:DeleteGroup
	synthetics:DescribeCanaries
	synthetics:DescribeCanariesLastRun
	synthetics:DescribeRuntimeVersions
	synthetics:DisassociateResource
	synthetics:GetCanary
	synthetics:GetCanaryRuns
	synthetics:GetGroup
	synthetics:ListAssociatedGroups
	synthetics:ListGroupResources
	synthetics:ListGroups
	synthetics:StartCanary
	synthetics:StopCanary
	synthetics:UpdateCanary

서비스 접두사	작업
tag	tag:DescribeReportCreation tag:GetComplianceSummary tag:GetResources tag:StartReportCreation

서비스 접두사	작업
textract	textract:AnalyzeDocument
	textract:AnalyzeExpense
	textract:AnalyzeID
	textract:CreateAdapter
	textract:CreateAdapterVersion
	textract:DeleteAdapter
	textract:DeleteAdapterVersion
	textract:DetectDocumentText
	textract:GetAdapter
	textract:GetAdapterVersion
	textract:GetDocumentAnalysis
	textract:GetDocumentTextDetection
	textract:GetExpenseAnalysis
	textract:GetLendingAnalysis
	textract:GetLendingAnalysisSummary
	textract:ListAdapters
	textract:ListAdapterVersions
	textract:StartDocumentAnalysis
	textract:StartDocumentTextDetection
	textract:StartExpenseAnalysis
	textract:StartLendingAnalysis

서비스 접두사	작업
	textract:UpdateAdapter

서비스 접두사	작업
timestream	timestream:CancelQuery timestream:CreateDatabase timestream:CreateScheduledQuery timestream:CreateTable timestream:DeleteDatabase timestream:DeleteScheduledQuery timestream:DeleteTable timestream:DescribeAccountSettings timestream:DescribeDatabase timestream:DescribeScheduledQuery timestream:DescribeTable timestream:ExecuteScheduledQuery timestream:ListBatchLoadTasks timestream:ListDatabases timestream:ListScheduledQueries timestream:ListTables timestream:PrepareQuery timestream:UpdateAccountSettings timestream:UpdateDatabase timestream:UpdateScheduledQuery timestream:UpdateTable

서비스 접두사	작업
tnb	tnb:CancelSolNetworkOperation
	tnb:CreateSolFunctionPackage
	tnb:CreateSolNetworkInstance
	tnb:CreateSolNetworkPackage
	tnb:DeleteSolFunctionPackage
	tnb:DeleteSolNetworkInstance
	tnb:DeleteSolNetworkPackage
	tnb:GetSolFunctionInstance
	tnb:GetSolFunctionPackage
	tnb:GetSolFunctionPackageContent
	tnb:GetSolFunctionPackageDescriptor
	tnb:GetSolNetworkInstance
	tnb:GetSolNetworkOperation
	tnb:GetSolNetworkPackage
	tnb:GetSolNetworkPackageContent
	tnb:GetSolNetworkPackageDescriptor
	tnb:InstantiateSolNetworkInstance
	tnb:ListSolFunctionInstances
	tnb:ListSolFunctionPackages
	tnb:ListSolNetworkInstances
	tnb:ListSolNetworkOperations

서비스 접두사	작업
	tnb:ListSolNetworkPackages
	tnb:PutSolFunctionPackageContent
	tnb:PutSolNetworkPackageContent
	tnb:TerminateSolNetworkInstance
	tnb:UpdateSolFunctionPackage
	tnb:UpdateSolNetworkInstance
	tnb:UpdateSolNetworkPackage
	tnb:ValidateSolFunctionPackageContent
	tnb:ValidateSolNetworkPackageContent

서비스 접두사	작업
transcribe	transcribe:CreateCallAnalyticsCategory transcribe:CreateLanguageModel transcribe:CreateMedicalVocabulary transcribe:CreateVocabulary transcribe:CreateVocabularyFilter transcribe:DeleteCallAnalyticsCategory transcribe:DeleteCallAnalyticsJob transcribe:DeleteLanguageModel transcribe:DeleteMedicalScribeJob transcribe:DeleteMedicalTranscriptionJob transcribe:DeleteMedicalVocabulary transcribe:DeleteTranscriptionJob transcribe:DeleteVocabulary transcribe:DeleteVocabularyFilter transcribe:DescribeLanguageModel transcribe:GetCallAnalyticsCategory transcribe:GetCallAnalyticsJob transcribe:GetMedicalScribeJob transcribe:GetMedicalTranscriptionJob transcribe:GetMedicalVocabulary transcribe:GetTranscriptionJob

서비스 접두사	작업
	transcribe:GetVocabulary transcribe:GetVocabularyFilter transcribe:ListCallAnalyticsCategories transcribe:ListCallAnalyticsJobs transcribe:ListLanguageModels transcribe:ListMedicalScribeJobs transcribe:ListMedicalTranscriptionJobs transcribe:ListMedicalVocabularies transcribe:ListTranscriptionJobs transcribe:ListVocabularies transcribe:ListVocabularyFilters transcribe:StartCallAnalyticsJob transcribe:StartCallAnalyticsStreamTranscription transcribe:StartCallAnalyticsStreamTranscriptionWebSocket transcribe:StartMedicalScribeJob transcribe:StartMedicalStreamTranscription transcribe:StartMedicalStreamTranscriptionWebSocket transcribe:StartMedicalTranscriptionJob transcribe:StartStreamTranscription transcribe:StartStreamTranscriptionWebSocket transcribe:StartTranscriptionJob

서비스 접두사	작업
	transcribe:UpdateCallAnalyticsCategory transcribe:UpdateMedicalVocabulary transcribe:UpdateVocabulary transcribe:UpdateVocabularyFilter

서비스 접두사	작업
전송	transfer:CreateAccess
	transfer:CreateAgreement
	transfer:CreateConnector
	transfer:CreateProfile
	transfer:CreateServer
	transfer:CreateUser
	transfer:CreateWorkflow
	transfer:DeleteAccess
	transfer:DeleteAgreement
	transfer:DeleteCertificate
	transfer:DeleteConnector
	transfer:DeleteHostKey
	transfer:DeleteProfile
	transfer:DeleteServer
	transfer:DeleteSshPublicKey
	transfer:DeleteUser
	transfer:DeleteWorkflow
	transfer:DescribeAccess
	transfer:DescribeAgreement
	transfer:DescribeCertificate
	transfer:DescribeConnector

서비스 접두사	작업
	transfer:DescribeExecution
	transfer:DescribeHostKey
	transfer:DescribeProfile
	transfer:DescribeSecurityPolicy
	transfer:DescribeServer
	transfer:DescribeUser
	transfer:DescribeWorkflow
	transfer:ImportCertificate
	transfer:ImportHostKey
	transfer:ImportSshPublicKey
	transfer:ListAccesses
	transfer:ListCertificates
	transfer:ListConnectors
	transfer:ListExecutions
	transfer:ListHostKeys
	transfer:ListProfiles
	transfer:ListSecurityPolicies
	transfer:ListServers
	transfer:ListUsers
	transfer:ListWorkflows
	transfer:SendWorkflowStepState

서비스 접두사	작업
	transfer:StartDirectoryListing
	transfer:StartFileTransfer
	transfer:StartServer
	transfer:StopServer
	transfer:TestConnection
	transfer:TestIdentityProvider
	transfer:UpdateAccess
	transfer:UpdateAgreement
	transfer:UpdateCertificate
	transfer:UpdateConnector
	transfer:UpdateHostKey
	transfer:UpdateProfile
	transfer:UpdateServer
	transfer:UpdateUser

서비스 접두사	작업
translate	translate:CreateParallelData translate:DeleteParallelData translate:DeleteTerminology translate:DescribeTextTranslationJob translate:GetParallelData translate:GetTerminology translate:ImportTerminology translate:ListLanguages translate:ListParallelData translate:ListTerminologies translate:ListTextTranslationJobs translate:StartTextTranslationJob translate:StopTextTranslationJob translate:TranslateDocument translate:TranslateText translate:UpdateParallelData

서비스 접두사	작업
voiceid	voiceid:AssociateFraudster
	voiceid:CreateDomain
	voiceid:CreateWatchlist
	voiceid:DeleteDomain
	voiceid:DeleteFraudster
	voiceid:DeleteSpeaker
	voiceid:DeleteWatchlist
	voiceid:DescribeDomain
	voiceid:DescribeFraudster
	voiceid:DescribeFraudsterRegistrationJob
	voiceid:DescribeSpeaker
	voiceid:DescribeSpeakerEnrollmentJob
	voiceid:DescribeWatchlist
	voiceid:DisassociateFraudster
	voiceid:EvaluateSession
	voiceid:ListDomains
	voiceid:ListFraudsterRegistrationJobs
	voiceid:ListFraudsters
	voiceid:ListSpeakerEnrollmentJobs
	voiceid:ListSpeakers
	voiceid:ListWatchlists

서비스 접두사	작업
	voiceid:OptOutSpeaker voiceid:StartFraudsterRegistrationJob voiceid:StartSpeakerEnrollmentJob voiceid:UpdateDomain voiceid:UpdateWatchlist

서비스 접두사	작업
vpc-lattice	vpc-lattice:CreateAccessLogSubscription vpc-lattice:CreateListener vpc-lattice:CreateRule vpc-lattice:CreateService vpc-lattice:CreateServiceNetwork vpc-lattice:CreateServiceNetworkServiceAssociation vpc-lattice:CreateServiceNetworkVpcAssociation vpc-lattice:CreateTargetGroup vpc-lattice:DeleteAccessLogSubscription vpc-lattice:DeleteAuthPolicy vpc-lattice:DeleteListener vpc-lattice:DeleteResourcePolicy vpc-lattice:DeleteRule vpc-lattice:DeleteService vpc-lattice:DeleteServiceNetwork vpc-lattice:DeleteServiceNetworkServiceAssociation vpc-lattice:DeleteServiceNetworkVpcAssociation vpc-lattice:DeleteTargetGroup vpc-lattice:DeregisterTargets vpc-lattice:GetAccessLogSubscription vpc-lattice:GetAuthPolicy

서비스 접두사	작업
	vpc-lattice:GetListener vpc-lattice:GetResourcePolicy vpc-lattice:GetRule vpc-lattice:GetService vpc-lattice:GetServiceNetwork vpc-lattice:GetServiceNetworkServiceAssociation vpc-lattice:GetServiceNetworkVpcAssociation vpc-lattice:GetTargetGroup vpc-lattice:ListAccessLogSubscriptions vpc-lattice:ListListeners vpc-lattice:ListRules vpc-lattice:ListServiceNetworks vpc-lattice:ListServiceNetworkServiceAssociations vpc-lattice:ListServiceNetworkVpcAssociations vpc-lattice:ListServices vpc-lattice:ListTargetGroups vpc-lattice:ListTargets vpc-lattice:PutAuthPolicy vpc-lattice:PutResourcePolicy vpc-lattice:RegisterTargets vpc-lattice:UpdateAccessLogSubscription

서비스 접두사	작업
	vpc-lattice:UpdateListener vpc-lattice:UpdateRule vpc-lattice:UpdateService vpc-lattice:UpdateServiceNetwork vpc-lattice:UpdateServiceNetworkVpcAssociation vpc-lattice:UpdateTargetGroup

서비스 접두사	작업
wafv2	wafv2:AssociateWebACL
	wafv2:CheckCapacity
	wafv2:CreateAPIKey
	wafv2:CreateIPSet
	wafv2:CreateRegexPatternSet
	wafv2:CreateRuleGroup
	wafv2:CreateWebACL
	wafv2:DeleteAPIKey
	wafv2:DeleteFirewallManagerRuleGroups
	wafv2:DeleteIPSet
	wafv2:DeleteLoggingConfiguration
	wafv2:DeletePermissionPolicy
	wafv2:DeleteRegexPatternSet
	wafv2:DeleteRuleGroup
	wafv2:DeleteWebACL
	wafv2:DescribeAllManagedProducts
	wafv2:DescribeManagedProductsByVendor
	wafv2:DescribeManagedRuleGroup
	wafv2:DisassociateWebACL
	wafv2:GenerateMobileSdkReleaseUrl
	wafv2:GetDecryptedAPIKey

서비스 접두사	작업
	wafv2:GetIPSet
	wafv2:GetLoggingConfiguration
	wafv2:GetManagedRuleSet
	wafv2:GetMobileSdkRelease
	wafv2:GetPermissionPolicy
	wafv2:GetRateBasedStatementManagedKeys
	wafv2:GetRegexPatternSet
	wafv2:GetRuleGroup
	wafv2:GetSampledRequests
	wafv2:GetWebACLForResource
	wafv2:ListAPIKeys
	wafv2:ListAvailableManagedRuleGroups
	wafv2:ListAvailableManagedRuleGroupVersions
	wafv2:ListIPSets
	wafv2:ListLoggingConfigurations
	wafv2:ListManagedRuleSets
	wafv2:ListMobileSdkReleases
	wafv2:ListRegexPatternSets
	wafv2:ListResourcesForWebACL
	wafv2:ListRuleGroups
	wafv2:ListWebACLs

서비스 접두사	작업
	wafv2:PutLoggingConfiguration wafv2:PutManagedRuleSetVersions wafv2:PutPermissionPolicy wafv2:UpdateIPSet wafv2:UpdateManagedRuleSetVersionExpiryDate wafv2:UpdateRegexPatternSet wafv2:UpdateRuleGroup wafv2:UpdateWebACL

서비스 접두사	작업
wellarchitected	wellarchitected:AssociateLenses wellarchitected:AssociateProfiles wellarchitected:CreateLensShare wellarchitected:CreateLensVersion wellarchitected:CreateMilestone wellarchitected:CreateProfile wellarchitected:CreateProfileShare wellarchitected:CreateReviewTemplate wellarchitected:CreateWorkload wellarchitected:CreateWorkloadShare wellarchitected:DeleteLens wellarchitected:DeleteLensShare wellarchitected:DeleteProfile wellarchitected:DeleteProfileShare wellarchitected:DeleteReviewTemplate wellarchitected:DeleteTemplateShare wellarchitected:DeleteWorkload wellarchitected:DeleteWorkloadShare wellarchitected:DisassociateLenses wellarchitected:DisassociateProfiles wellarchitected:ExportLens

서비스 접두사	작업
	wellarchitected:GetAnswer wellarchitected:GetConsolidatedReport wellarchitected:GetGlobalSettings wellarchitected:GetLens wellarchitected:GetLensReview wellarchitected:GetLensReviewReport wellarchitected:GetLensVersionDifference wellarchitected:GetMilestone wellarchitected:GetProfile wellarchitected:GetProfileTemplate wellarchitected:GetReviewTemplate wellarchitected:GetReviewTemplateAnswer wellarchitected:GetReviewTemplateLensReview wellarchitected:GetWorkload wellarchitected:ImportLens wellarchitected:ListAnswers wellarchitected:ListCheckDetails wellarchitected:ListCheckSummaries wellarchitected:ListLenses wellarchitected:ListLensReviewImprovements wellarchitected:ListLensReviews

서비스 접두사	작업
	wellarchitected:ListLensShares wellarchitected:ListMilestones wellarchitected:ListNotifications wellarchitected:ListProfileNotifications wellarchitected:ListProfiles wellarchitected:ListProfileShares wellarchitected:ListReviewTemplateAnswers wellarchitected:ListReviewTemplates wellarchitected:ListShareInvitations wellarchitected:ListTemplateShares wellarchitected:ListWorkloads wellarchitected:ListWorkloadShares wellarchitected:UpdateAnswer wellarchitected:UpdateGlobalSettings wellarchitected:UpdateIntegration wellarchitected:UpdateLensReview wellarchitected:UpdateProfile wellarchitected:UpdateReviewTemplate wellarchitected:UpdateReviewTemplateLensReview wellarchitected:UpdateShareInvitation wellarchitected:UpdateWorkload

서비스 접두사	작업
	wellarchitected:UpdateWorkloadShare wellarchitected:UpgradeLensReview wellarchitected:UpgradeProfileVersion wellarchitected:UpgradeReviewTemplateLensReview

서비스 접두사	작업
wisdom	wisdom:CreateAssistant
	wisdom:CreateAssistantAssociation
	wisdom:CreateContent
	wisdom:CreateKnowledgeBase
	wisdom:CreateQuickResponse
	wisdom:CreateSession
	wisdom:DeleteAssistant
	wisdom:DeleteAssistantAssociation
	wisdom:DeleteContent
	wisdom:DeleteImportJob
	wisdom:DeleteKnowledgeBase
	wisdom:DeleteQuickResponse
	wisdom:GetAssistant
	wisdom:GetAssistantAssociation
	wisdom:GetContent
	wisdom:GetContentSummary
	wisdom:GetImportJob
	wisdom:GetKnowledgeBase
	wisdom:GetRecommendations
	wisdom:GetSession
	wisdom:ListAssistantAssociations

서비스 접두사	작업
	wisdom:ListAssistants
	wisdom:ListContents
	wisdom:ListImportJobs
	wisdom:ListKnowledgeBases
	wisdom:ListQuickResponses
	wisdom:NotifyRecommendationsReceived
	wisdom:QueryAssistant
	wisdom:RemoveKnowledgeBaseTemplateUri
	wisdom:SearchContent
	wisdom:SearchQuickResponses
	wisdom:SearchSessions
	wisdom:StartContentUpload
	wisdom:StartImportJob
	wisdom:UpdateContent
	wisdom:UpdateKnowledgeBaseTemplateUri
	wisdom:UpdateQuickResponse
	wisdom:UpdateSession

서비스 접두사	작업
worklink	worklink:AssociateDomain worklink:AssociateWebsiteAuthorizationProvider worklink:AssociateWebsiteCertificateAuthority worklink:CreateFleet worklink:DeleteFleet worklink:DescribeAuditStreamConfiguration worklink:DescribeCompanyNetworkConfiguration worklink:DescribeDevice worklink:DescribeDevicePolicyConfiguration worklink:DescribeDomain worklink:DescribeFleetMetadata worklink:DescribeIdentityProviderConfiguration worklink:DescribeWebsiteCertificateAuthority worklink:DisassociateDomain worklink:DisassociateWebsiteAuthorizationProvider worklink:DisassociateWebsiteCertificateAuthority worklink:ListDevices worklink:ListDomains worklink:ListFleets worklink:ListWebsiteAuthorizationProviders worklink:ListWebsiteCertificateAuthorities

서비스 접두사	작업
	worklink:RestoreDomainAccess worklink:RevokeDomainAccess worklink:SignOutUser worklink:UpdateAuditStreamConfiguration worklink:UpdateCompanyNetworkConfiguration worklink:UpdateDevicePolicyConfiguration worklink:UpdateDomainMetadata worklink:UpdateFleetMetadata worklink:UpdateIdentityProviderConfiguration

서비스 접두사	작업
WorkSpaces	workspaces:AcceptAccountLinkInvitation
	workspaces:AssociateConnectionAlias
	workspaces:AssociateIpGroups
	workspaces:AssociateWorkspaceApplication
	workspaces:CopyWorkspaceImage
	workspaces:CreateAccountLinkInvitation
	workspaces:CreateConnectClientAddIn
	workspaces:CreateConnectionAlias
	workspaces:CreateIpGroup
	workspaces:CreateStandbyWorkspaces
	workspaces:CreateUpdatedWorkspaceImage
	workspaces:CreateWorkspaceBundle
	workspaces:CreateWorkspaceImage
	workspaces:CreateWorkspaces
	workspaces:DeleteAccountLinkInvitation
	workspaces:DeleteClientBranding
	workspaces:DeleteConnectClientAddIn
	workspaces:DeleteConnectionAlias
	workspaces:DeleteIpGroup
	workspaces:DeleteWorkspaceBundle
	workspaces:DeleteWorkspaceImage

서비스 접두사	작업
	workspaces:DeployWorkspaceApplications
	workspaces:DeregisterWorkspaceDirectory
	workspaces:DescribeAccount
	workspaces:DescribeAccountModifications
	workspaces:DescribeApplicationAssociations
	workspaces:DescribeApplications
	workspaces:DescribeBundleAssociations
	workspaces:DescribeClientBranding
	workspaces:DescribeClientProperties
	workspaces:DescribeConnectClientAddIns
	workspaces:DescribeConnectionAliases
	workspaces:DescribeConnectionAliasPermissions
	workspaces:DescribeImageAssociations
	workspaces:DescribeIpGroups
	workspaces:DescribeWorkspaceAssociations
	workspaces:DescribeWorkspaceBundles
	workspaces:DescribeWorkspaceDirectories
	workspaces:DescribeWorkspaceImagePermissions
	workspaces:DescribeWorkspaces
	workspaces:DescribeWorkspacesConnectionStatus
	workspaces:DescribeWorkspaceSnapshots

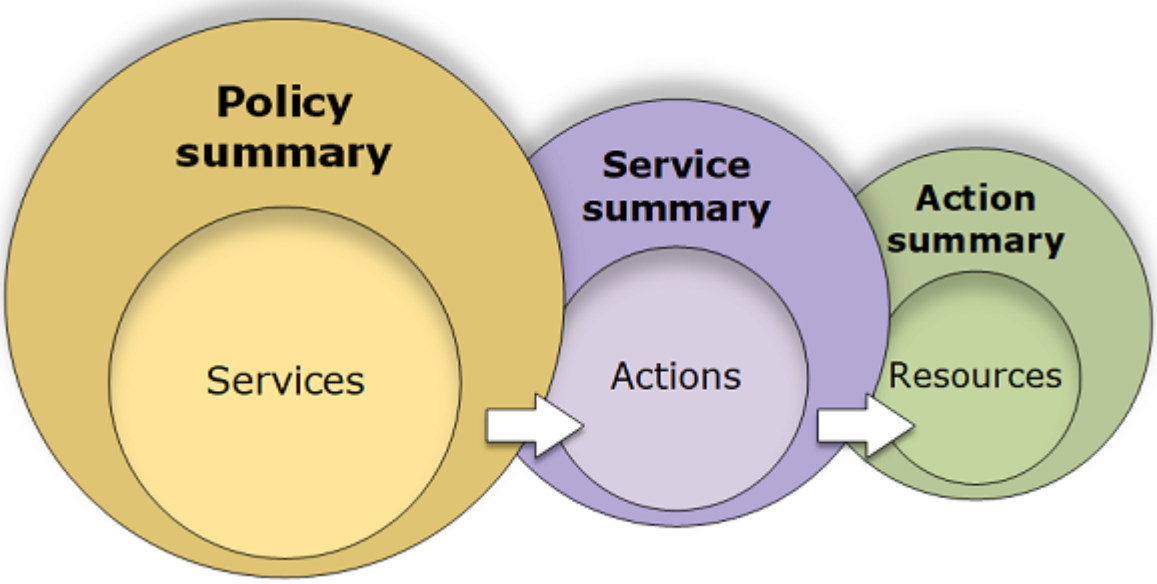
서비스 접두사	작업
	workspaces:DescribeWorkspacesPools
	workspaces:DescribeWorkspacesPoolSessions
	workspaces:DisassociateConnectionAlias
	workspaces:DisassociateIpGroups
	workspaces:DisassociateWorkspaceApplication
	workspaces:GetAccountLink
	workspaces:ImportClientBranding
	workspaces:ImportWorkspaceImage
	workspaces:ListAccountLinks
	workspaces:ListAvailableManagementCidrRanges
	workspaces:MigrateWorkspace
	workspaces:ModifyAccount
	workspaces:ModifyCertificateBasedAuthProperties
	workspaces:ModifyClientProperties
	workspaces:ModifySamlProperties
	workspaces:ModifySelfservicePermissions
	workspaces:ModifyStreamingProperties
	workspaces:ModifyWorkspaceAccessProperties
	workspaces:ModifyWorkspaceCreationProperties
	workspaces:ModifyWorkspaceProperties
	workspaces:ModifyWorkspaceState

서비스 접두사	작업
	workspaces:RebootWorkspaces
	workspaces:RebuildWorkspaces
	workspaces:RegisterWorkspaceDirectory
	workspaces:RejectAccountLinkInvitation
	workspaces:RestoreWorkspace
	workspaces:StartWorkspaces
	workspaces:StartWorkspacesPool
	workspaces:StopWorkspaces
	workspaces:StopWorkspacesPool
	workspaces:TerminateWorkspaces
	workspaces:TerminateWorkspacesPool
	workspaces:TerminateWorkspacesPoolSession
	workspaces:UpdateConnectClientAddIn
	workspaces:UpdateConnectionAliasPermission
	workspaces:UpdateWorkspaceBundle
	workspaces:UpdateWorkspaceImagePermission
	workspaces:UpdateWorkspacesPool

서비스 접두사	작업
xray	xray:CreateGroup xray:CreateSamplingRule xray:DeleteGroup xray:DeleteResourcePolicy xray:DeleteSamplingRule xray:GetEncryptionConfig xray:GetGroup xray:GetGroups xray:GetInsight xray:GetInsightEvents xray:GetInsightImpactGraph xray:GetInsightSummaries xray:GetSamplingRules xray:ListResourcePolicies xray:PutEncryptionConfig xray:PutResourcePolicy xray:UpdateGroup xray:UpdateSamplingRule

정책에 의해 부여된 권한 이해

IAM 콘솔에는 정책에서 각 서비스에 대해 허용되거나 거부되는 액세스 수준, 리소스, 조건을 설명하는 정책 요약 테이블이 포함되어 있습니다. 정책은 3가지 테이블, 즉 정책 요약, 서비스 요약, 작업 요약으로 요약됩니다. 정책 요약 테이블에는 서비스 목록이 포함되어 있습니다. 서비스 요약을 보려면 여기서 서비스를 선택합니다. 이 요약 테이블에는 작업 목록과 선택한 서비스에 대해 연결된 권한이 포함되어 있습니다. 해당 테이블에서 작업을 선택하여 작업 요약을 볼 수 있습니다. 이 테이블에는 리소스 목록과 선택한 작업에 대한 조건이 포함되어 있습니다.



사용자 페이지 또는 역할 페이지에서 해당 사용자에 연결된 모든 정책(관리형 및 인라인)에 대한 정책 요약을 볼 수 있습니다. 정책 페이지에서 모든 관리형 정책에 대한 요약을 봅니다. 관리형 정책에는 AWS 관리형 정책, AWS 관리형 직무 정책, 고객 관리형 정책이 포함되어 있습니다. 정책이 사용자 또는 다른 IAM 자격 증명에 연결되어 있는지 여부와 상관없이 정책 페이지에서 이러한 정책에 대한 요약을 볼 수 있습니다.

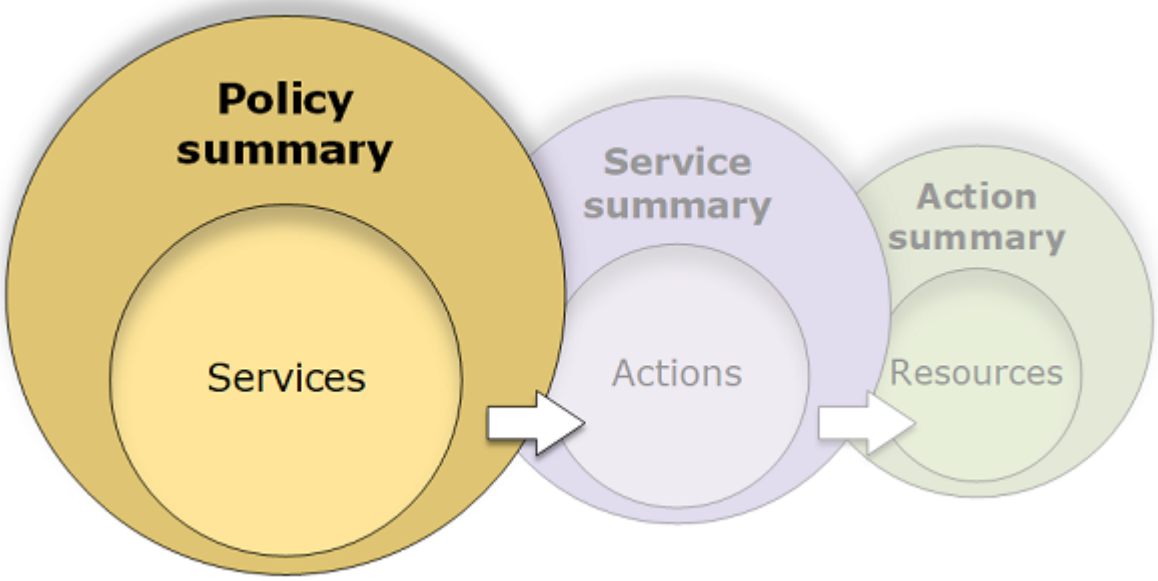
정책 요약의 정보를 사용하여 정책에서 허용되거나 거부된 권한을 확인할 수 있습니다. 정책 요약은 예상한 권한을 제공하지 않는 정책의 문제를 해결하고 정책을 수정하는 데 도움이 됩니다.

주제

- 정책 요약(서비스 목록)
- 서비스 요약(작업 목록)
- 작업 요약(리소스 목록)
- 정책 요약 예시

정책 요약(서비스 목록)

정책은 3가지 테이블, 즉 정책 요약, 서비스 요약, 작업 요약으로 요약됩니다. 정책 요약 테이블에는 서비스 목록과 선택한 정책에 의해 정의된 권한의 요약이 포함되어 있습니다.



정책 요약 테이블은 하나 이상의 Uncategorized services(미분류 서비스), 명시적 거부, 허용 섹션으로 그룹화됩니다. IAM에서 인식하지 못하는 서비스가 정책에 포함되어 있으면 해당 서비스는 테이블의 미분류 서비스(Uncategorized services) 섹션에 포함됩니다. IAM에서 서비스를 인식하면 해당 서비스는 정책(Deny 또는 Allow)의 효과에 따라 테이블의 명시적 거부 또는 허용 섹션에 포함됩니다.

정책 요약 보기

사용자 세부 정보 페이지의 권한 탭에서 정책 이름을 선택하여 사용자에게 연결된 정책에 대한 요약을 볼 수 있습니다. 역할 세부 정보 페이지의 권한 탭에서 정책 이름을 선택하여 역할에 연결된 정책에 대한 요약을 볼 수 있습니다. 정책 페이지에서 관리형 정책에 대한 정책 요약을 볼 수 있습니다. 정책에 정책 요약이 포함되지 않은 경우 정책 요약 누락을 참조하여 이유를 알아보세요.

정책 페이지에서 정책 요약을 확인하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 목록에서 보려는 정책의 이름을 선택합니다.
4. 정책 요약을 보려면 해당 정책의 정책 세부 정보 페이지에서 권한 탭을 확인합니다.

사용자에 연결된 정책의 요약을 확인하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 사용자 목록에서 정책을 보려는 사용자의 이름을 선택합니다.
4. 사용자에게 직접 연결되거나 그룹에서 연결된 정책의 목록을 보려면 해당 사용자의 요약 페이지에서 권한 탭을 봅니다.
5. 사용자에 대한 정책 테이블에서 보려는 정책의 행을 확장합니다.

역할에 연결된 정책의 요약 정보를 보려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 역할 목록에서 정책을 보려는 역할의 이름을 선택합니다.
4. 역할의 요약 페이지에서 권한 탭을 보고 역할에 연결된 정책 목록을 확인합니다.
5. 역할에 대한 정책 테이블에서 보려는 정책의 행을 확장합니다.

정책을 편집하여 경고 수정

정책 요약을 보는 동안 정책에서 예상한 권한을 제공하지 않는다는 알림이나 오타를 찾을 수 있습니다. 정책 요약을 직접 편집할 수 없습니다. 그러나 정책 요약이 보고하는 것과 동일한 여러 개의 오류 및 경고를 파악하는 시각적 정책 편집기를 사용하여 고객 관리형 정책을 편집할 수 있습니다. 그런 다음 정책 요약의 변경 사항을 확인하여 모든 문제가 수정되었는지 확인할 수 있습니다. 인라인 정책을 편집하는 방법에 대해 자세히 알아보려면 the section called “IAM 정책 편집” 섹션을 참조하세요. 단, AWS 관리형 정책은 편집할 수 없습니다.

시각적 편집기 옵션을 사용하여 정책 요약에 대한 정책을 편집하려면

1. 이전 절차에서 설명한 대로 정책의 요약을 엽니다.
2. 편집을 선택합니다.

   사용자 페이지에서 해당 사용자에게 연결된 고객 관리형 정책을 편집하려는 경우, 정책 페이지로 리디렉션됩니다. 고객 관리형 정책은 정책 페이지에서만 편집할 수 있습니다.

3. 시각적 편집기 옵션을 선택하면 정책의 편집 가능한 시각적 표현을 볼 수 있습니다. IAM은 시각적 편집기에서 모양을 최적화하고 문제를 쉽게 찾아 수정하기 위해 정책을 재구성할 수 있습니다. 페이지의 경고 및 오류 메시지는 정책 문제를 수정하도록 안내할 수 있습니다. IAM 정책을 재구성하는 방법에 대한 자세한 내용은 정책 재구성 섹션을 참조하세요.
4. 정책을 편집하고 다음을 선택하여 정책 요약에 반영된 변경 사항을 봅니다. 문제가 계속 표시되면 이전을 선택하여 편집 화면으로 돌아갑니다.
5. 변경 사항을 저장하려면 변경 사항 저장을 선택합니다.

JSON 옵션을 사용하여 정책 요약에 대한 정책을 편집하려면

1. 이전 절차에서 설명한 대로 정책의 요약을 엽니다.
2. 요약 및 JSON 버튼을 사용하여 정책 요약과 JSON 정책 문서를 비교합니다. 이 정보를 사용하여 정책 문서에서 변경할 행을 결정할 수 있습니다.
3. 편집을 선택한 다음, JSON 옵션을 선택하여 JSON 정책 문서를 편집합니다.

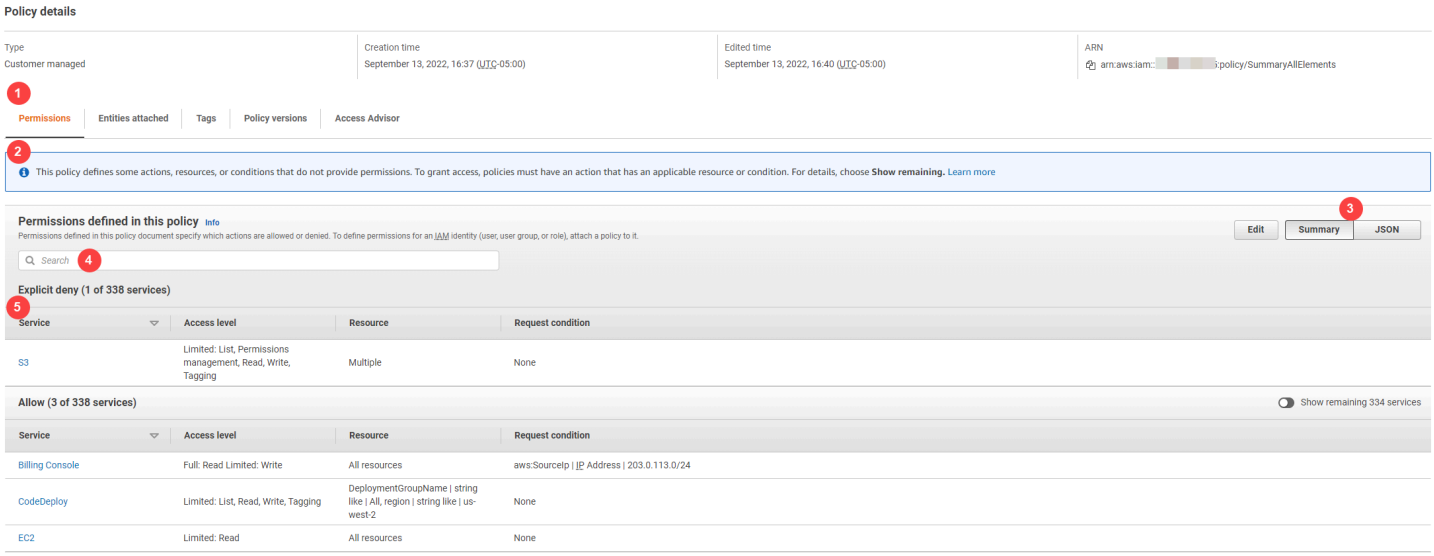
> **Note**
>
> 언제든지 시각적 편집기 옵션과 JSON 편집기 옵션을 서로 전환할 수 있습니다. 그러나 변경을 적용하거나 시각적 편집기 옵션에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화되도록 정책을 재구성할 수 있습니다. 자세한 내용은 정책 재구성 단원을 참조하십시오.

사용자 페이지에서 해당 사용자에게 연결된 고객 관리형 정책을 편집하려는 경우, 정책 페이지로 리디렉션됩니다. 고객 관리형 정책은 정책 페이지에서만 편집할 수 있습니다.

4. 정책을 편집합니다. 정책 검증 동안 생성된 모든 보안 경고, 오류 또는 일반 경고를 해결하고 다음을 선택합니다. 문제가 계속 표시되면 이전을 선택하여 편집 화면으로 돌아갑니다.
5. 변경 사항을 저장하려면 변경 사항 저장을 선택합니다.

정책 요약의 요소 이해

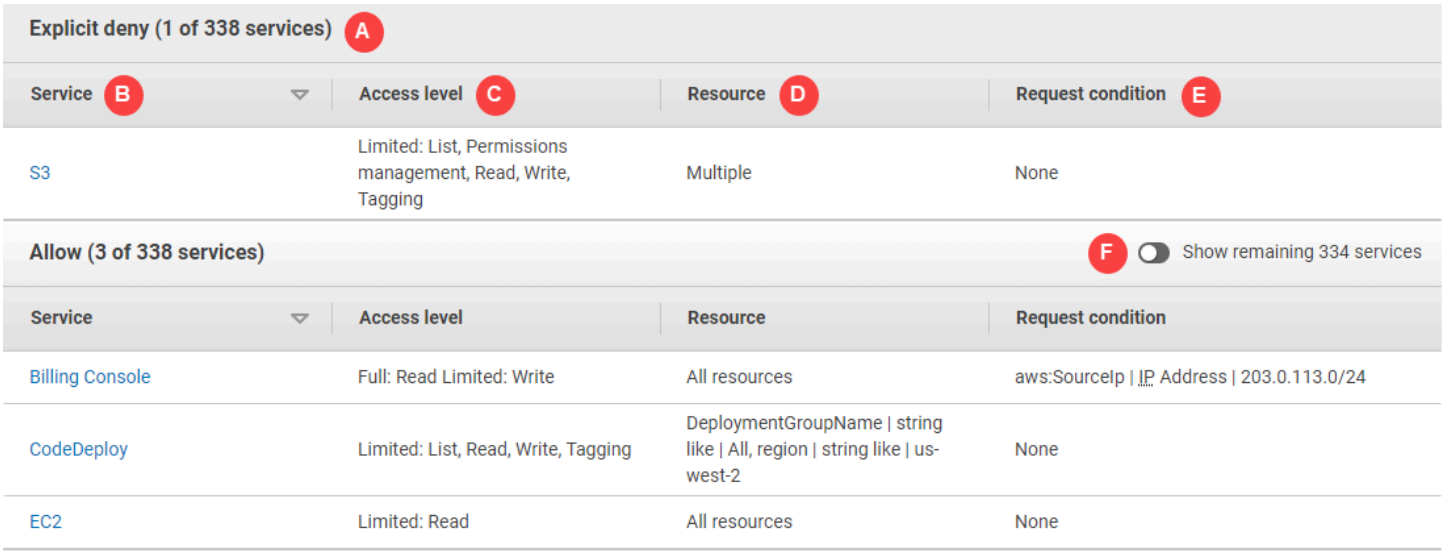
다음의 정책 세부 정보 페이지 예시에서 SummaryAllElements 정책은 사용자에 직접 연결된 관리형 정책(고객 관리형 정책)입니다. 이 정책이 확장되어 정책 요약을 표시합니다.



이전 이미지에서 정책 요약은 정책 페이지에 표시되어 있습니다.

1. 권한 탭에는 정책에 정의된 권한이 포함됩니다.
2. 정책에서 정책에 정의된 일부 작업, 리소스 및 조건에 권한을 부여하지 않는 경우 페이지 상단에 경고 또는 오류 배너가 나타납니다. 그런 다음 정책 요약에 문제에 대한 세부 정보가 포함됩니다. 정책 요약이 정책에서 부여하는 권한을 이해하고 문제를 해결하는 데 얼마나 도움이 되는지 알아보려면 the section called “정책이 필요한 권한을 부여하지 않음” 섹션을 참조하세요.
3. 요약 및 JSON 버튼을 사용하여 정책 요약과 JSON 정책 문서 사이를 전환합니다.
4. 검색 상자를 사용하여 서비스 목록을 제한하면 용이하게 특정 서비스를 찾을 수 있습니다.
5. 확장된 보기는 SummaryAllElements 정책의 세부 정보를 보여 줍니다.

다음 정책 요약 테이블 이미지는 정책 세부 정보 페이지에서 확장된 SummaryAllElements 정책을 표시합니다.



Explicit deny (1 of 338 services) A

Service B	Access level C	Resource D	Request condition E
S3	Limited: List, Permissions management, Read, Write, Tagging	Multiple	None

Allow (3 of 338 services) F Show remaining 334 services

Service	Access level	Resource	Request condition
Billing Console	Full: Read Limited: Write	All resources	aws:SourceIp | IP Address | 203.0.113.0/24
CodeDeploy	Limited: List, Read, Write, Tagging	DeploymentGroupName | string like | All, region | string like | us-west-2	None
EC2	Limited: Read	All resources	None

이전 이미지에서 정책 요약은 정책 페이지에 표시되어 있습니다.

A. IAM에서 인식하는 해당 서비스의 경우 정책이 서비스 사용을 허용하거나 명시적으로 거부하는지 여부에 따라 서비스가 정렬됩니다. 이 예시에서는 정책에 Amazon S3 서비스에 대한 `Deny` 설명문과 결제, CodeDeploy 및 Amazon EC2 서비스에 대한`Allow` 설명문이 포함됩니다.

B. 서비스 - 이 열에는 정책에서 정의된 서비스가 나열되고 각 서비스의 세부 정보를 제공합니다. 정책 요약 테이블에서 각 서비스 이름은 서비스 요약 테이블에 대한 링크(서비스 요약(작업 목록) 단원 참조)입니다. 이 예시에서는 Amazon S3, 결제, CodeDeploy 및 Amazon EC2 서비스에 대해 권한이 정의되어 있습니다.

C. 액세스 수준 - 이 열은 정책이 각 액세스 수준(`List`, `Read`, `Write`, `Permission Management` 및 `Tagging`)의 작업에 대해 `Full` 또는 `Limited` 중 어느 권한을 정의했는지 보여줍니다. 액세스 레벨 요약에 대한 자세한 정보 및 예시는 정책 요약의 액세스 레벨 이해하기 섹션을 참조하세요.

- 전체 액세스 - 이 항목은 해당 서비스가 서비스에 대해 사용 가능한 4개 액세스 수준 모두에서 모든 작업에 액세스할 수 있음을 나타냅니다.
- 항목에 Full access(전체 액세스)가 포함되지 않은 경우 해당 서비스는 서비스를 위한 모든 작업이 아니라 일부 작업에 액세스할 수 있습니다. 그러면 액세스 권한이 액세스 레벨(`List`, `Read`, `Write`, `Permission Management` 및 `Tagging`) 각각에 대한 다음의 설명으로 정의됩니다.

  Full(전체): 정책이 나열된 각 액세스 레벨 분류의 모든 작업에 대한 액세스 권한을 제공합니다. 이 예시에서는 정책이 모든 결제 `Read` 작업에 대한 액세스 권한을 제공합니다.

  Limited(제한): 정책이 나열된 각 액세스 레벨 분류에서 하나 이상의 작업(모든 작업은 아님)에 대한 액세스 권한을 제공합니다. 이 예시에서는 정책이 일부 결제 `Write` 작업에 대한 액세스 권한을 제공합니다.

D. 리소스 - 이 열은 정책이 각 서비스에 대해 지정한 리소스를 보여줍니다.

- 다중 - 정책이 서비스 내 둘 이상(모든 리소스는 아님)의 리소스를 포함합니다. 이 예시에서는 둘 이상의 Amazon S3 리소스에 대한 액세스가 명시적으로 거부됩니다.
- 모든 리소스 – 정책이 서비스의 모든 리소스에 대해 정의되어 있습니다. 이 예시에서는 정책이 모든 결제 리소스에 대해 나열된 작업을 수행할 수 있도록 허용합니다.
- Resource text - 정책이 서비스의 리소스 하나를 포함합니다. 이 예시에서는 나열된 작업이 `DeploymentGroupName` CodeDeploy 리소스에서만 허용됩니다. 서비스가 IAM에 제공하는 정보에 따라 ARN이 표시되거나 정의된 리소스 유형이 표시될 수 있습니다.

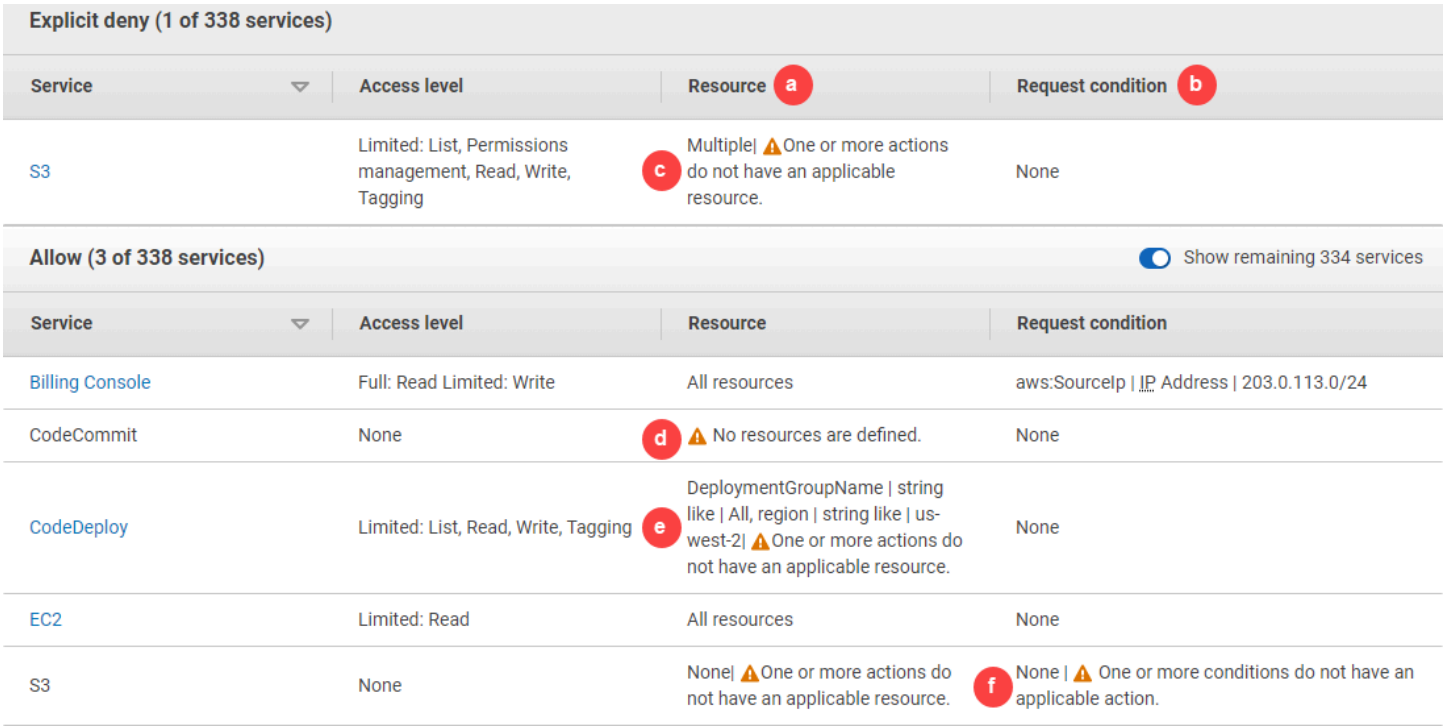
> **Note**
>
> 이 열은 다른 서비스의 리소스를 포함할 수 있습니다. 리소스를 포함하는 정책 설명에 동일한 서비스의 작업과 리소스를 모두 포함하지 않으면 정책에 일치하지 않는 리소스가 포함됩니다. IAM은 정책을 생성하거나 정책 요약에서 정책을 볼 때 일치하지 않는 리소스에 대해 경고하지 않습니다. 이 열에 일치하지 않는 리소스가 포함되어 있으면 정책에 오류가 있는지 검토해야 합니다. 정책을 더 잘 이해하려면 항상 정책 시뮬레이터로 테스트합니다.

E. 요청 조건(Request condition) - 이 열은 리소스와 연결된 서비스 또는 작업에 조건이 적용되는지 여부를 나타냅니다.

- 없음 - 정책이 서비스에 대한 조건을 포함하지 않습니다. 이 예시에서는 Amazon S3 서비스에서 거부된 작업에 적용된 조건이 없습니다.
- Condition text - 정책이 서비스에 대한 조건 하나를 포함합니다. 이 예시에서는 소스의 IP 주소가 `203.0.113.0/24`와 일치하는 경우에만 나열된 결제 작업이 허용됩니다.
- 다중 - 정책이 서비스에 대해 둘 이상의 조건을 포함합니다. 정책에 대한 여러 조건을 각각 보려면 JSON을 선택하여 정책 문서를 봅니다.

F. 나머지 서비스 보기 - 이 버튼을 전환하여 정책에 의해 정의되지 않은 서비스를 포함하도록 테이블을 확장합니다. 이러한 서비스는 이 정책 내에서 명시적으로 거부(또는 기본적으로 거부)됩니다. 그러나 다른 정책 문으로 서비스를 사용하여 허용하거나 명시적으로 거부할 수 있습니다. 정책 요약에는 단일 정책의 권한이 요약되어 있습니다. AWS 서비스가 지정된 요청을 허용하거나 거부할지 여부를 결정하는 방법에 대해 알아보려면 정책 평가 로직을 참조하세요.

정책 또는 정책 내 요소가 권한을 부여하지 않는 경우 IAM은 정책 요약에 추가 경고 및 정보를 제공합니다. 다음 정책 요약 테이블은 SummaryAllElements 정책 세부 정보 페이지에 확장된 나머지 서비스 보기 서비스와 가능한 경고를 보여줍니다.



Explicit deny (1 of 338 services)

Service	Access level	Resource a	Request condition b
S3	Limited: List, Permissions management, Read, Write, Tagging	c Multiple| ⚠ One or more actions do not have an applicable resource.	None

Allow (3 of 338 services) Show remaining 334 services

Service	Access level	Resource	Request condition
Billing Console	Full: Read Limited: Write	All resources	aws:SourceIp | IP Address | 203.0.113.0/24
CodeCommit	None	d ⚠ No resources are defined.	None
CodeDeploy	Limited: List, Read, Write, Tagging	e DeploymentGroupName | string like | All, region | string like | us-west-2| ⚠ One or more actions do not have an applicable resource.	None
EC2	Limited: Read	All resources	None
S3	None	None| ⚠ One or more actions do not have an applicable resource.	f None | ⚠ One or more conditions do not have an applicable action.

앞의 그림에는 권한이 없이 정의된 작업, 리소스 또는 조건을 포함하는 모든 서비스가 나와 있습니다.

a. 리소스 경고(Resource warnings) - 포함된 모든 작업이나 리소스에 대해 권한을 제공하지 않는 서비스의 경우 테이블의 리소스 열에 다음 경고 중 하나가 나타납니다.

   - ⚠

     정의된 리소스가 없습니다. - 서비스에 작업이 정의되었지만 정책에 지원되는 리소스가 포함되지 않았음을 의미합니다.

   - ⚠

     하나 이상의 작업이 적용할 리소스가 없습니다. - 서비스에 정의된 작업이 있지만 일부 작업에 지원되는 리소스가 없음을 의미합니다.

   - ⚠

     하나 이상의 리소스가 적용할 작업이 없습니다. - 서비스에 정의된 리소스가 있지만 일부 리소스에 지원 작업이 없음을 의미합니다.

   서비스에 적용 가능한 리소스가 없는 작업과 적용 가능한 작업이 있는 리소스가 있는 경우, 하나 이상의 리소스에 적용할 작업이 없습니다. 경고가 표시됩니다. 그 이유는 서비스에 대한 서비스 요약을 볼 때 어떤 작업에도 적용되지 않는 리소스가 표시되지 않기 때문입니다. `ListAllMyBuckets` 작업의 경우 리소스 수준 권한을 지원하지 않고 `s3:x-amz-acl` 조건 키를 지원하지 않기 때문에

이 정책에 마지막 경고가 포함됩니다. 리소스 문제나 조건 문제를 수정한 경우 나머지 문제가 세부 경고에 나타납니다.

b. 요청 조건 경고(Request condition warnings) - 포함된 모든 조건에 대해 권한을 제공하지 않는 서비스의 경우 테이블의 요청 조건(Request condition)열에 다음 경고 중 하나가 나타납니다.

- 

  하나 이상의 작업이 적용할 조건이 없습니다. - 서비스에 정의된 작업이 있지만 일부 작업에 지원되는 조건이 없음을 의미합니다.

- 

  하나 이상의 조건이 적용할 작업이 없습니다. - 서비스에 정의된 조건이 있지만 일부 조건에 지원 작업이 없음을 의미합니다.

c. Multiple |

하나 이상의 작업이 적용할 리소스가 없습니다. - Amazon S3의 Deny 문에 리소스가 두 개 이상 포함되어 있습니다. 또한 작업이 두 개 이상 포함되어 있으며, 일부 작업은 리소스를 지원하고, 일부 작업은 리소스를 지원하지 않습니다. 정책을 보려면 the section called “SummaryAllElements JSON 정책 문서” 섹션을 참조하세요. 이 경우 정책에는 모든 Amazon S3 작업과, 정의된 버킷 또는 버킷 객체에서 수행될 수 있는 작업만 포함됩니다.

d. 

정의된 리소스가 없음 - 서비스에 정의된 작업이 있지만, 지원되는 리소스가 정책에 없으므로 서비스가 권한을 제공하지 않습니다. 이 경우 정책에 CodeCommit 작업이 포함되지만 CodeCommit 리소스는 포함되지 않습니다.

e. DeploymentGroupName | 문자열 | 전체, 리전 | 문자열 | us-west-2

| 

하나 이상의 작업에 해당 리소스가 없습니다. - 서비스에 정의된 작업이 한 개 있고, 지원 리소스가 없는 작업이 한 개 이상 있습니다.

f. 없음 |

하나 이상의 조건이 적용할 작업이 없습니다. - 서비스에 지원 작업이 없는 조건 키가 한 개 이상입니다.

SummaryAllElements JSON 정책 문서

SummaryAllElements 정책은 해당 계정의 권한을 정의하는 데 사용하기 위한 것이 아닙니다. 이것은 정책 요약을 보는 중 만날 수 있는 오류와 경고를 보여주기 위한 것입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "billing:Get*",
                "payments:List*",
                "payments:Update*",
                "account:Get*",
                "account:List*",
                "cur:GetUsage*"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "203.0.113.0/24"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "arn:aws:s3:::customer",
                "arn:aws:s3:::customer/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:GetConsoleScreenshots"
            ],
            "Resource": [
```

```
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "codedploy:*",
                "codecommit:*"
            ],
            "Resource": [
                "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup:*",
                "arn:aws:codebuild:us-east-1:123456789012:project/my-demo-project"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:GetObject",
                "s3:DeletObject",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::developer_bucket",
                "arn:aws:s3:::developer_bucket/*",
                "arn:aws:autoscling:us-east-2:123456789012:autoscalgrp"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": [
                        "public-read"
                    ],
                    "s3:prefix": [
                        "custom",
                        "other"
                    ]
                }
            }
        }
    ]
}
```

정책 요약의 액세스 레벨 이해하기

AWS 액세스 수준 요약

정책 요약에는 해당 정책에서 언급된 각 서비스에 정의된 작업 권한을 설명하는 액세스 레벨 요약이 포함됩니다. 정책 요약에 대한 자세한 내용은 정책에 의해 부여된 권한 이해 섹션을 참조하세요. 액세스 레벨 요약은 각 액세스 레벨(`List`, `Read`, `Tagging`, `Write` 및 `Permissions management`)의 작업에 정책에서 정의된 `Full` 또는 `Limited` 권한이 있는지 여부를 나타냅니다. 서비스의 각 작업에 할당된 액세스 레벨 분류를 보려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

다음 예제에서는 한 정책이 지정된 서비스에 대해 제공하는 액세스 권한을 설명합니다. 전체 JSON 정책 문서 및 관련 요약의 예는 정책 요약 예시 섹션을 참조하세요.

Service	액세스 레벨	이 정책은 다음을 제공합니다.
IAM	모든 액세스	IAM 서비스 내의 모든 작업에 대한 액세스 권한
CloudWatch	전체: 목록	`List` 액세스 수준의 모든 CloudWatch 작업에 대한 액세스 권한. 하지만 `Read`, `Write` 또는 `Permissions management` 액세스 수준 분류의 작업에 대한 액세스 권한은 제공하지 않음
Data Pipeline	제한: 목록, 읽기	`List` 및 `Read` 액세스 레벨의 AWS Data Pipeline 작업 하나 이상(모든 작업은 아님)에 대한 액세스 권한. 단, `Write` 또는 `Permissions management` 작업에 대한 액세스 권한은 제외됨
EC2	전체: 목록, 읽기 제한: 쓰기	모든 Amazon EC2 `List` 및 `Read` 작업에 대한 액세스 권한, 하나 이상의 Amazon EC2 `Write` 작업(모든 작업은 아님)에 대한 액세스 권한. 단, `Permissions management` 액세스 수준 분류의 작업에 대한 액세스 권한은 제외됨
S3	제한: 읽기, 쓰기, 권한 관리	하나 이상의 Amazon S3 `Read`, `Write` 및 `Permissions management` 작업(모든 작업은 아님)에 대한 액세스 권한

Service	액세스 레벨	이 정책은 다음을 제공합니다.
CodeDeploy	(비어 있음)	알 수 없는 액세스(IAM에서 이 서비스를 인식하지 않음)
API Gateway	None	정책에 정의된 액세스 없음
CodeBuild	⚠ 의된 작업 없음.	서비스에 대해 작업이 정의되지 않아서 액세스할 수 없습니다. 이 문제를 이해하고 문제를 해결하는 방법을 보려면 the section called “정책이 필요한 권한을 부여하지 않음” 섹션을 참조하세요.

앞서 언급한 바와 같이, 모든 액세스는 정책이 서비스 내 모든 작업에 대한 액세스를 제공함을 나타냅니다. 서비스의 모든 작업이 아니라 일부에 대한 액세스 권한을 제공하는 정책은 액세스 레벨 분류에 따라 추가로 그룹화됩니다. 이는 다음 액세스 레벨 그룹 중 하나에 의해 표시됩니다.

- 전체: 정책이 지정된 액세스 레벨 분류의 모든 작업에 대한 액세스 권한을 제공합니다.
- 제한: 정책이 지정된 액세스 레벨 분류 내 하나 이상의 작업(모든 작업은 아님)에 대한 액세스 권한을 제공합니다.
- 없음: 정책에서 액세스를 제공하지 않습니다.
- (비어 있음): IAM 에서 이 서비스를 인식하지 않습니다. 서비스 이름에 오타가 포함되어 있으면 정책은 서비스에 대한 액세스를 제공하지 않습니다. 서비스 이름이 정확하면 서비스는 정책 요약을 지원할 수 없거나 프리뷰에 있을 수 있습니다. 이 경우 정책은 액세스를 제공할 수 있지만 해당 액세스를 정책 요약에 표시할 수 없습니다. 일반적으로 사용할 수 있는(GA) 서비스에 대한 정책 요약 지원을 요청하려면 서비스가 IAM 정책 요약을 지원하지 않음을 참조하세요.

작업에 대한 제한적(부분적) 액세스 권한을 포함하는 액세스 레벨 요약은 AWS 액세스 레벨 분류 `List`, `Read`, `Tagging`, `Write` 또는 `Permissions management`을 사용하여 그룹화됩니다.

AWS 액세스 수준

AWS는 서비스의 작업에 대해 다음과 같은 액세스 레벨 분류를 정의합니다.

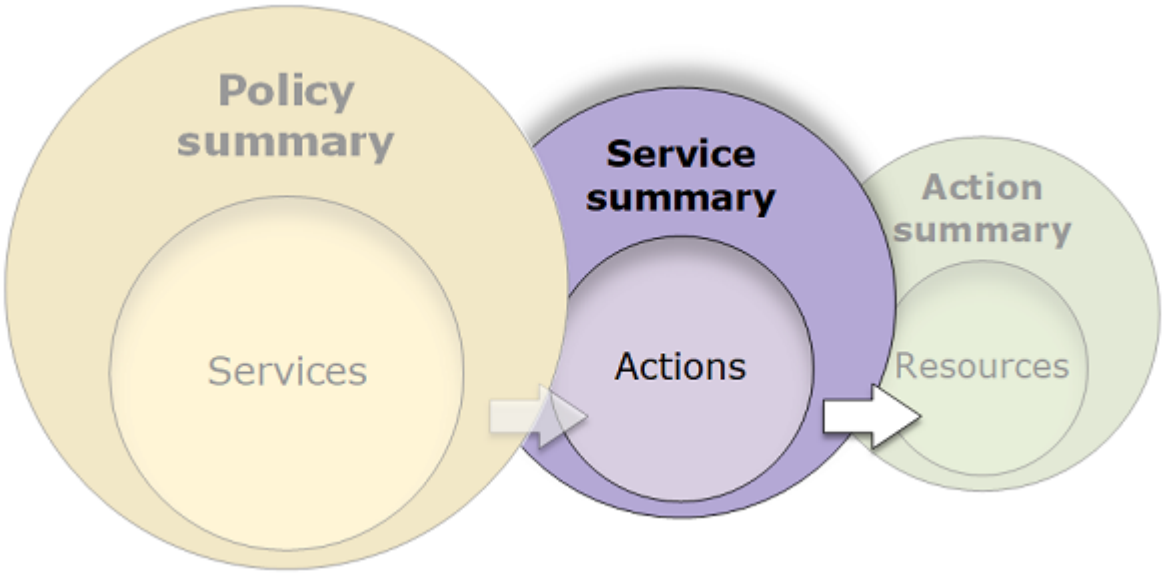
- 목록: 서비스의 리소스를 나열하여 객체가 존재하는지 판단할 수 있는 권한입니다. 이 액세스 레벨의 작업은 객체를 나열할 수 있으나 리소스의 내용을 확인할 수 없습니다. 예를 들어 Amazon S3 작업 `ListBucket`의 액세스 수준은 목록입니다.
- 읽기: 서비스에서 리소스 내용과 속성을 읽을 수 있으나 편집할 수 없는 권한입니다. 예를 들어 Amazon S3 작업 `GetObject` 및 `GetBucketLocation`의 액세스 수준은 Read입니다.
- 태그 지정: 리소스 태그의 상태만 변경하는 작업을 수행할 수 있는 권한입니다. 예를 들어 IAM 작업 `TagRole` 및 `UntagRole`은 역할에 대한 태그 지정 또는 태그 취소만 허용하므로 태그 지정 액세스 수준입니다. 그러나 `CreateRole` 작업은 사용자가 역할을 생성할 때 역할 리소스에 태그를 지정하도록 허용합니다. 이 작업은 태그를 추가하는 것만이 아니므로 `Write` 액세스 레벨입니다.
- 쓰기: 서비스에서 리소스를 생성, 삭제하거나 수정할 수 있는 권한입니다. 예를 들어 Amazon S3 작업 `CreateBucket`, `DeleteBucket` 및 `PutObject`는 쓰기 액세스 수준입니다. `Write` 작업은 리소스 태그 수정을 허용할 수도 있습니다. 그러나 태그 변경만 허용하는 작업은 `Tagging` 액세스 레벨입니다.
- 권한 관리: 서비스에서 리소스 권한을 부여하거나 수정할 수 있는 권한입니다. 예를 들어 대부분의 IAM 및 AWS Organizations 작업과 Amazon S3 작업 `PutBucketPolicy` 및 `DeleteBucketPolicy` 등의 액세스 수준은 권한 관리입니다.

> **도움말**
>
> AWS 계정의 보안을 개선하려면 권한 관리 액세스 레벨 분류를 포함하는 정책을 제한하거나 정기적으로 모니터링합니다.

서비스의 각 작업에 대한 액세스 수준 분류를 보려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

서비스 요약(작업 목록)

정책은 3가지 테이블, 즉 정책 요약, 서비스 요약, 작업 요약으로 요약됩니다. 서비스 요약 테이블에는 작업 목록과 선택한 서비스의 정책에 의해 정의된 권한의 요약이 포함되어 있습니다.



권한을 부여하는 정책 요약에 나열되어 있는 각 서비스에 대해 서비스 요약을 볼 수 있습니다. 이 테이블은 Uncategorized actions(미분류 작업), Uncategorized resource types(미분류 리소스 유형) 및 액세스 수준 섹션으로 분류되어 있습니다. IAM에서 인식하지 못하는 작업이 정책에 포함되어 있으면 해당 작업은 테이블의 미분류 작업(Uncategorized actions) 섹션에 포함됩니다. IAM에서 작업을 인식하면 해당 작업은 테이블의 액세스 수준(List(목록), Read(읽기), Write(쓰기), Permissions management(권한 관리)) 섹션 중 하나에 포함됩니다. 서비스의 각 작업에 할당된 액세스 레벨 분류를 보려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

서비스 요약 보기

정책 페이지에서는 관리형 정책에 대한 서비스 요약을 볼 수 있습니다.

관리형 정책의 서비스 요약을 확인하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 목록에서 보려는 정책의 이름을 선택합니다.
4. 정책 요약을 보려면 해당 정책의 정책 세부 정보 페이지에서 권한 탭을 확인합니다.
5. 서비스의 정책 요약 목록에서 확인하려는 서비스의 이름을 선택합니다.

사용자에게 연결된 정책의 서비스 요약을 확인하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. 사용자 목록에서 정책을 보려는 사용자의 이름을 선택합니다.
4. 사용자에게 직접 연결되거나 그룹에서 연결된 정책의 목록을 보려면 해당 사용자의 요약 페이지에서 권한 탭을 봅니다.
5. 사용자에 대한 정책 테이블에서 확인하려는 정책의 이름을 선택합니다.

   사용자 페이지에서 해당 사용자에게 연결된 정책의 서비스 요약을 보려고 하는 경우, 정책 페이지로 리디렉션됩니다. 서비스 요약은 정책 페이지에서만 볼 수 있습니다.
6. 요약을 선택합니다. 서비스의 정책 요약 목록에서 확인하려는 서비스의 이름을 선택합니다.

> **Note**
>
> 선택한 정책이 사용자에게 직접 연결된 인라인 정책인 경우 서비스 요약 테이블이 표시됩니다. 정책이 그룹에서 연결한 인라인 정책인 경우 해당 그룹의 JSON 정책 문서로 자동으로 이동합니다. 정책이 관리형 정책인 경우 정책 페이지에서 해당 정책의 서비스 요약이 게시된 부분으로 자동으로 이동합니다.

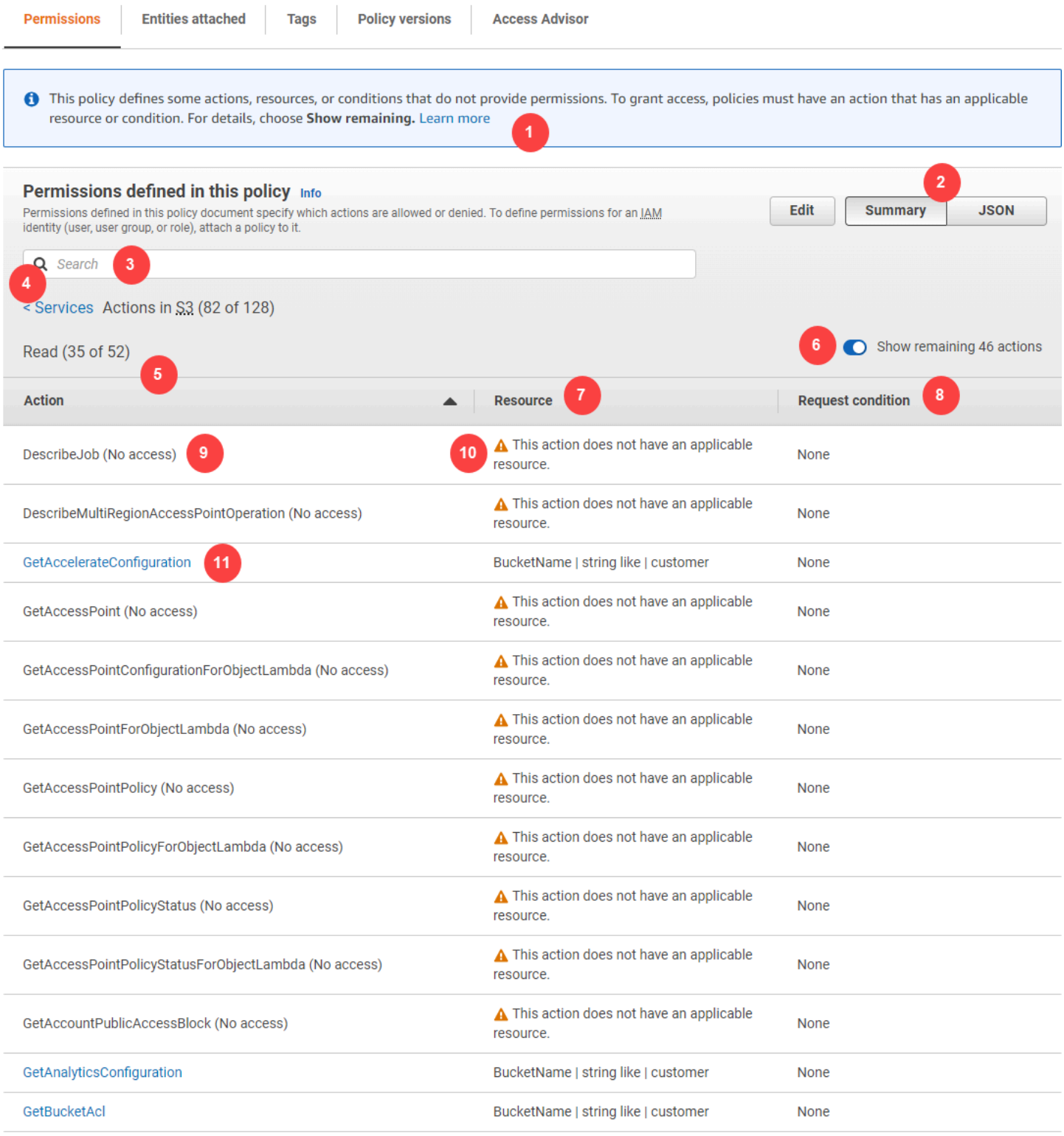
역할에 연결된 정책의 서비스 요약 정보를 보려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 역할 목록에서 정책을 보려는 역할의 이름을 선택합니다.
4. 역할의 요약 페이지에서 권한 탭을 보고 역할에 연결된 정책 목록을 확인합니다.
5. 역할에 대한 정책 테이블에서 보려는 정책의 이름을 선택합니다.

   역할 페이지에서 해당 사용자에게 연결된 정책의 서비스 요약을 보려고 하는 경우, 정책 페이지로 리디렉션됩니다. 서비스 요약은 정책 페이지에서만 볼 수 있습니다.
6. 서비스의 정책 요약 목록에서 확인하려는 서비스의 이름을 선택합니다.

서비스 요약의 요소 이해하기

아래 예시는 Amazon S3 작업에 대한 서비스 요약이며, 정책 요약에서 허용됩니다. 이 서비스에 대한 작업은 액세스 수준별로 그룹화됩니다. 예를 들어 서비스에서 이용 가능한 총 52개 읽기 작업 중에서 35개 읽기 작업이 정의됩니다.



관리형 정책에 대한 서비스 요약 페이지에 포함되는 정보는 다음과 같습니다.

1. 정책에서 정책의 서비스에 대해 정의된 일부 작업, 리소스 및 조건에 권한을 부여하지 않는 경우 페이지 상단에 경고 배너가 나타납니다. 그런 다음 서비스 요약에 문제에 대한 세부 정보가 포함됩니

다. 정책 요약이 정책에서 부여하는 권한을 이해하고 문제를 해결하는 데 얼마나 도움이 되는지 알아보려면 the section called “정책이 필요한 권한을 부여하지 않음” 섹션을 참조하세요.

2. JSON을 선택하면 정책에 대한 추가 세부 정보를 볼 수 있습니다. 이를 통해 작업에 적용된 모든 조건을 볼 수 있습니다. (사용자에게 직접 연결된 인라인 정책의 서비스 요약을 보려면 서비스 요약 대화 상자를 닫고 정책 요약으로 돌아가 JSON 정책 문서에 액세스해야 합니다.)
3. 특정 작업의 요약을 보려면 검색 상자에 키워드를 입력하여, 사용할 수 있는 작업의 목록을 줄이세요.
4. 서비스 뒤로 화살표 옆에 서비스 이름(이 경우 S3)이 표시됩니다. 이 서비스의 서비스 요약에는 정책에서 정의한 허용되거나 거부되는 작업의 목록이 수록되어 있습니다. 서비스가 권한 탭의 (명시적 거부) 아래에 표시될 경우, 서비스 요약 테이블에 나열된 작업이 명시적으로 거부됩니다. 서비스가 권한 탭의 허용 아래에 표시될 경우, 서비스 요약 테이블에 나열된 작업이 허용됩니다.
5. 작업 - 이 열에는 정책 내에 정의된 작업이 나열되고 각 작업에 해당하는 리소스와 조건이 제시됩니다. 정책에서 작업에 권한을 부여하거나 거부하는 경우 작업 이름이 작업 요약 테이블에 링크됩니다. 테이블은 정책이 허용하거나 거부하는 액세스 레벨에 따라 최소 1개 이상에서 최대 5개의 섹션으로 이러한 작업을 그룹화합니다. 섹션은 목록, 읽기, 쓰기, 권한 관리 및 태그 지정입니다. 개수는 각 액세스 레벨에서 권한을 제공하는 인식할 수 있는 작업의 수를 나타냅니다. 총계는 서비스에 대해 알려진 작업의 수입니다. 이 예시에서는 총 52개의 알려진 S3 읽기 작업에서 35개의 작업이 권한을 제공합니다. 서비스의 각 작업에 할당된 액세스 레벨 분류를 보려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요.
6. 나머지 작업 보기 - 알려졌지만 이 서비스에 대한 권한을 제공하지 않는 작업을 포함하는 테이블을 펼치거나 숨기려면 이 버튼을 선택합니다. 버튼을 전환하면 권한을 제공하지 않는 모든 요소에 대해 경고가 표시됩니다.
7. 리소스 - 이 열은 정책이 서비스에 대해 정의한 리소스를 보여줍니다. IAM은 리소스가 각 작업에 적용되는지 여부를 확인하지 않습니다. 이 예시에서는 Amazon S3 서비스의 작업이 `developer_bucket` Amazon S3 버킷 리소스에서만 허용됩니다. 서비스가 IAM에 제공하는 정보에 따라 `arn:aws:s3:::developer_bucket/*` 등의 ARN이 표시되거나 `BucketName = developer_bucket` 등의 정의된 리소스 유형이 표시될 수 있습니다.

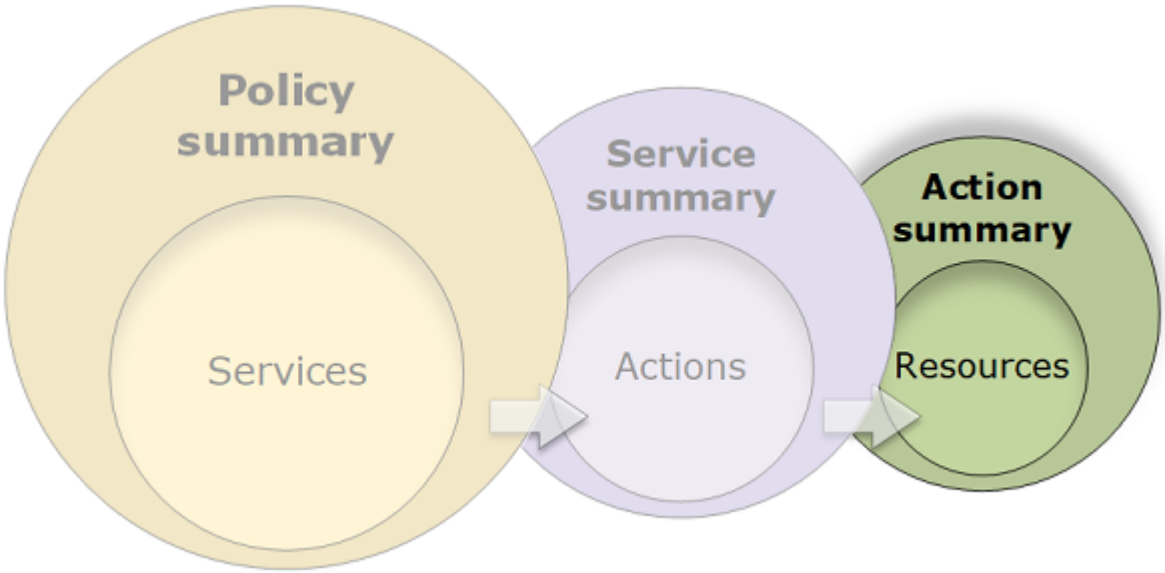
> **Note**
>
> 이 열은 다른 서비스의 리소스를 포함할 수 있습니다. 리소스를 포함하는 정책 설명에 동일한 서비스의 작업과 리소스를 모두 포함하지 않으면 정책에 일치하지 않는 리소스가 포함됩니다. IAM은 정책을 생성하거나 서비스 요약에서 정책을 볼 때 일치하지 않는 리소스에 대해 경고하지 않습니다. IAM은 작업이 리소스에 적용되는지도 나타내지 않고 서비스 일치 여

부만 나타냅니다. 이 열에 일치하지 않는 리소스가 포함되어 있으면 정책에 오류가 있는지 검토해야 합니다. 정책을 더 잘 이해하려면 항상 정책 시뮬레이터로 테스트합니다.

8. 요청 조건 - 이 열은 리소스와 연결된 작업에 조건이 적용되는지 여부를 나타냅니다. 이러한 조건에 대해 자세히 알아보려면 JSON을 선택하여 JSON 정책 문서를 검토합니다.
9. (액세스 권한 없음) - 이 정책에 권한을 제공하지 않는 작업이 한 개 있습니다.
10. 리소스 경고 - 전체 권한을 제공하지 않는 리소스를 포함하는 작업의 경우 다음 경고 중 하나가 나타납니다.
    - 이 작업은 리소스 수준 권한을 지원하지 않습니다.(This action does not support resource-level permissions.) 리소스에 대한 와일드카드(*)가 필요합니다.(This requires a wildcard (*) for the resource.) - 정책에 리소스 수준 권한이 있지만, 이 작업에 대한 권한을 제공하려면 `"Resource": ["*"]`를 포함해야 함을 의미합니다.
    - 이 작업은 적용할 리소스가 없습니다. - 지원되는 리소스 없이 작업이 정책에 포함됨을 의미합니다.
    - 이 작업은 적용할 리소스 및 조건이 없습니다. - 지원되는 리소스 및 조건 없이 작업이 정책에 포함됨을 의미합니다. 이 경우 이 서비스의 정책에 포함된 조건도 있지만 이 작업에 적용되는 조건은 없습니다.
11. 권한을 제공하지 않는 작업에는 작업 요약에 대한 링크가 포함됩니다.

작업 요약(리소스 목록)

정책은 3가지 테이블, 즉 정책 요약, 서비스 요약, 작업 요약으로 요약됩니다. 작업 요약 테이블에는 리소스 목록과 선택한 작업에 적용되는 연결 조건이 포함되어 있습니다.



권한을 부여하는 각 작업에 대한 작업 요약을 보려면 서비스 요약의 링크를 선택합니다. 작업 요약 테이블에는 리소스의 리전 및 계정을 비롯하여 리소스에 대한 세부 정보가 포함되어 있습니다. 또한 각 리소스에 적용하는 조건을 볼 수 있습니다. 이를 통해 일부 리소스에 적용되고 다른 리소스에는 적용되지 않는 조건을 볼 수 있습니다.

작업 요약 보기

정책 페이지에서 관리형 정책, 사용자에게 연결된 모든 정책, 역할에 연결된 모든 정책에 대한 작업 요약을 볼 수 있습니다.

관리형 정책의 작업 요약을 확인하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 목록에서 보려는 정책의 이름을 선택합니다.
4. 정책 요약을 보려면 해당 정책의 정책 세부 정보 페이지에서 권한 탭을 확인합니다.
5. 서비스의 정책 요약 목록에서 확인하려는 서비스의 이름을 선택합니다.
6. 작업의 서비스 요약 목록에서 확인하려는 작업의 이름을 선택합니다.

사용자에게 연결된 정책의 작업 요약을 확인하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

2. 탐색 창에서 사용자를 선택합니다.
3. 사용자 목록에서 정책을 보려는 사용자의 이름을 선택합니다.
4. 사용자에게 직접 연결되거나 그룹에서 연결된 정책의 목록을 보려면 해당 사용자의 요약 페이지에서 권한 탭을 봅니다.
5. 사용자에 대한 정책 테이블에서 확인하려는 정책의 이름을 선택합니다.

   사용자 페이지에서 해당 사용자에게 연결된 정책의 서비스 요약을 보려고 하는 경우, 정책 페이지로 리디렉션됩니다. 서비스 요약은 정책 페이지에서만 볼 수 있습니다.
6. 서비스의 정책 요약 목록에서 확인하려는 서비스의 이름을 선택합니다.

   > **Note**
   >
   > 선택한 정책이 사용자에게 직접 연결된 인라인 정책인 경우 서비스 요약 테이블이 표시됩니다. 정책이 그룹에서 연결한 인라인 정책인 경우 해당 그룹의 JSON 정책 문서로 자동으로 이동합니다. 정책이 관리형 정책인 경우 정책 페이지에서 해당 정책의 서비스 요약이 게시된 부분으로 자동으로 이동합니다.
7. 작업의 서비스 요약 목록에서 확인하려는 작업의 이름을 선택합니다.

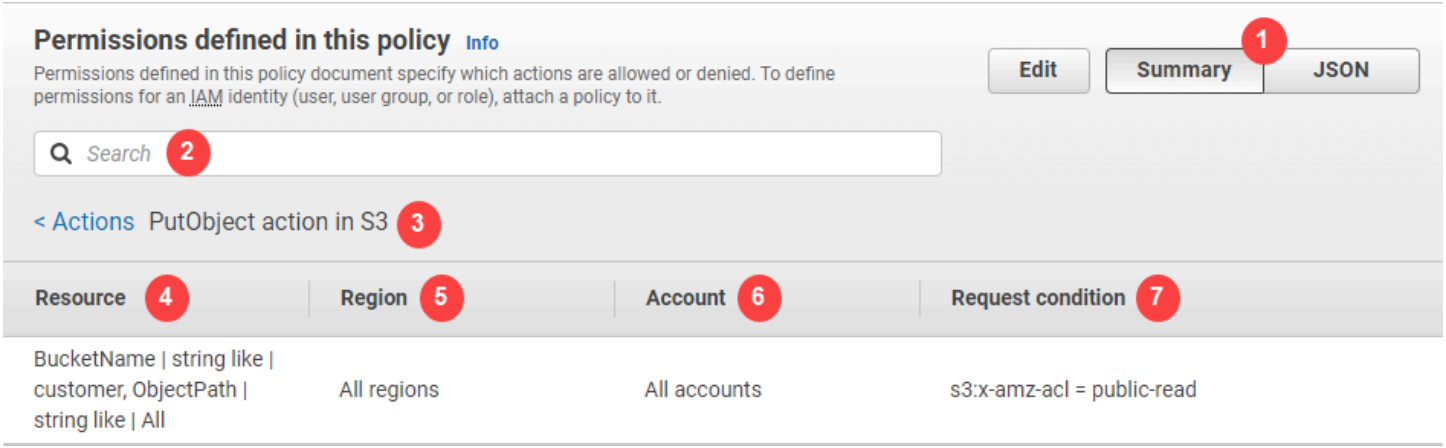
역할 연결된 정책의 작업 요약을 보려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 역할 목록에서 정책을 보려는 역할의 이름을 선택합니다.
4. 역할의 요약 페이지에서 권한 탭을 보고 역할에 연결된 정책 목록을 확인합니다.
5. 역할에 대한 정책 테이블에서 보려는 정책의 이름을 선택합니다.

   역할 페이지에서 해당 사용자에게 연결된 정책의 서비스 요약을 보려고 하는 경우, 정책 페이지로 리디렉션됩니다. 서비스 요약은 정책 페이지에서만 볼 수 있습니다.
6. 서비스의 정책 요약 목록에서 확인하려는 서비스의 이름을 선택합니다.
7. 작업의 서비스 요약 목록에서 확인하려는 작업의 이름을 선택합니다.

작업 요약의 요소 이해하기

아래 예시는 Amazon S3 서비스 요약의 `PutObject`(쓰기) 작업에 대한 작업 요약입니다(서비스 요약(작업 목록) 참조). 이 작업의 경우 정책이 단일 리소스에 대한 여러 조건을 정의합니다.



작업 요약 페이지에 포함되는 정보는 다음과 같습니다.

1. JSON을 선택하면 작업에 적용되는 여러 가지 조건 등 정책에 관한 추가 세부 정보를 볼 수 있습니다. (사용자에게 직접 연결된 인라인 정책의 서비스 요약을 보려면 단계가 다릅니다. 그 경우 JSON 정책 문서에 액세스하려면 서비스 요약 대화 상자를 닫고 정책 요약으로 돌아가야 합니다.)
2. 특정 리소스의 요약을 보려면 검색 상자에 키워드를 입력하여 사용할 수 있는 리소스의 목록을 줄입니다.
3. 작업 뒤로 화살표 옆에 서비스와 작업의 이름이 `action name action in service` 형식으로 표시됩니다(이 경우 S3의 PutObject 작업). 이 서비스의 작업 요약에는 정책에서 정의된 리소스의 목록이 포함되어 있습니다.
4. 리소스 - 이 열에는 정책이 선택한 서비스에 대해 정의한 리소스가 나열됩니다. 이 예시에서는 PutObject 작업이 모든 객체 경로와 `developer_bucket` Amazon S3 버킷 리소스에서만 허용됩니다. 서비스가 IAM에 제공하는 정보에 따라 `arn:aws:s3:::developer_bucket/*` 등의 ARN이 표시되거나 `BucketName = developer_bucket, ObjectPath = All` 등의 정의된 리소스 유형이 표시될 수 있습니다.
5. 리전 - 이 열은 리소스가 정의된 리전을 보여줍니다. 리소스는 모든 리전 또는 단일 리전에 대해 정의할 수 있습니다. 리소스는 둘 이상의 리전에 존재할 수 없습니다.
   - 모든 리전 - 리소스와 연결된 작업은 모든 리전에 적용됩니다. 이 예시에서는 작업이 전역적 서비스 Amazon S3에 속합니다. 전역적 서비스에 속하는 작업은 모든 리전에 적용됩니다.
   - 리전 텍스트(Region text) - 리소스와 연결된 작업은 한 리전에 적용됩니다. 예를 들어 정책은 리소스에 대한 `us-east-2` 리전을 지정할 수 있습니다.

6. 계정 - 이 열은 리소스와 연결된 서비스 또는 작업이 특정 계정에 적용되는지를 나타냅니다. 리소스는 모든 계정 또는 단일 계정에 존재할 수 있습니다. 리소스는 둘 이상의 특정 계정에 존재할 수 없습니다.
   - 모든 계정(All accounts) - 리소스와 연결된 작업은 모든 계정에 적용됩니다. 이 예시에서는 작업이 전역적 서비스 Amazon S3에 속합니다. 전역적 서비스에 속하는 작업은 모든 계정에 적용됩니다.
   - 현재 계정 - 리소스와 연결된 작업은 현재 계정에만 적용됩니다.
   - 계정 번호(Account number) - 리소스와 연결된 작업은 하나의 계정(현재 로그인되지 않은 계정)에 적용됩니다. 예를 들어 정책이 리소스에 대한 `123456789012` 계정을 지정하면 계정 번호가 정책 요약에 나타납니다.
7. 요청 조건(Request condition) - 이 열은 리소스와 연결된 작업에 조건이 적용되는지를 보여줍니다. 이 예시에는 `s3:x-amz-acl = public-read` 조건이 포함됩니다. 이러한 조건에 대해 자세히 알아보려면 JSON을 선택하여 JSON 정책 문서를 검토합니다.

정책 요약 예시

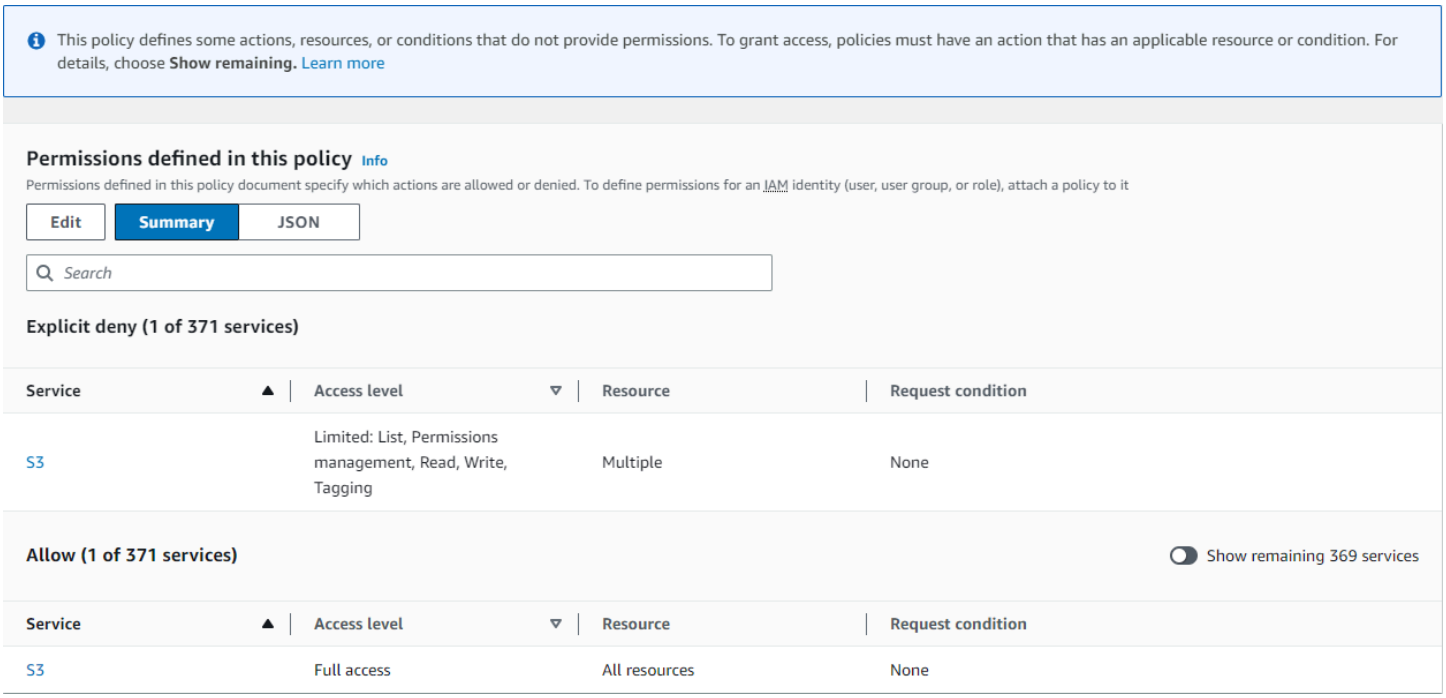
다음 예시에는 정책을 통해 부여되는 권한을 이해하는 데 도움이 되는 JSON 정책과 그에 연결된 정책 요약, 서비스 요약, 작업 요약이 포함되어 있습니다.

정책 1: DenyCustomerBucket

이 정책은 동일한 서비스에 대한 허용과 거부를 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccess",
            "Effect": "Allow",
            "Action": ["s3:*"],
            "Resource": ["*"]
        },
        {
            "Sid": "DenyCustomerBucket",
            "Action": ["s3:*"],
            "Effect": "Deny",
            "Resource": ["arn:aws:s3:::customer", "arn:aws:s3:::customer/*" ]
        }
    ]
}
```

DenyCustomerBucket 정책 요약:



DenyCustomerBucket S3 (Explicit deny) 서비스 요약:

< Services Actions in S3 (82 of 130) Show remaining 48 actions

Read (35 of 53)

Action	Resource	Request condition
GetAccelerateConfiguration	BucketName| string like |customer	None
GetAnalyticsConfiguration	BucketName| string like |customer	None
GetBucketAcl	BucketName| string like |customer	None
GetBucketCORS	BucketName| string like |customer	None
GetBucketLocation	BucketName| string like |customer	None
GetBucketLogging	BucketName| string like |customer	None
GetBucketNotification	BucketName| string like |customer	None
GetBucketObjectLockConfiguration	BucketName| string like |customer	None
GetBucketOwnershipControls	BucketName| string like |customer	None
GetBucketPolicy	BucketName| string like |customer	None
GetBucketPolicyStatus	BucketName| string like |customer	None
GetBucketPublicAccessBlock	BucketName| string like |customer	None
GetBucketRequestPayment	BucketName| string like |customer	None
GetBucketTagging	BucketName| string like |customer	None
GetBucketVersioning	BucketName| string like |customer	None
GetBucketWebsite	BucketName| string like |customer	None

GetObject(읽기) 작업 요약:

< Actions GetObject action in S3

Resource	Region	Account	Request condition
BucketName| string like |customer, ObjectPath| string like |All	-	All accounts	None

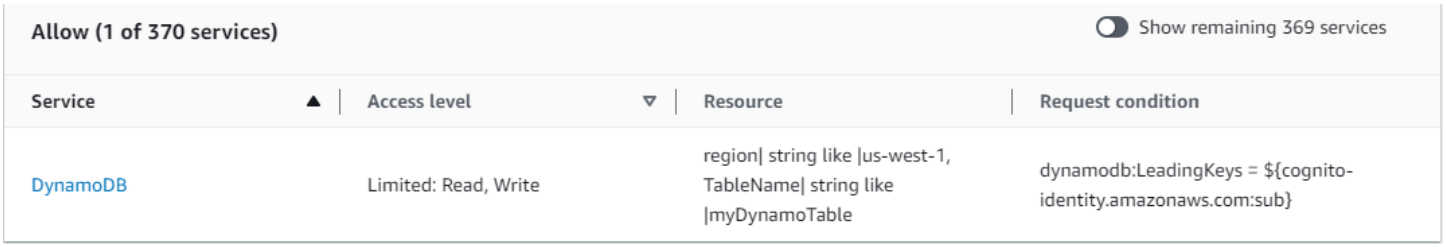
정책 2: DynamoDbRowCognitoID

이 정책은 사용자의 Amazon Cognito ID를 기반으로 Amazon DynamoDB에 대한 행 수준 액세스 권한을 제공합니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-1:123456789012:table/myDynamoTable"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:LeadingKeys": [
                        "${cognito-identity.amazonaws.com:sub}"
                    ]
                }
            }
        }
    ]
}
```

DynamoDbRowCognitoID 정책 요약:



DynamoDbRowCognitoID DynamoDB(허용) 서비스 요약:

< Services Actions in DynamoDB (4 of 65) Show remaining 61 actions

Read (1 of 26)

Action	Resource	Request condition
GetItem	region| string like |us-west-1, TableName| string like |myDynamoTable	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}

Write (3 of 33)

Action	Resource	Request condition
DeleteItem	region| string like |us-west-1, TableName| string like |myDynamoTable	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}
PutItem	region| string like |us-west-1, TableName| string like |myDynamoTable	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}
UpdateItem	region| string like |us-west-1, TableName| string like |myDynamoTable	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}

GetItem(나열) 작업 요약:

< Actions GetItem action in DynamoDB

Resource	Region	Account	Request condition
region| string like |us-west-1, TableName| string like |myDynamoTable	us-west-1	123456789012	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}

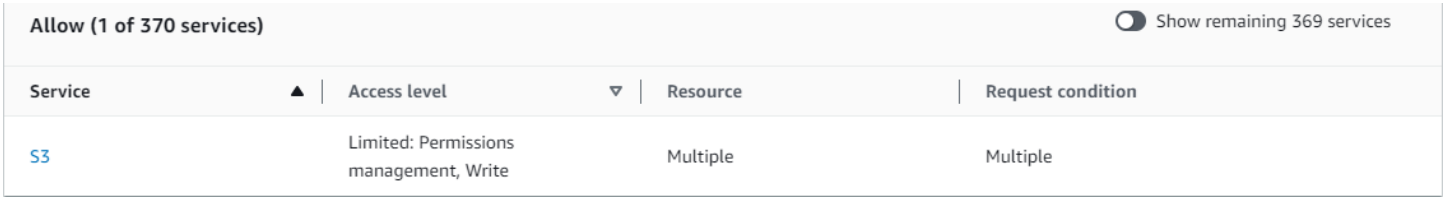
정책 3: MultipleResourceCondition

이 정책에는 다수의 리소스와 조건이 포함됩니다.

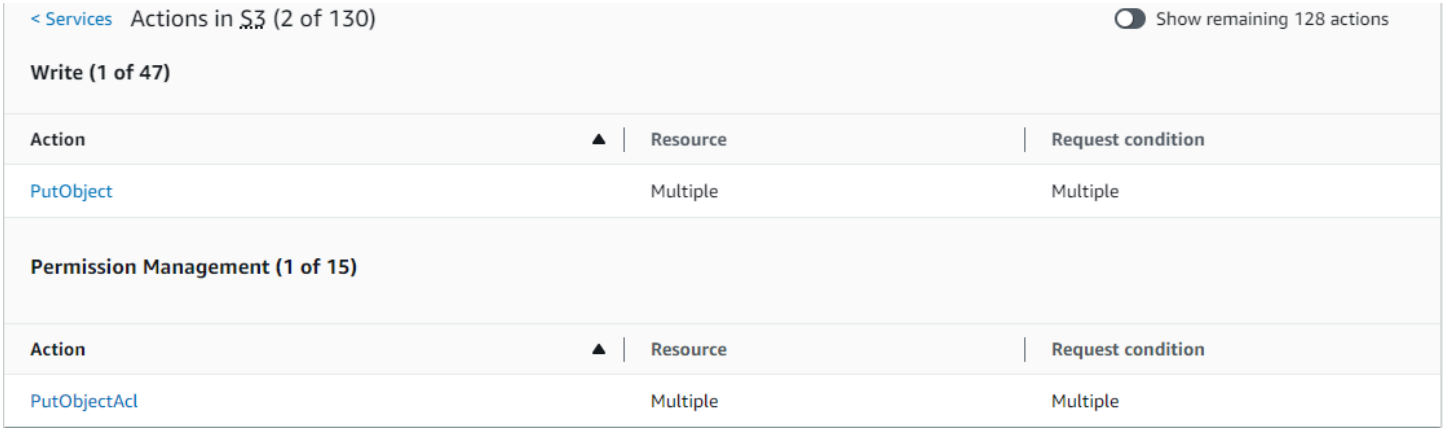
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": ["arn:aws:s3:::Apple_bucket/*"],
            "Condition": {"StringEquals": {"s3:x-amz-acl": ["public-read"]}}
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": ["arn:aws:s3:::Orange_bucket/*"],
            "Condition": {"StringEquals": {
                "s3:x-amz-acl": ["custom"],
                "s3:x-amz-grant-full-control": ["1234"]
            }}
        }
    ]
}
```

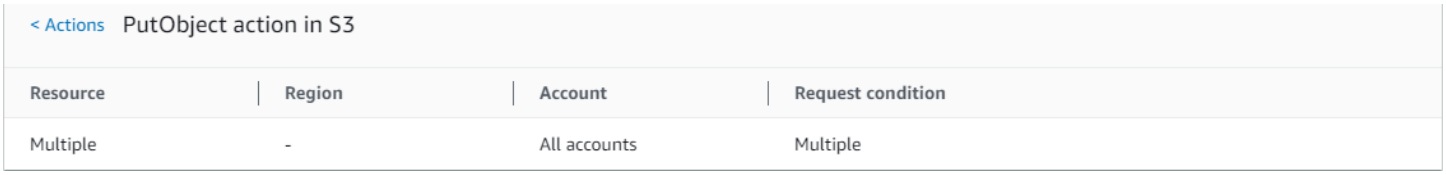
MultipleResourceCondition 정책 요약:



MultipleResourceCondition S3(허용) 서비스 요약:



PutObject(쓰기) 작업 요약:



정책 4: EC2_troubleshoot

다음 정책을 통해 사용자는 실행 중인 Amazon EC2 인스턴스의 스크린샷을 만들어 EC2 문제 해결에 필요한 도움을 얻을 수 있습니다. 또한 이 정책은 Amazon S3 개발자 버킷의 항목에 대한 정보를 확인할 수 있도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:GetConsoleScreenshot"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::developer"
            ]
        }
    ]
}
```

EC2_Troubleshoot 정책 요약:

Allow (2 of 370 services) — Show remaining 368 services

Service	Access level	Resource	Request condition
EC2	Limited: Read	All resources	None
S3	Limited: List	BucketName| string like |developer	None

EC2_Troubleshoot S3(허용) 서비스 요약:

< Services Actions in S3 (1 of 130) Show remaining 129 actions

List (1 of 7)

Action	Resource	Request condition
ListBucket	BucketName| string like |developer	None

ListBucket(나열) 작업 요약:

< Actions ListBucket action in S3

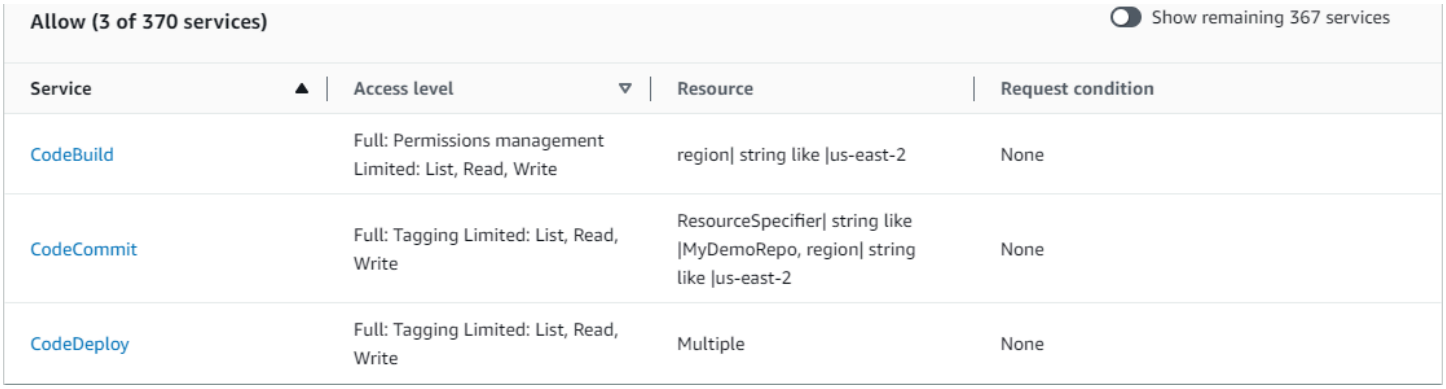
Resource	Region	Account	Request condition
BucketName| string like |developer	-	All accounts	None

정책 5: CodeBuild_CodeCommit_CodeDeploy

이 정책은 특정 CodeBuild, CodeCommit, CodeDeploy 리소스에 대한 액세스 권한을 제공합니다. 이러한 리소스는 각 서비스에 고유하므로 일치하는 서비스에서만 나타납니다. `Action` 요소에 서비스와 일치하지 않는 리소스를 포함하는 경우 리소스가 모든 작업 요약에 나타납니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1487980617000",
            "Effect": "Allow",
            "Action": [
                "codebuild:*",
                "codecommit:*",
                "codedeploy:*"
            ],
            "Resource": [
                "arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project",
                "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo",
                "arn:aws:codedeploy:us-east-2:123456789012:application:WordPress_App",
                "arn:aws:codedeploy:us-east-2:123456789012:instance/AssetTag*"
            ]
        }
    ]
}
```

CodeBuild_CodeCommit_CodeDeploy 정책 요약:



Allow (3 of 370 services) Show remaining 367 services

Service	Access level	Resource	Request condition
CodeBuild	Full: Permissions management Limited: List, Read, Write	region| string like |us-east-2	None
CodeCommit	Full: Tagging Limited: List, Read, Write	ResourceSpecifier| string like |MyDemoRepo, region| string like |us-east-2	None
CodeDeploy	Full: Tagging Limited: List, Read, Write	Multiple	None

CodeBuild_CodeCommit_CodeDeploy CodeBuild(허용) 서비스 요약:

< Services Actions in CodeBuild (24 of 53) Show remaining 29 actions

Read (4 of 9)

Action	Resource	Request condition
BatchGetBuildBatches	region| string like |us-east-2	None
BatchGetBuilds	region| string like |us-east-2	None
BatchGetProjects	region| string like |us-east-2	None
GetResourcePolicy	region| string like |us-east-2	None

Write (16 of 28)

Action	Resource	Request condition
BatchDeleteBuilds	region| string like |us-east-2	None
CreateProject	region| string like |us-east-2	None
CreateWebhook	region| string like |us-east-2	None
DeleteBuildBatch	region| string like |us-east-2	None
DeleteProject	region| string like |us-east-2	None
DeleteWebhook	region| string like |us-east-2	None
InvalidateProjectCache	region| string like |us-east-2	None
RetryBuild	region| string like |us-east-2	None
RetryBuildBatch	region| string like |us-east-2	None
StartBuild	region| string like |us-east-2	None
StartBuildBatch	region| string like |us-east-2	None
StopBuild	region| string like |us-east-2	None
StopBuildBatch	region| string like |us-east-2	None
UpdateProject	region| string like |us-east-2	None
UpdateProjectVisibility	region| string like |us-east-2	None
UpdateWebhook	region| string like |us-east-2	None

List (2 of 14)

CodeBuild_CodeCommit_CodeDeploy StartBuild (Write) 작업 요약:

< Actions StartBuild action in CodeBuild

Resource	Region	Account	Request condition
region| string like |us-east-2	us-east-2	123456789012	None

IAM 리소스에 액세스하는 데 필요한 권한

리소스는 서비스 내의 객체입니다. IAM 리소스에는 그룹, 사용자, 역할 및 정책이 있습니다. AWS 계정 루트 사용자 보안 인증 정보로 로그인한 경우 IAM 보안 인증 또는 IAM 리소스를 관리하는 데 아무런 제한이 없습니다. 하지만 IAM 사용자가 자격 증명이나 IAM 리소스를 관리하려면 그러한 권한이 명시적으로 부여되어야 합니다. 자격 증명 기반 정책을 사용자에 연결하여 권한을 부여할 수 있습니다.

> **Note**
>
> AWS 설명서 전체에서 특정 범주를 언급하지 않고 IAM 정책을 칭할 때는 자격 증명 기반 고객 관리형 정책을 의미합니다. 정책 범주에 대한 자세한 내용은 the section called “정책 및 권한” 섹션을 참조하세요.

IAM 자격 증명을 관리하기 위한 권한

IAM 그룹, 사용자, 역할 및 자격 증명을 관리하는 데 필요한 권한은 일반적으로 작업에 대한 API 작업에 해당합니다. 예를 들어 IAM 사용자를 생성하려면 해당하는 API 명령 `CreateUser`가 있는 `iam:CreateUser` 권한이 있어야 합니다. IAM 사용자가 다른 IAM 사용자를 생성할 수 있도록 다음과 같은 IAM 정책을 해당 사용자에게 연결할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:CreateUser",
    "Resource": "*"
  }
}
```

정책에서 `Resource` 요소의 값은 작업 및 그 작업이 적용될 수 있는 리소스에 따라 다릅니다. 앞의 예에서 정책은 사용자가 어떤 사용자도 생성할 수 있도록 허용합니다(*는 모든 문자열을 나타내는 와일드카드). 반면에, 사용자가 자신의 액세스 키(API 작업 `CreateAccessKey` 및 `UpdateAccessKey`)만 변경할 수 있도록 하는 정책에는 일반적으로 `Resource` 요소가 포함됩니다. 이 경우 ARN에는 다음 예제와 같이 현재 사용자의 이름을 해석하는 변수(`${aws:username}`)가 포함됩니다.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "ListUsersForConsole",
            "Effect": "Allow",
            "Action": "iam:ListUsers",
            "Resource": "arn:aws:iam::*:*"
        },
        {
            "Sid": "ViewAndUpdateAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:UpdateAccessKey",
                "iam:CreateAccessKey",
                "iam:ListAccessKeys"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

앞의 예에서 `${aws:username}`은 현재 사용자의 사용자 이름으로 변환되는 변수입니다. 정책 변수에 대한 자세한 내용은 IAM 정책 요소: 변수 및 태그 섹션을 참조하세요.

작업 이름에 와일드카드 문자(*)를 사용하면 특정 작업에 관련된 모든 작업에 대한 권한을 쉽게 부여할 수 있습니다. 예를 들어 사용자가 IAM 작업을 수행할 수 있게 하려면 그 작업에 대해 `iam:*`를 사용하면 됩니다. 사용자가 액세스 키에 관련된 작업만 수행할 수 있게 하려면 정책 문의 `iam:*AccessKey*` 요소에 `Action`를 사용하면 됩니다. 이렇게 하면 사용자에게 `CreateAccessKey`, `DeleteAccessKey`, `GetAccessKeyLastUsed`, `ListAccessKeys` 및 `UpdateAccessKey`작업을 수행할 수 있는 권한이 부여됩니다. (나중에 이름에 "AccessKey"가 포함되는 작업이 IAM에 추가될 경우에도 `iam:*AccessKey*` 요소에 대해 `Action`를 사용하며 사용자에게 새 작업에 대한 권한이 부여됩니다.) 다음 예는 사용자가 자신의 액세스 키에 속하는 모든 작업을 수행할 수 있게 허용하는 정책을 보여 줍니다(*`account-id`*를 해당 AWS 계정 ID로 변경).

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::account-id:user/${aws:username}"
  }
}
```

그룹 삭제와 같은 일부 작업에는 여러 작업이 포함됩니다. 즉, 먼저 그룹에서 사용자를 제거한 후 그룹의 정책을 분리 또는 삭제하고 나서 실제로 그룹을 삭제합니다. 사용자가 그룹을 삭제할 수 있게 하려는 경우 이러한 모든 관련 작업을 수행할 수 있는 권한을 부여해야 합니다.

AWS Management Console에서의 작업 권한

앞의 예는 사용자가 AWS CLI 또는 AWS SDK를 사용하여 작업을 수행할 수 있게 허용하는 정책을 보여 줍니다.

사용자가 콘솔에서 작업할 경우 콘솔은 그룹, 사용자, 역할 및 정책을 나열하고 그룹, 사용자 또는 역할과 연결된 정책을 가져오는 요청을 IAM에 보냅니다. 또한 콘솔은 AWS 계정 정보와 보안 주체에 대한 정보를 가져오는 요청도 보냅니다. 보안 주체는 콘솔에서 요청하는 사용자입니다.

일반적으로 작업을 수행하려면 일치하는 작업만 정책에 포함해야 합니다. 사용자를 생성하려면 `CreateUser` 작업을 호출하는 권한이 필요합니다. 콘솔을 사용하여 작업을 수행할 때 경우에 따라 콘솔에서 리소스를 표시하고 나열하며 가져오거나 볼 권한이 있어야 합니다. 이는 콘솔을 탐색하여 지정된 작업을 수행하기 위해 필요합니다. 예를 들어 Jorge라는 사용자가 콘솔을 사용하여 자신의 액세스 키를 변경하려면 IAM 콘솔에서 사용자를 선택할 것입니다. 이 작업은 콘솔에서 `ListUsers` 요청을 생성하게 합니다. Jorge에게 `iam:ListUsers` 작업에 대한 권한이 없을 경우 사용자를 나열하려고 시도할 때 콘솔이 액세스를 거부합니다. 그 결과, Jorge는 `CreateAccessKey` 및 `UpdateAccessKey` 작업에 대한 권한이 있는 경우에도 자신의 이름과 액세스 키를 가져올 수 없습니다.

사용자에게 AWS Management Console에서 그룹, 사용자, 역할, 정책, 자격 증명을 관리할 수 있는 권한을 부여하려면 콘솔에서 수행하는 작업에 대한 권한도 포함시켜야 합니다. 사용자에게 이러한 권한들을 부여하는 데 사용할 수 있는 몇 가지 정책의 예를 보려면 IAM 리소스를 관리하기 위한 정책의 예 섹션을 참조하세요.

전 AWS 계정에 권한 부여

계정의 IAM 사용자에게 리소스에 대한 액세스 권한을 직접 부여할 수 있습니다. 다른 계정의 사용자에게 리소스에 대한 액세스 권한이 필요한 경우, 권한을 포함하지만 특정 사용자와 연결되지 않는 엔터티인 IAM 역할을 생성할 수 있습니다. 다른 계정의 사용자는 해당 역할을 사용하여 해당 역할에 할당된 권한에 따라 리소스에 액세스할 수 있습니다. 자세한 내용은 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 섹션을 참조하세요.

> **Note**
>
> 일부 서비스만이 자격 증명 기반 정책 및 리소스 기반 정책(예: Amazon S3, Amazon SNS, Amazon SQS)에 설명된 리소스 기반 정책을 지원합니다. 그런 서비스의 역할 사용 대안은 공

유할 리소스(버킷, 주제 또는 대기열)에 정책을 연결하는 것입니다. 리소스 기반 정책은 리소스에 대한 액세스 허가를 받은 AWS 계정을 지정할 수 있습니다.

한 서비스에서 다른 서비스에 액세스할 권한

많은 AWS 제품은 다른 AWS 제품에 액세스합니다. 예를 들어, 여러 AWS 서비스(Amazon EMR, Elastic Load Balancing, Amazon EC2 Auto Scaling 포함)는 Amazon EC2 인스턴스를 관리합니다. 기타 AWS 서비스는 Amazon S3 버킷, Amazon SNS 주제, Amazon SQS 대기열 등을 사용합니다.

이러한 경우 권한 관리 시나리오가 서비스별로 다릅니다. 다음은 다양한 서비스에 대한 권한을 처리하는 방법의 예입니다.

- Amazon EC2 Auto Scaling에서 사용자에게 Auto Scaling을 사용할 권한이 있어야 하지만, Amazon EC2 인스턴스를 관리할 권한을 명시적으로 부여할 필요는 없습니다.
- AWS Data Pipeline에서 IAM 역할은 파이프라인에서 수행할 수 있는 작업을 결정합니다. 또한 사용자에게는 해당 역할을 수임할 권한이 필요합니다. (자세한 내용은 AWS Data Pipeline 개발자 가이드의 IAM으로 파이프라인에 대한 권한 부여를 참조하세요.)

AWS 제품에서 원하는 작업을 수행할 수 있도록 권한을 적절히 구성하는 방법에 대한 자세한 내용은 요청할 서비스 설명서를 참조하세요. 서비스에 대한 역할을 생성하는 방법에 대해 알아보려면 AWS 서비스에 대한 권한을 위임할 역할 생성 섹션을 참조하세요.

사용자를 대신하여 작동하도록 IAM 역할로 서비스 구성

AWS 서비스를 사용자를 대신하여 작동하도록 구성하려면 일반적으로 서비스에서 수행할 수 있는 작업을 정의하는 IAM 역할의 ARN을 입력합니다. AWS는 사용자에게 서비스에 역할을 전달할 권한이 있는지 확인합니다. 자세한 내용은 사용자에게 AWS 서비스에 역할을 전달할 권한 부여 섹션을 참조하세요.

필수 작업

작업은 리소스 보기, 생성, 편집 및 삭제와 같이 리소스에 대해 수행할 수 있는 사항입니다. 작업은 각 AWS 서비스별로 정의됩니다.

누군가 작업을 수행할 수 있도록 허용하려면 호출 자격 증명 또는 영향을 받은 리소스에 적용되는 정책에 필요한 작업을 포함시켜야 합니다. 일반적으로 작업을 수행하는 데 필요한 권한을 제공하려면 정책에 해당 작업을 포함시켜야 합니다. 예를 들어 사용자를 생성하려면 정책에 CreateUser 작업을 추가해야 합니다.

경우에 따라 정책에 관련된 작업을 추가로 포함해야 할 수도 있습니다. 예를 들어, `ds:CreateDirectory` 작업을 사용하여 AWS Directory Service에서 누군가에게 디렉터리를 생성할 권한을 제공하려면 정책에 다음 작업을 포함시켜야 합니다.

- `ds:CreateDirectory`
- `ec2:DescribeSubnets`
- `ec2:DescribeVpcs`
- `ec2:CreateSecurityGroup`
- `ec2:CreateNetworkInterface`
- `ec2:DescribeNetworkInterfaces`
- `ec2:AuthorizeSecurityGroupIngress`
- `ec2:AuthorizeSecurityGroupEgress`

시각적 편집기를 사용하여 정책을 생성하거나 편집할 때 경고 및 정책에 필요한 모든 작업을 선택하라는 메시지가 표시됩니다.

AWS Directory Service에서 디렉터리를 생성하는 데 필요한 권한에 대한 자세한 내용은 예제 2: 사용자에게 디렉터리 생성 허용을 참조하세요.

IAM 리소스를 관리하기 위한 정책의 예

다음은 사용자가 IAM 사용자, 그룹 및 자격 증명을 관리하기 위한 작업을 수행할 수 있게 하는 IAM 정책의 예시입니다. 여기에는 사용자가 자신의 암호, 액세스 키 및 멀티 팩터 인증(MFA) 디바이스를 관리할 수 있게 하는 정책이 포함됩니다.

사용자가 다른 AWS 제품 Amazon S3, Amazon EC2, DynamoDB 등으로 작업을 수행할 수 있도록 허용하는 예제 정책은 IAM 자격 증명 기반 정책의 예 섹션을 참조하세요.

주제

- 사용자가 보고를 목적으로 계정의 그룹, 사용자, 정책 및 그 이상의 정보를 조회할 수 있도록 허용
- 사용자가 그룹의 멤버십을 관리할 수 있도록 허용
- IAM 사용자를 관리할 수 있도록 허용
- 사용자가 계정 암호 정책을 설정할 수 있도록 허용
- 사용자가 IAM 자격 증명 보고서를 생성하고 검색할 수 있도록 허용
- 모든 IAM 작업 허용(관리자 액세스 권한)

사용자가 보고를 목적으로 계정의 그룹, 사용자, 정책 및 그 이상의 정보를 조회할 수 있도록 허용

다음 정책은 사용자가 `Get` 또는 `List` 문자열로 시작하는 모든 IAM 작업을 호출하고, 보고서를 생성할 수 있게 허용합니다. 예시 정책을 보려면 IAM: 콘솔에 대한 읽기 전용 액세스 허용 섹션을 참조하세요.

사용자가 그룹의 멤버십을 관리할 수 있도록 허용

다음 정책은 사용자가 MarketingGroup이라는 그룹의 멤버십을 업데이트할 수 있도록 허용합니다. 예시 정책을 보려면 IAM: 프로그래밍 방식으로, 그리고 콘솔에서 그룹의 멤버십을 관리하도록 허용 섹션을 참조하세요.

IAM 사용자를 관리할 수 있도록 허용

다음 정책은 사용자가 IAM 사용자 관리와 관련된 모든 작업을 수행할 수 있게 허용하지만 그룹이나 정책의 생성과 같은 다른 엔터티에 대한 작업 수행은 허용하지 않습니다. 허용되는 작업은 다음과 같습니다.

- 사용자 생성(`CreateUser` 작업).
- 사용자 삭제. 이 작업은 `DeleteSigningCertificate`, `DeleteLoginProfile`, `RemoveUserFromGroup`, `DeleteUser` 작업 모두를 수행할 수 있는 권한이 필요합니다.
- 계정 및 그룹의 사용자 나열(`GetUser`, `ListUsers` 및 `ListGroupsForUser` 작업)
- 사용자의 정책 나열 및 제거(`ListUserPolicies`, `ListAttachedUserPolicies`, `DetachUserPolicy`, `DeleteUserPolicy` 작업)
- 사용자의 경로 이름 바꾸기 또는 변경(`UpdateUser` 작업). `Resource` 요소에는 소스 경로와 대상 경로를 모두 다루는 ARN이 포함되어야 합니다. 경로에 대한 자세한 내용은 표시 이름 및 경로 섹션을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUsersToPerformUserActions",
            "Effect": "Allow",
            "Action": [
                "iam:ListPolicies",
                "iam:GetPolicy",
                "iam:UpdateUser",
```

```
                "iam:AttachUserPolicy",
                "iam:ListEntitiesForPolicy",
                "iam:DeleteUserPolicy",
                "iam:DeleteUser",
                "iam:ListUserPolicies",
                "iam:CreateUser",
                "iam:RemoveUserFromGroup",
                "iam:AddUserToGroup",
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:PutUserPolicy",
                "iam:ListAttachedUserPolicies",
                "iam:ListUsers",
                "iam:GetUser",
                "iam:DetachUserPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUsersToSeeStatsOnIAMConsoleDashboard",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccount*",
                "iam:ListAccount*"
            ],
            "Resource": "*"
        }
    ]
}
```

위의 정책에 포함된 많은 권한은 사용자가 AWS Management Console에서 작업을 수행하도록 허용합니다. 사용자가 AWS CLI, AWS SDK 또는 IAM HTTP 쿼리 API를 사용하여 사용자 관련 작업을 수행할 경우 특정 권한이 필요하지 않을 수 있습니다. 예를 들어 사용자가 어떤 사용자에게서 연결을 해제할 정책의 ARN을 이미 알고 있다면 `iam:ListAttachedUserPolicies` 권한이 필요하지 않습니다. 사용자에게 필요한 권한의 정확한 목록은 사용자가 다른 사용자를 관리할 때 수행해야 하는 작업에 따라 다릅니다.

정책에 있는 다음 권한들은 AWS Management Console을 통해 사용자 작업에 액세스할 수 있도록 허용합니다.

- `iam:GetAccount*`
- `iam:ListAccount*`

사용자가 계정 암호 정책을 설정할 수 있도록 허용

일부 사용자들에게 AWS 계정의 암호 정책을 확인하고 업데이트할 수 있는 권한을 부여할 수 있습니다. 예시 정책을 보려면 IAM: 프로그래밍 방식으로, 그리고 콘솔에서 계정 암호 요구 사항을 설정하도록 허용 섹션을 참조하세요.

사용자가 IAM 자격 증명 보고서를 생성하고 검색할 수 있도록 허용

AWS 계정에 있는 모든 사용자가 나열된 보고서를 생성하고 다운로드할 수 있는 권한을 사용자에게 부여할 수 있습니다. 이 보고서에는 암호, 액세스 키, MFA 디바이스, 서명 인증서를 포함한 다양한 사용자 자격 증명의 상태도 나열됩니다. 자격 증명 보고서에 대한 자세한 정보는 섹션을 참조하세요AWS 계정의 자격 증명 보고서 생성 예시 정책을 보려면 IAM: IAM 자격 증명 보고서 생성 및 검색 섹션을 참조하세요.

모든 IAM 작업 허용(관리자 액세스 권한)

일부 사용자들에게 암호, 액세스 키, MFA 디바이스, 사용자 인증서 관리를 비롯하여 IAM에서의 모든 작업을 수행할 수 있는 권한을 부여할 수 있습니다. 다음 예시와 같은 정책은 이러한 권한들을 부여합니다.

> **Warning**
>
> 사용자에게 IAM에 대한 모든 액세스 권한을 부여하면 해당 사용자가 자기 자신 또는 다른 사용자에게 부여할 수 있는 권한에 제한이 없습니다. 사용자는 새로운 IAM 엔터티(사용자나 역할)를 생성하고 그러한 엔터티에 AWS 계정의 모든 리소스에 대한 모든 액세스 권한을 부여할 수 있습니다. 사용자에게 IAM에 대한 모든 액세스 권한을 부여하면 실제로 사용자들에게 AWS 계정의 모든 리소스에 대한 모든 액세스 권한을 부여하는 것입니다. 여기에는 모든 리소스를 삭제하는 권한도 포함됩니다. 이러한 권한은 신뢰할 수 있는 관리자에게만 부여해야 하며, 그러한 관리자에 대해 멀티 팩터 인증(MFA)을 적용해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*",
    "Resource": "*"
  }
}
```

AWS SDK를 사용한 IAM용 코드 예제

다음 코드 예제는 IAM을 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여줍니다.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예시

- AWS SDK 또는 CLI와 함께 DeactivateMfaDevice 사용
- AWS SDK 또는 CLI와 함께 DeleteAccessKey 사용
- AWS SDK 또는 CLI와 함께 DeleteAccountAlias 사용
- AWS SDK 또는 CLI와 함께 DeleteAccountPasswordPolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteGroup 사용
- AWS SDK 또는 CLI와 함께 DeleteGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 DeleteLoginProfile 사용
- AWS SDK 또는 CLI와 함께 DeleteOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 DeletePolicy 사용
- AWS SDK 또는 CLI와 함께 DeletePolicyVersion 사용
- AWS SDK 또는 CLI와 함께 DeleteRole 사용
- AWS SDK 또는 CLI와 함께 DeleteRolePermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 DeleteRolePolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteSAMLProvider 사용
- AWS SDK 또는 CLI와 함께 DeleteServerCertificate 사용
- AWS SDK 또는 CLI와 함께 DeleteServiceLinkedRole 사용
- AWS SDK 또는 CLI와 함께 DeleteSigningCertificate 사용
- AWS SDK 또는 CLI와 함께 DeleteUser 사용
- AWS SDK 또는 CLI와 함께 DeleteUserPermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 DeleteUserPolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteVirtualMfaDevice 사용
- AWS SDK 또는 CLI와 함께 DetachGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 DetachRolePolicy 사용
- AWS SDK 또는 CLI와 함께 DetachUserPolicy 사용
- AWS SDK 또는 CLI와 함께 EnableMfaDevice 사용
- AWS SDK 또는 CLI와 함께 GenerateCredentialReport 사용
- AWS SDK 또는 CLI와 함께 GenerateServiceLastAccessedDetails 사용
- AWS SDK 또는 CLI와 함께 GetAccessKeyLastUsed 사용
- AWS SDK 또는 CLI와 함께 GetAccountAuthorizationDetails 사용

- AWS SDK 또는 CLI와 함께 GetAccountPasswordPolicy 사용
- AWS SDK 또는 CLI와 함께 GetAccountSummary 사용
- AWS SDK 또는 CLI와 함께 GetContextKeysForCustomPolicy 사용
- AWS SDK 또는 CLI와 함께 GetContextKeysForPrincipalPolicy 사용
- AWS SDK 또는 CLI와 함께 GetCredentialReport 사용
- AWS SDK 또는 CLI와 함께 GetGroup 사용
- AWS SDK 또는 CLI와 함께 GetGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 GetInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 GetLoginProfile 사용
- AWS SDK 또는 CLI와 함께 GetOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 GetPolicy 사용
- AWS SDK 또는 CLI와 함께 GetPolicyVersion 사용
- AWS SDK 또는 CLI와 함께 GetRole 사용
- AWS SDK 또는 CLI와 함께 GetRolePolicy 사용
- AWS SDK 또는 CLI와 함께 GetSamlProvider 사용
- AWS SDK 또는 CLI와 함께 GetServerCertificate 사용
- AWS SDK 또는 CLI와 함께 GetServiceLastAccessedDetails 사용
- AWS SDK 또는 CLI와 함께 GetServiceLastAccessedDetailsWithEntities 사용
- AWS SDK 또는 CLI와 함께 GetServiceLinkedRoleDeletionStatus 사용
- AWS SDK 또는 CLI와 함께 GetUser 사용
- AWS SDK 또는 CLI와 함께 GetUserPolicy 사용
- AWS SDK 또는 CLI와 함께 ListAccessKeys 사용
- AWS SDK 또는 CLI와 함께 ListAccountAliases 사용
- AWS SDK 또는 CLI와 함께 ListAttachedGroupPolicies 사용
- AWS SDK 또는 CLI와 함께 ListAttachedRolePolicies 사용
- AWS SDK 또는 CLI와 함께 ListAttachedUserPolicies 사용
- AWS SDK 또는 CLI와 함께 ListEntitiesForPolicy 사용
- AWS SDK 또는 CLI와 함께 ListGroupPolicies 사용
- AWS SDK 또는 CLI와 함께 ListGroups 사용
- AWS SDK 또는 CLI와 함께 ListGroupsForUser 사용

- AWS SDK 또는 CLI와 함께 ListInstanceProfiles 사용
- AWS SDK 또는 CLI와 함께 ListInstanceProfilesForRole 사용
- AWS SDK 또는 CLI와 함께 ListMfaDevices 사용
- AWS SDK 또는 CLI와 함께 ListOpenIdConnectProviders 사용
- AWS SDK 또는 CLI와 함께 ListPolicies 사용
- AWS SDK 또는 CLI와 함께 ListPolicyVersions 사용
- AWS SDK 또는 CLI와 함께 ListRolePolicies 사용
- AWS SDK 또는 CLI와 함께 ListRoleTags 사용
- AWS SDK 또는 CLI와 함께 ListRoles 사용
- AWS SDK 또는 CLI와 함께 ListSAMLProviders 사용
- AWS SDK 또는 CLI와 함께 ListServerCertificates 사용
- AWS SDK 또는 CLI와 함께 ListSigningCertificates 사용
- AWS SDK 또는 CLI와 함께 ListUserPolicies 사용
- AWS SDK 또는 CLI와 함께 ListUserTags 사용
- AWS SDK 또는 CLI와 함께 ListUsers 사용
- AWS SDK 또는 CLI와 함께 ListVirtualMfaDevices 사용
- AWS SDK 또는 CLI와 함께 PutGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 PutRolePermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 PutRolePolicy 사용
- AWS SDK 또는 CLI와 함께 PutUserPermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 PutUserPolicy 사용
- AWS SDK 또는 CLI와 함께 RemoveClientIdFromOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 RemoveRoleFromInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 RemoveUserFromGroup 사용
- AWS SDK 또는 CLI와 함께 ResyncMfaDevice 사용
- AWS SDK 또는 CLI와 함께 SetDefaultPolicyVersion 사용
- AWS SDK 또는 CLI와 함께 TagRole 사용
- AWS SDK 또는 CLI와 함께 TagUser 사용
- AWS SDK 또는 CLI와 함께 UntagRole 사용
- AWS SDK 또는 CLI와 함께 UntagUser 사용

        - AWS SDK 또는 CLI와 함께 DecodeAuthorizationMessage 사용
        - AWS SDK 또는 CLI와 함께 GetFederationToken 사용
        - AWS SDK 또는 CLI와 함께 GetSessionToken 사용
- AWS SDK를 사용하는 AWS STS 시나리오
    - AWS SDK를 사용하여 AWS STS에서 MFA 토큰이 필요한 IAM 역할 수임
    - AWS SDK를 사용하여 페더레이션 사용자를 위해 AWS STS로 URL 구성
    - AWS SDK를 사용하여 AWS STS에서 MFA 토큰이 필요한 세션 토큰 가져오기

AWS SDK를 사용한 IAM용 코드 예제

다음 코드 예제는 IAM을 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여 주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

시작하기

Hello IAM

다음 코드 예제에서는 IAM을 사용하여 시작하는 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace IAMActions;

public class HelloIAM
{
    static async Task Main(string[] args)
    {
        // Getting started with AWS Identity and Access Management (IAM). List
        // the policies for the account.
        var iamClient = new AmazonIdentityManagementServiceClient();

        var listPoliciesPaginator = iamClient.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        Console.WriteLine("Here are the policies defined for your account:\n");
        policies.ForEach(policy =>
        {
            Console.WriteLine($"Created:
 {policy.CreateDate}\t{policy.PolicyName}\t{policy.Description}");
        });
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListPolicies를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iam)

# Set this project's name.
project("hello_iam")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iam.cpp)

target_link_libraries(${PROJECT_NAME}
```

```
        ${AWSSDK_LINK_LIBRARIES})
```

iam.cpp 소스 파일의 코드입니다.

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListPoliciesRequest.h>
#include <iostream>
#include <iomanip>

/*
 *  A "Hello IAM" starter application which initializes an AWS Identity and
 Access Management (IAM) client
 *  and lists the IAM policies.
 *
 *  main function
 *
 *  Usage: 'hello_iam'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        const Aws::String DATE_FORMAT("%Y-%m-%d");
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IAM::IAMClient iamClient(clientConfig);
        Aws::IAM::Model::ListPoliciesRequest request;

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = iamClient.ListPolicies(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Failed to list iam policies: " <<
```

```
                          outcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(55) << "Name" <<
                          std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                          std::setw(64) << "Description" << std::setw(12) <<
                          "CreateDate" << std::endl;
                header = true;
            }

            const auto &policies = outcome.GetResult().GetPolicies();
            for (const auto &policy: policies) {
                std::cout << std::left << std::setw(55) <<
                          policy.GetPolicyName() << std::setw(30) <<
                          policy.GetPolicyId() << std::setw(80) <<
 policy.GetArn() <<
                          std::setw(64) << policy.GetDescription() <<
 std::setw(12) <<
                          policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
 <<
                          std::endl;
            }

            if (outcome.GetResult().GetIsTruncated()) {
                request.SetMarker(outcome.GetResult().GetMarker());
            } else {
                done = true;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ListPolicies를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/iam"
)

// main uses the AWS SDK for Go (v2) to create an AWS Identity and Access
 Management (IAM)
// client and list up to 10 policies in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 iamClient := iam.NewFromConfig(sdkConfig)
 const maxPols = 10
 fmt.Printf("Let's list up to %v policies for your account.\n", maxPols)
 result, err := iamClient.ListPolicies(context.TODO(), &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPols),
 })
 if err != nil {
```

```
  fmt.Printf("Couldn't list policies for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Policies) == 0 {
  fmt.Println("You don't have any policies!")
 } else {
  for _, policy := range result.Policies {
   fmt.Printf("\t%v\n", *policy.PolicyName)
  }
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ListPolicies를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloIAM {
```

```
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListPolicies를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { IAMClient, paginateListPolicies } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listLocalPolicies = async () => {
  /**
   * In v3, the clients expose paginateOperationName APIs that are written using
 async generators so that you can use async iterators in a for await..of loop.
   * https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators
   */
  const paginator = paginateListPolicies(
```

```
    { client, pageSize: 10 },
    // List only customer managed policies.
    { Scope: "Local" },
  );

  console.log("IAM policies defined in your account:");
  let policyCount = 0;
  for await (const page of paginator) {
    if (page.Policies) {
      page.Policies.forEach((p) => {
        console.log(`${p.PolicyName}`);
        policyCount++;
      });
    }
  }
  console.log(`Found ${policyCount} policies.`);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListPolicies를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import boto3


def main():
    """
    Lists the managed policies in your AWS account using the AWS SDK for Python
 (Boto3).
    """
    iam = boto3.client("iam")
```

```
    try:
        # Get a paginator for the list_policies operation
        paginator = iam.get_paginator("list_policies")

        # Iterate through the pages of results
        for page in paginator.paginate(Scope="All", OnlyAttached=False):
            for policy in page["Policies"]:
                print(f"Policy name: {policy['PolicyName']}")
                print(f"  Policy ARN: {policy['Arn']}")
    except boto3.exceptions.BotoCoreError as e:
        print(f"Encountered an error while listing policies: {e}")


if __name__ == "__main__":
    main()
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 ListPolicies를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-iam'
require 'logger'

# IAMManager is a class responsible for managing IAM operations
# such as listing all IAM policies in the current AWS account.
class IAMManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end
```

```
  # Lists and prints all IAM policies in the current AWS account.
  def list_policies
    @logger.info('Here are the IAM policies in your account:')

    paginator = @client.list_policies
    policies = []

    paginator.each_page do |page|
      policies.concat(page.policies)
    end

    if policies.empty?
      @logger.info("You don't have any IAM policies.")
    else
      policies.each do |policy|
        @logger.info("- #{policy.policy_name}")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  iam_client = Aws::IAM::Client.new
  manager = IAMManager.new(iam_client)
  manager.list_policies
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListPolicies를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

src/bin/hello.rs에서

```
use aws_sdk_iam::error::SdkError;
use aws_sdk_iam::operation::list_policies::ListPoliciesError;
use clap::Parser;

const PATH_PREFIX_HELP: &str = "The path prefix for filtering the results.";

#[derive(Debug, clap::Parser)]
#[command(about)]
struct HelloScenarioArgs {
    #[arg(long, default_value="/", help=PATH_PREFIX_HELP)]
    pub path_prefix: String,
}

#[tokio::main]
async fn main() -> Result<(), SdkError<ListPoliciesError>> {
    let sdk_config = aws_config::load_from_env().await;
    let client = aws_sdk_iam::Client::new(&sdk_config);

    let args = HelloScenarioArgs::parse();

    iam_service::list_policies(client, args.path_prefix).await?;

    Ok(())
}
```

src/iam-service-lib.rs에서

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;
```

```
    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();

    Ok(policy_names)
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListPolicies을 참조하십시오.

코드 예시

- AWS SDK를 사용한 IAM용 기본 예제
  - Hello IAM
  - AWS SDK를 사용한 IAM 작업
    - AWS SDK 또는 CLI와 함께 AddClientIdToOpenIdConnectProvider 사용
    - AWS SDK 또는 CLI와 함께 AddRoleToInstanceProfile 사용
    - AWS SDK 또는 CLI와 함께 AddUserToGroup 사용
    - AWS SDK 또는 CLI와 함께 AttachGroupPolicy 사용
    - AWS SDK 또는 CLI와 함께 AttachRolePolicy 사용
    - AWS SDK 또는 CLI와 함께 AttachUserPolicy 사용
    - AWS SDK 또는 CLI와 함께 ChangePassword 사용
    - AWS SDK 또는 CLI와 함께 CreateAccessKey 사용
    - AWS SDK 또는 CLI와 함께 CreateAccountAlias 사용
    - AWS SDK 또는 CLI와 함께 CreateGroup 사용
    - AWS SDK 또는 CLI와 함께 CreateInstanceProfile 사용
    - AWS SDK 또는 CLI와 함께 CreateLoginProfile 사용
    - AWS SDK 또는 CLI와 함께 CreateOpenIdConnectProvider 사용
    - AWS SDK 또는 CLI와 함께 CreatePolicy 사용

- AWS SDK 또는 CLI와 함께 CreatePolicyVersion 사용
- AWS SDK 또는 CLI와 함께 CreateRole 사용
- AWS SDK 또는 CLI와 함께 CreateSAMLProvider 사용
- AWS SDK 또는 CLI와 함께 CreateServiceLinkedRole 사용
- AWS SDK 또는 CLI와 함께 CreateUser 사용
- AWS SDK 또는 CLI와 함께 CreateVirtualMfaDevice 사용
- AWS SDK 또는 CLI와 함께 DeactivateMfaDevice 사용
- AWS SDK 또는 CLI와 함께 DeleteAccessKey 사용
- AWS SDK 또는 CLI와 함께 DeleteAccountAlias 사용
- AWS SDK 또는 CLI와 함께 DeleteAccountPasswordPolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteGroup 사용
- AWS SDK 또는 CLI와 함께 DeleteGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 DeleteLoginProfile 사용
- AWS SDK 또는 CLI와 함께 DeleteOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 DeletePolicy 사용
- AWS SDK 또는 CLI와 함께 DeletePolicyVersion 사용
- AWS SDK 또는 CLI와 함께 DeleteRole 사용
- AWS SDK 또는 CLI와 함께 DeleteRolePermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 DeleteRolePolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteSAMLProvider 사용
- AWS SDK 또는 CLI와 함께 DeleteServerCertificate 사용
- AWS SDK 또는 CLI와 함께 DeleteServiceLinkedRole 사용
- AWS SDK 또는 CLI와 함께 DeleteSigningCertificate 사용
- AWS SDK 또는 CLI와 함께 DeleteUser 사용
- AWS SDK 또는 CLI와 함께 DeleteUserPermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 DeleteUserPolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteVirtualMfaDevice 사용
- AWS SDK 또는 CLI와 함께 DetachGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 DetachRolePolicy 사용

- AWS SDK 또는 CLI와 함께 DetachUserPolicy 사용
- AWS SDK 또는 CLI와 함께 EnableMfaDevice 사용
- AWS SDK 또는 CLI와 함께 GenerateCredentialReport 사용
- AWS SDK 또는 CLI와 함께 GenerateServiceLastAccessedDetails 사용
- AWS SDK 또는 CLI와 함께 GetAccessKeyLastUsed 사용
- AWS SDK 또는 CLI와 함께 GetAccountAuthorizationDetails 사용
- AWS SDK 또는 CLI와 함께 GetAccountPasswordPolicy 사용
- AWS SDK 또는 CLI와 함께 GetAccountSummary 사용
- AWS SDK 또는 CLI와 함께 GetContextKeysForCustomPolicy 사용
- AWS SDK 또는 CLI와 함께 GetContextKeysForPrincipalPolicy 사용
- AWS SDK 또는 CLI와 함께 GetCredentialReport 사용
- AWS SDK 또는 CLI와 함께 GetGroup 사용
- AWS SDK 또는 CLI와 함께 GetGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 GetInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 GetLoginProfile 사용
- AWS SDK 또는 CLI와 함께 GetOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 GetPolicy 사용
- AWS SDK 또는 CLI와 함께 GetPolicyVersion 사용
- AWS SDK 또는 CLI와 함께 GetRole 사용
- AWS SDK 또는 CLI와 함께 GetRolePolicy 사용
- AWS SDK 또는 CLI와 함께 GetSamlProvider 사용
- AWS SDK 또는 CLI와 함께 GetServerCertificate 사용
- AWS SDK 또는 CLI와 함께 GetServiceLastAccessedDetails 사용
- AWS SDK 또는 CLI와 함께 GetServiceLastAccessedDetailsWithEntities 사용
- AWS SDK 또는 CLI와 함께 GetServiceLinkedRoleDeletionStatus 사용
- AWS SDK 또는 CLI와 함께 GetUser 사용
- AWS SDK 또는 CLI와 함께 GetUserPolicy 사용
- AWS SDK 또는 CLI와 함께 ListAccessKeys 사용
- AWS SDK 또는 CLI와 함께 ListAccountAliases 사용
- AWS SDK 또는 CLI와 함께 ListAttachedGroupPolicies 사용

- AWS SDK 또는 CLI와 함께 ListAttachedRolePolicies 사용
- AWS SDK 또는 CLI와 함께 ListAttachedUserPolicies 사용
- AWS SDK 또는 CLI와 함께 ListEntitiesForPolicy 사용
- AWS SDK 또는 CLI와 함께 ListGroupPolicies 사용
- AWS SDK 또는 CLI와 함께 ListGroups 사용
- AWS SDK 또는 CLI와 함께 ListGroupsForUser 사용
- AWS SDK 또는 CLI와 함께 ListInstanceProfiles 사용
- AWS SDK 또는 CLI와 함께 ListInstanceProfilesForRole 사용
- AWS SDK 또는 CLI와 함께 ListMfaDevices 사용
- AWS SDK 또는 CLI와 함께 ListOpenIdConnectProviders 사용
- AWS SDK 또는 CLI와 함께 ListPolicies 사용
- AWS SDK 또는 CLI와 함께 ListPolicyVersions 사용
- AWS SDK 또는 CLI와 함께 ListRolePolicies 사용
- AWS SDK 또는 CLI와 함께 ListRoleTags 사용
- AWS SDK 또는 CLI와 함께 ListRoles 사용
- AWS SDK 또는 CLI와 함께 ListSAMLProviders 사용
- AWS SDK 또는 CLI와 함께 ListServerCertificates 사용
- AWS SDK 또는 CLI와 함께 ListSigningCertificates 사용
- AWS SDK 또는 CLI와 함께 ListUserPolicies 사용
- AWS SDK 또는 CLI와 함께 ListUserTags 사용
- AWS SDK 또는 CLI와 함께 ListUsers 사용
- AWS SDK 또는 CLI와 함께 ListVirtualMfaDevices 사용
- AWS SDK 또는 CLI와 함께 PutGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 PutRolePermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 PutRolePolicy 사용
- AWS SDK 또는 CLI와 함께 PutUserPermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 PutUserPolicy 사용
- AWS SDK 또는 CLI와 함께 RemoveClientIdFromOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 RemoveRoleFromInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 RemoveUserFromGroup 사용

- AWS SDK 또는 CLI와 함께 ResyncMfaDevice 사용
- AWS SDK 또는 CLI와 함께 SetDefaultPolicyVersion 사용
- AWS SDK 또는 CLI와 함께 TagRole 사용
- AWS SDK 또는 CLI와 함께 TagUser 사용
- AWS SDK 또는 CLI와 함께 UntagRole 사용
- AWS SDK 또는 CLI와 함께 UntagUser 사용
- AWS SDK 또는 CLI와 함께 UpdateAccessKey 사용
- AWS SDK 또는 CLI와 함께 UpdateAccountPasswordPolicy 사용
- AWS SDK 또는 CLI와 함께 UpdateAssumeRolePolicy 사용
- AWS SDK 또는 CLI와 함께 UpdateGroup 사용
- AWS SDK 또는 CLI와 함께 UpdateLoginProfile 사용
- AWS SDK 또는 CLI와 함께 UpdateOpenIdConnectProviderThumbprint 사용
- AWS SDK 또는 CLI와 함께 UpdateRole 사용
- AWS SDK 또는 CLI와 함께 UpdateRoleDescription 사용
- AWS SDK 또는 CLI와 함께 UpdateSamlProvider 사용
- AWS SDK 또는 CLI와 함께 UpdateServerCertificate 사용
- AWS SDK 또는 CLI와 함께 UpdateSigningCertificate 사용
- AWS SDK 또는 CLI와 함께 UpdateUser 사용
- AWS SDK 또는 CLI와 함께 UploadServerCertificate 사용
- AWS SDK 또는 CLI와 함께 UploadSigningCertificate 사용

- AWS SDK를 사용하는 IAM 시나리오
  - AWS SDK를 사용하여 복원력이 뛰어난 서비스 구축 및 관리
  - AWS SDK를 사용하여 IAM 그룹을 생성하고 사용자를 그룹에 추가
  - AWS SDK를 사용하여 AWS STS에서 IAM 사용자 생성 및 역할 수임
  - AWS SDK를 사용하여 읽기 전용 및 읽기-쓰기 IAM 사용자 생성
  - AWS SDK를 사용하여 IAM 액세스 키 관리
  - AWS SDK를 사용하여 IAM 정책 관리
  - AWS SDK를 사용하여 IAM 역할 관리
  - AWS SDK를 사용하여 IAM 계정 관리
  - AWS SDK를 사용하여 IAM 정책 버전 롤백

- AWS SDK를 사용하여 IAM 정책 빌더 API 작업

AWS SDK를 사용한 IAM용 기본 예제

다음 코드 예제에서는 AWS SDK에서 AWS Identity and Access Management(IAM)의 기본 사항을 사용하는 방법을 보여줍니다.

예시

- Hello IAM
- AWS SDK를 사용한 IAM 작업
  - AWS SDK 또는 CLI와 함께 AddClientIdToOpenIdConnectProvider 사용
  - AWS SDK 또는 CLI와 함께 AddRoleToInstanceProfile 사용
  - AWS SDK 또는 CLI와 함께 AddUserToGroup 사용
  - AWS SDK 또는 CLI와 함께 AttachGroupPolicy 사용
  - AWS SDK 또는 CLI와 함께 AttachRolePolicy 사용
  - AWS SDK 또는 CLI와 함께 AttachUserPolicy 사용
  - AWS SDK 또는 CLI와 함께 ChangePassword 사용
  - AWS SDK 또는 CLI와 함께 CreateAccessKey 사용
  - AWS SDK 또는 CLI와 함께 CreateAccountAlias 사용
  - AWS SDK 또는 CLI와 함께 CreateGroup 사용
  - AWS SDK 또는 CLI와 함께 CreateInstanceProfile 사용
  - AWS SDK 또는 CLI와 함께 CreateLoginProfile 사용
  - AWS SDK 또는 CLI와 함께 CreateOpenIdConnectProvider 사용
  - AWS SDK 또는 CLI와 함께 CreatePolicy 사용
  - AWS SDK 또는 CLI와 함께 CreatePolicyVersion 사용
  - AWS SDK 또는 CLI와 함께 CreateRole 사용
  - AWS SDK 또는 CLI와 함께 CreateSAMLProvider 사용
  - AWS SDK 또는 CLI와 함께 CreateServiceLinkedRole 사용
  - AWS SDK 또는 CLI와 함께 CreateUser 사용
  - AWS SDK 또는 CLI와 함께 CreateVirtualMfaDevice 사용
  - AWS SDK 또는 CLI와 함께 DeactivateMfaDevice 사용
  - AWS SDK 또는 CLI와 함께 DeleteAccessKey 사용

- AWS SDK 또는 CLI와 함께 DeleteAccountAlias 사용
- AWS SDK 또는 CLI와 함께 DeleteAccountPasswordPolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteGroup 사용
- AWS SDK 또는 CLI와 함께 DeleteGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 DeleteLoginProfile 사용
- AWS SDK 또는 CLI와 함께 DeleteOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 DeletePolicy 사용
- AWS SDK 또는 CLI와 함께 DeletePolicyVersion 사용
- AWS SDK 또는 CLI와 함께 DeleteRole 사용
- AWS SDK 또는 CLI와 함께 DeleteRolePermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 DeleteRolePolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteSAMLProvider 사용
- AWS SDK 또는 CLI와 함께 DeleteServerCertificate 사용
- AWS SDK 또는 CLI와 함께 DeleteServiceLinkedRole 사용
- AWS SDK 또는 CLI와 함께 DeleteSigningCertificate 사용
- AWS SDK 또는 CLI와 함께 DeleteUser 사용
- AWS SDK 또는 CLI와 함께 DeleteUserPermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 DeleteUserPolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteVirtualMfaDevice 사용
- AWS SDK 또는 CLI와 함께 DetachGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 DetachRolePolicy 사용
- AWS SDK 또는 CLI와 함께 DetachUserPolicy 사용
- AWS SDK 또는 CLI와 함께 EnableMfaDevice 사용
- AWS SDK 또는 CLI와 함께 GenerateCredentialReport 사용
- AWS SDK 또는 CLI와 함께 GenerateServiceLastAccessedDetails 사용
- AWS SDK 또는 CLI와 함께 GetAccessKeyLastUsed 사용
- AWS SDK 또는 CLI와 함께 GetAccountAuthorizationDetails 사용
- AWS SDK 또는 CLI와 함께 GetAccountPasswordPolicy 사용
- AWS SDK 또는 CLI와 함께 GetAccountSummary 사용

- AWS SDK 또는 CLI와 함께 GetContextKeysForCustomPolicy 사용
- AWS SDK 또는 CLI와 함께 GetContextKeysForPrincipalPolicy 사용
- AWS SDK 또는 CLI와 함께 GetCredentialReport 사용
- AWS SDK 또는 CLI와 함께 GetGroup 사용
- AWS SDK 또는 CLI와 함께 GetGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 GetInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 GetLoginProfile 사용
- AWS SDK 또는 CLI와 함께 GetOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 GetPolicy 사용
- AWS SDK 또는 CLI와 함께 GetPolicyVersion 사용
- AWS SDK 또는 CLI와 함께 GetRole 사용
- AWS SDK 또는 CLI와 함께 GetRolePolicy 사용
- AWS SDK 또는 CLI와 함께 GetSamlProvider 사용
- AWS SDK 또는 CLI와 함께 GetServerCertificate 사용
- AWS SDK 또는 CLI와 함께 GetServiceLastAccessedDetails 사용
- AWS SDK 또는 CLI와 함께 GetServiceLastAccessedDetailsWithEntities 사용
- AWS SDK 또는 CLI와 함께 GetServiceLinkedRoleDeletionStatus 사용
- AWS SDK 또는 CLI와 함께 GetUser 사용
- AWS SDK 또는 CLI와 함께 GetUserPolicy 사용
- AWS SDK 또는 CLI와 함께 ListAccessKeys 사용
- AWS SDK 또는 CLI와 함께 ListAccountAliases 사용
- AWS SDK 또는 CLI와 함께 ListAttachedGroupPolicies 사용
- AWS SDK 또는 CLI와 함께 ListAttachedRolePolicies 사용
- AWS SDK 또는 CLI와 함께 ListAttachedUserPolicies 사용
- AWS SDK 또는 CLI와 함께 ListEntitiesForPolicy 사용
- AWS SDK 또는 CLI와 함께 ListGroupPolicies 사용
- AWS SDK 또는 CLI와 함께 ListGroups 사용
- AWS SDK 또는 CLI와 함께 ListGroupsForUser 사용
- AWS SDK 또는 CLI와 함께 ListInstanceProfiles 사용
- AWS SDK 또는 CLI와 함께 ListInstanceProfilesForRole 사용

- AWS SDK 또는 CLI와 함께 ListMfaDevices 사용
- AWS SDK 또는 CLI와 함께 ListOpenIdConnectProviders 사용
- AWS SDK 또는 CLI와 함께 ListPolicies 사용
- AWS SDK 또는 CLI와 함께 ListPolicyVersions 사용
- AWS SDK 또는 CLI와 함께 ListRolePolicies 사용
- AWS SDK 또는 CLI와 함께 ListRoleTags 사용
- AWS SDK 또는 CLI와 함께 ListRoles 사용
- AWS SDK 또는 CLI와 함께 ListSAMLProviders 사용
- AWS SDK 또는 CLI와 함께 ListServerCertificates 사용
- AWS SDK 또는 CLI와 함께 ListSigningCertificates 사용
- AWS SDK 또는 CLI와 함께 ListUserPolicies 사용
- AWS SDK 또는 CLI와 함께 ListUserTags 사용
- AWS SDK 또는 CLI와 함께 ListUsers 사용
- AWS SDK 또는 CLI와 함께 ListVirtualMfaDevices 사용
- AWS SDK 또는 CLI와 함께 PutGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 PutRolePermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 PutRolePolicy 사용
- AWS SDK 또는 CLI와 함께 PutUserPermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 PutUserPolicy 사용
- AWS SDK 또는 CLI와 함께 RemoveClientIdFromOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 RemoveRoleFromInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 RemoveUserFromGroup 사용
- AWS SDK 또는 CLI와 함께 ResyncMfaDevice 사용
- AWS SDK 또는 CLI와 함께 SetDefaultPolicyVersion 사용
- AWS SDK 또는 CLI와 함께 TagRole 사용
- AWS SDK 또는 CLI와 함께 TagUser 사용
- AWS SDK 또는 CLI와 함께 UntagRole 사용
- AWS SDK 또는 CLI와 함께 UntagUser 사용
- AWS SDK 또는 CLI와 함께 UpdateAccessKey 사용
- AWS SDK 또는 CLI와 함께 UpdateAccountPasswordPolicy 사용

- AWS SDK 또는 CLI와 함께 UpdateAssumeRolePolicy 사용
- AWS SDK 또는 CLI와 함께 UpdateGroup 사용
- AWS SDK 또는 CLI와 함께 UpdateLoginProfile 사용
- AWS SDK 또는 CLI와 함께 UpdateOpenIdConnectProviderThumbprint 사용
- AWS SDK 또는 CLI와 함께 UpdateRole 사용
- AWS SDK 또는 CLI와 함께 UpdateRoleDescription 사용
- AWS SDK 또는 CLI와 함께 UpdateSamlProvider 사용
- AWS SDK 또는 CLI와 함께 UpdateServerCertificate 사용
- AWS SDK 또는 CLI와 함께 UpdateSigningCertificate 사용
- AWS SDK 또는 CLI와 함께 UpdateUser 사용
- AWS SDK 또는 CLI와 함께 UploadServerCertificate 사용
- AWS SDK 또는 CLI와 함께 UploadSigningCertificate 사용

Hello IAM

다음 코드 예제에서는 IAM을 사용하여 시작하는 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace IAMActions;

public class HelloIAM
{
    static async Task Main(string[] args)
    {
        // Getting started with AWS Identity and Access Management (IAM). List
        // the policies for the account.
        var iamClient = new AmazonIdentityManagementServiceClient();
```

```
        var listPoliciesPaginator = iamClient.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        Console.WriteLine("Here are the policies defined for your account:\n");
        policies.ForEach(policy =>
        {
            Console.WriteLine($"Created:
 {policy.CreateDate}\t{policy.PolicyName}\t{policy.Description}");
        });
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListPolicies를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iam)

# Set this project's name.
project("hello_iam")
```

```
# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iam.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

iam.cpp 소스 파일의 코드입니다.

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListPoliciesRequest.h>
#include <iostream>
```

```
#include <iomanip>

/*
 *  A "Hello IAM" starter application which initializes an AWS Identity and
 Access Management (IAM) client
 *  and lists the IAM policies.
 *
 *  main function
 *
 *  Usage: 'hello_iam'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        const Aws::String DATE_FORMAT("%Y-%m-%d");
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IAM::IAMClient iamClient(clientConfig);
        Aws::IAM::Model::ListPoliciesRequest request;

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = iamClient.ListPolicies(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Failed to list iam policies: " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(55) << "Name" <<
                          std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                          std::setw(64) << "Description" << std::setw(12) <<
                          "CreateDate" << std::endl;
```

```
                header = true;
            }

            const auto &policies = outcome.GetResult().GetPolicies();
            for (const auto &policy: policies) {
                std::cout << std::left << std::setw(55) <<
                          policy.GetPolicyName() << std::setw(30) <<
                          policy.GetPolicyId() << std::setw(80) <<
 policy.GetArn() <<
                          std::setw(64) << policy.GetDescription() <<
 std::setw(12) <<
                          policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
 <<
                          std::endl;
            }

            if (outcome.GetResult().GetIsTruncated()) {
                request.SetMarker(outcome.GetResult().GetMarker());
            } else {
                done = true;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ListPolicies를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/iam"
)

// main uses the AWS SDK for Go (v2) to create an AWS Identity and Access
 Management (IAM)
// client and list up to 10 policies in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 iamClient := iam.NewFromConfig(sdkConfig)
 const maxPols = 10
 fmt.Printf("Let's list up to %v policies for your account.\n", maxPols)
 result, err := iamClient.ListPolicies(context.TODO(), &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPols),
 })
 if err != nil {
  fmt.Printf("Couldn't list policies for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Policies) == 0 {
  fmt.Println("You don't have any policies!")
 } else {
  for _, policy := range result.Policies {
   fmt.Printf("\t%v\n", *policy.PolicyName)
  }
 }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ListPolicies를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
```

```
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListPolicies를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { IAMClient, paginateListPolicies } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listLocalPolicies = async () => {
  /**
   * In v3, the clients expose paginateOperationName APIs that are written using
 async generators so that you can use async iterators in a for await..of loop.
   * https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators
   */
  const paginator = paginateListPolicies(
    { client, pageSize: 10 },
    // List only customer managed policies.
    { Scope: "Local" },
  );

  console.log("IAM policies defined in your account:");
  let policyCount = 0;
  for await (const page of paginator) {
    if (page.Policies) {
      page.Policies.forEach((p) => {
        console.log(`${p.PolicyName}`);
```

```
        policyCount++;
      });
    }
  }
  console.log(`Found ${policyCount} policies.`);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListPolicies를 참조하십시오.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import boto3


def main():
    """
    Lists the managed policies in your AWS account using the AWS SDK for Python
 (Boto3).
    """
    iam = boto3.client("iam")

    try:
        # Get a paginator for the list_policies operation
        paginator = iam.get_paginator("list_policies")

        # Iterate through the pages of results
        for page in paginator.paginate(Scope="All", OnlyAttached=False):
            for policy in page["Policies"]:
                print(f"Policy name: {policy['PolicyName']}")
                print(f"  Policy ARN: {policy['Arn']}")
    except boto3.exceptions.BotoCoreError as e:
        print(f"Encountered an error while listing policies: {e}")
```

```
if __name__ == "__main__":
    main()
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 ListPolicies를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-iam'
require 'logger'

# IAMManager is a class responsible for managing IAM operations
# such as listing all IAM policies in the current AWS account.
class IAMManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all IAM policies in the current AWS account.
  def list_policies
    @logger.info('Here are the IAM policies in your account:')

    paginator = @client.list_policies
    policies = []

    paginator.each_page do |page|
      policies.concat(page.policies)
    end
```

```
    if policies.empty?
      @logger.info("You don't have any IAM policies.")
    else
      policies.each do |policy|
        @logger.info("- #{policy.policy_name}")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  iam_client = Aws::IAM::Client.new
  manager = IAMManager.new(iam_client)
  manager.list_policies
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListPolicies를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

src/bin/hello.rs에서

```
use aws_sdk_iam::error::SdkError;
use aws_sdk_iam::operation::list_policies::ListPoliciesError;
use clap::Parser;

const PATH_PREFIX_HELP: &str = "The path prefix for filtering the results.";

#[derive(Debug, clap::Parser)]
#[command(about)]
struct HelloScenarioArgs {
```

```
    #[arg(long, default_value="/", help=PATH_PREFIX_HELP)]
    pub path_prefix: String,
}

#[tokio::main]
async fn main() -> Result<(), SdkError<ListPoliciesError>> {
    let sdk_config = aws_config::load_from_env().await;
    let client = aws_sdk_iam::Client::new(&sdk_config);

    let args = HelloScenarioArgs::parse();

    iam_service::list_policies(client, args.path_prefix).await?;

    Ok(())
}
```

src/iam-service-lib.rs에서

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();
```

```
    Ok(policy_names)
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListPolicies을 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 IAM 작업

다음 코드 예제는 AWS SDK를 통해 개별 IAM 작업을 수행하는 방법을 보여줍니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

이들 발췌문은 IAM API를 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. AWS SDK를 사용하는 IAM 시나리오 에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 AWS Identity and Access Management(IAM) API 참조를 참조하세요.

예시

- AWS SDK 또는 CLI와 함께 AddClientIdToOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 AddRoleToInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 AddUserToGroup 사용
- AWS SDK 또는 CLI와 함께 AttachGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 AttachRolePolicy 사용
- AWS SDK 또는 CLI와 함께 AttachUserPolicy 사용
- AWS SDK 또는 CLI와 함께 ChangePassword 사용
- AWS SDK 또는 CLI와 함께 CreateAccessKey 사용
- AWS SDK 또는 CLI와 함께 CreateAccountAlias 사용
- AWS SDK 또는 CLI와 함께 CreateGroup 사용
- AWS SDK 또는 CLI와 함께 CreateInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 CreateLoginProfile 사용

- AWS SDK 또는 CLI와 함께 CreateOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 CreatePolicy 사용
- AWS SDK 또는 CLI와 함께 CreatePolicyVersion 사용
- AWS SDK 또는 CLI와 함께 CreateRole 사용
- AWS SDK 또는 CLI와 함께 CreateSAMLProvider 사용
- AWS SDK 또는 CLI와 함께 CreateServiceLinkedRole 사용
- AWS SDK 또는 CLI와 함께 CreateUser 사용
- AWS SDK 또는 CLI와 함께 CreateVirtualMfaDevice 사용
- AWS SDK 또는 CLI와 함께 DeactivateMfaDevice 사용
- AWS SDK 또는 CLI와 함께 DeleteAccessKey 사용
- AWS SDK 또는 CLI와 함께 DeleteAccountAlias 사용
- AWS SDK 또는 CLI와 함께 DeleteAccountPasswordPolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteGroup 사용
- AWS SDK 또는 CLI와 함께 DeleteGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 DeleteLoginProfile 사용
- AWS SDK 또는 CLI와 함께 DeleteOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 DeletePolicy 사용
- AWS SDK 또는 CLI와 함께 DeletePolicyVersion 사용
- AWS SDK 또는 CLI와 함께 DeleteRole 사용
- AWS SDK 또는 CLI와 함께 DeleteRolePermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 DeleteRolePolicy 사용
- AWS SDK 또는 CLI와 함께 DeleteSAMLProvider 사용
- AWS SDK 또는 CLI와 함께 DeleteServerCertificate 사용
- AWS SDK 또는 CLI와 함께 DeleteServiceLinkedRole 사용
- AWS SDK 또는 CLI와 함께 DeleteSigningCertificate 사용
- AWS SDK 또는 CLI와 함께 DeleteUser 사용
- AWS SDK 또는 CLI와 함께 DeleteUserPermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 DeleteUserPolicy 사용

- AWS SDK 또는 CLI와 함께 DeleteVirtualMfaDevice 사용
- AWS SDK 또는 CLI와 함께 DetachGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 DetachRolePolicy 사용
- AWS SDK 또는 CLI와 함께 DetachUserPolicy 사용
- AWS SDK 또는 CLI와 함께 EnableMfaDevice 사용
- AWS SDK 또는 CLI와 함께 GenerateCredentialReport 사용
- AWS SDK 또는 CLI와 함께 GenerateServiceLastAccessedDetails 사용
- AWS SDK 또는 CLI와 함께 GetAccessKeyLastUsed 사용
- AWS SDK 또는 CLI와 함께 GetAccountAuthorizationDetails 사용
- AWS SDK 또는 CLI와 함께 GetAccountPasswordPolicy 사용
- AWS SDK 또는 CLI와 함께 GetAccountSummary 사용
- AWS SDK 또는 CLI와 함께 GetContextKeysForCustomPolicy 사용
- AWS SDK 또는 CLI와 함께 GetContextKeysForPrincipalPolicy 사용
- AWS SDK 또는 CLI와 함께 GetCredentialReport 사용
- AWS SDK 또는 CLI와 함께 GetGroup 사용
- AWS SDK 또는 CLI와 함께 GetGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 GetInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 GetLoginProfile 사용
- AWS SDK 또는 CLI와 함께 GetOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 GetPolicy 사용
- AWS SDK 또는 CLI와 함께 GetPolicyVersion 사용
- AWS SDK 또는 CLI와 함께 GetRole 사용
- AWS SDK 또는 CLI와 함께 GetRolePolicy 사용
- AWS SDK 또는 CLI와 함께 GetSamlProvider 사용
- AWS SDK 또는 CLI와 함께 GetServerCertificate 사용
- AWS SDK 또는 CLI와 함께 GetServiceLastAccessedDetails 사용
- AWS SDK 또는 CLI와 함께 GetServiceLastAccessedDetailsWithEntities 사용
- AWS SDK 또는 CLI와 함께 GetServiceLinkedRoleDeletionStatus 사용
- AWS SDK 또는 CLI와 함께 GetUser 사용

- AWS SDK 또는 CLI와 함께 GetUserPolicy 사용
- AWS SDK 또는 CLI와 함께 ListAccessKeys 사용
- AWS SDK 또는 CLI와 함께 ListAccountAliases 사용
- AWS SDK 또는 CLI와 함께 ListAttachedGroupPolicies 사용
- AWS SDK 또는 CLI와 함께 ListAttachedRolePolicies 사용
- AWS SDK 또는 CLI와 함께 ListAttachedUserPolicies 사용
- AWS SDK 또는 CLI와 함께 ListEntitiesForPolicy 사용
- AWS SDK 또는 CLI와 함께 ListGroupPolicies 사용
- AWS SDK 또는 CLI와 함께 ListGroups 사용
- AWS SDK 또는 CLI와 함께 ListGroupsForUser 사용
- AWS SDK 또는 CLI와 함께 ListInstanceProfiles 사용
- AWS SDK 또는 CLI와 함께 ListInstanceProfilesForRole 사용
- AWS SDK 또는 CLI와 함께 ListMfaDevices 사용
- AWS SDK 또는 CLI와 함께 ListOpenIdConnectProviders 사용
- AWS SDK 또는 CLI와 함께 ListPolicies 사용
- AWS SDK 또는 CLI와 함께 ListPolicyVersions 사용
- AWS SDK 또는 CLI와 함께 ListRolePolicies 사용
- AWS SDK 또는 CLI와 함께 ListRoleTags 사용
- AWS SDK 또는 CLI와 함께 ListRoles 사용
- AWS SDK 또는 CLI와 함께 ListSAMLProviders 사용
- AWS SDK 또는 CLI와 함께 ListServerCertificates 사용
- AWS SDK 또는 CLI와 함께 ListSigningCertificates 사용
- AWS SDK 또는 CLI와 함께 ListUserPolicies 사용
- AWS SDK 또는 CLI와 함께 ListUserTags 사용
- AWS SDK 또는 CLI와 함께 ListUsers 사용
- AWS SDK 또는 CLI와 함께 ListVirtualMfaDevices 사용
- AWS SDK 또는 CLI와 함께 PutGroupPolicy 사용
- AWS SDK 또는 CLI와 함께 PutRolePermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 PutRolePolicy 사용

- AWS SDK 또는 CLI와 함께 PutUserPermissionsBoundary 사용
- AWS SDK 또는 CLI와 함께 PutUserPolicy 사용
- AWS SDK 또는 CLI와 함께 RemoveClientIdFromOpenIdConnectProvider 사용
- AWS SDK 또는 CLI와 함께 RemoveRoleFromInstanceProfile 사용
- AWS SDK 또는 CLI와 함께 RemoveUserFromGroup 사용
- AWS SDK 또는 CLI와 함께 ResyncMfaDevice 사용
- AWS SDK 또는 CLI와 함께 SetDefaultPolicyVersion 사용
- AWS SDK 또는 CLI와 함께 TagRole 사용
- AWS SDK 또는 CLI와 함께 TagUser 사용
- AWS SDK 또는 CLI와 함께 UntagRole 사용
- AWS SDK 또는 CLI와 함께 UntagUser 사용
- AWS SDK 또는 CLI와 함께 UpdateAccessKey 사용
- AWS SDK 또는 CLI와 함께 UpdateAccountPasswordPolicy 사용
- AWS SDK 또는 CLI와 함께 UpdateAssumeRolePolicy 사용
- AWS SDK 또는 CLI와 함께 UpdateGroup 사용
- AWS SDK 또는 CLI와 함께 UpdateLoginProfile 사용
- AWS SDK 또는 CLI와 함께 UpdateOpenIdConnectProviderThumbprint 사용
- AWS SDK 또는 CLI와 함께 UpdateRole 사용
- AWS SDK 또는 CLI와 함께 UpdateRoleDescription 사용
- AWS SDK 또는 CLI와 함께 UpdateSamlProvider 사용
- AWS SDK 또는 CLI와 함께 UpdateServerCertificate 사용
- AWS SDK 또는 CLI와 함께 UpdateSigningCertificate 사용
- AWS SDK 또는 CLI와 함께 UpdateUser 사용
- AWS SDK 또는 CLI와 함께 UploadServerCertificate 사용
- AWS SDK 또는 CLI와 함께 UploadSigningCertificate 사용

AWS SDK 또는 CLI와 함께 **`AddClientIdToOpenIdConnectProvider`** 사용

다음 코드 예제는 `AddClientIdToOpenIdConnectProvider`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

OIDC(Open-ID Connect) 제공업체에 클라이언트 ID(대상) 추가

다음 `add-client-id-to-open-id-connect-provider` 명령은 `server.example.com`이라는 OIDC 제공업체에게 클라이언트 ID `my-application-ID`를 추가합니다.

```
aws iam add-client-id-to-open-id-connect-provider \
    --client-id my-application-ID \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
server.example.com
```

이 명령은 출력을 생성하지 않습니다.

OIDC 제공업체를 생성하려면 `create-open-id-connect-provider` 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM에서 OIDC(OpenID Connect) ID 제공업체 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 AddClientIdToOpenIdConnectProvider를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 명령은 클라이언트 ID 또는 대상 **`my-application-ID`**를 **`server.example.com`**이라는 기존 OIDC 제공업체에 추가합니다.

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 AddClientIdToOpenIdConnectProvider를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **AddRoleToInstanceProfile** 사용

다음 코드 예제는 AddRoleToInstanceProfile의 사용 방법을 보여 줍니다.

CLI

AWS CLI

인스턴스 프로파일에 역할 추가

다음 add-role-to-instance-profile 명령은 이름이 Webserver인 인스턴스 프로파일에 이름이 S3Access인 역할을 추가합니다.

```
aws iam add-role-to-instance-profile \
    --role-name S3Access \
    --instance-profile-name Webserver
```

이 명령은 출력을 생성하지 않습니다.

인스턴스 프로파일을 생성하려면 create-instance-profile 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 AddRoleToInstanceProfile을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 명령은 **webserver**라는 기존 인스턴스 프로파일에 **S3Access**라는 역할을 추가합니다. 인스턴스 프로파일을 생성하려면 **New-IAMInstanceProfile** 명령을 사용합니다. 이 명령을 사용하여 인스턴스 프로파일을 생성하고 이를 역할과 연결한 후 EC2 인스턴스에 연결할 수 있습니다. 이를 수행하려면 **InstanceProfile_Arn** 또는 **InstanceProfile-Name** 파라미터와 함께 **New-EC2Instance** cmdlet을 사용하여 새 인스턴스를 시작합니다.

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName
 "webserver"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 AddRoleToInstanceProfile을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **AddUserToGroup** 사용

다음 코드 예제는 AddUserToGroup의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 그룹 생성 및 사용자 추가

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Add an existing IAM user to an existing IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to add.</param>
    /// <param name="groupName">The name of the group to add the user to.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AddUserToGroupAsync(string userName, string
groupName)
    {
        var response = await _IAMService.AddUserToGroupAsync(new
AddUserToGroupRequest
        {
            GroupName = groupName,
            UserName = userName,
        });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 AddUserToGroup을 참조하십시오.

CLI

AWS CLI

IAM 그룹에 사용자 추가

다음 `add-user-to-group` 명령은 이름이 `Admins`인 IAM 그룹에 이름이 `Bob`인 IAM 사용자를 추가합니다.

```
aws iam add-user-to-group \
    --user-name Bob \
    --group-name Admins
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 그룹에서 사용자 추가 및 제거를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 AddUserToGroup을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 명령은 **Bob**이라는 사용자를 **Admins**라는 그룹에 추가합니다.

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 AddUserToGroup을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **AttachGroupPolicy** 사용

다음 코드 예제는 `AttachGroupPolicy`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 그룹에 관리형 정책 연결

다음 `attach-group-policy` 명령은 이름이 `Finance`인 IAM 그룹에 이름이 `ReadOnlyAccess`인 AWS 관리형 정책을 연결합니다.

```
aws iam attach-group-policy \
    --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \
    --group-name Finance
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 관리형 정책과 인라인 정책을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 AttachGroupPolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **TesterPolicy**라는 고객 관리형 정책을 IAM 그룹 **Testers**에 연결합니다. 해당 그룹의 사용자는 해당 정책의 기본 버전에 정의된 권한의 영향을 즉시 받습니다.

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

예제 2: 이 예제는 **AdministratorAccess**라는 AWS 관리형 정책을 IAM 그룹 **Admins**에 연결합니다. 해당 그룹의 사용자는 해당 정책의 최신 버전에 정의된 권한의 영향을 즉시 받습니다.

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 AttachGroupPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **AttachRolePolicy** 사용

다음 코드 예제는 AttachRolePolicy의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 그룹 생성 및 사용자 추가
- 사용자 생성 및 역할 수임
- 역할 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 AttachRolePolicy를 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
```

```
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API 세부 정보는 AWS CLI 명령 참조의 AttachRolePolicy를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::attachRolePolicy(const Aws::String &roleName,
                                   const Aws::String &policyArn,
                                   const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::ListAttachedRolePoliciesRequest list_request;
    list_request.SetRoleName(roleName);

    bool done = false;
    while (!done) {
        auto list_outcome = iam.ListAttachedRolePolicies(list_request);
        if (!list_outcome.IsSuccess()) {
            std::cerr << "Failed to list attached policies of role " <<
                      roleName << ": " << list_outcome.GetError().GetMessage() <<
                      std::endl;
            return false;
        }
```

```
        const auto &policies = list_outcome.GetResult().GetAttachedPolicies();
        if (std::any_of(policies.cbegin(), policies.cend(),
                        [=](const Aws::IAM::Model::AttachedPolicy &policy) {
                                return policy.GetPolicyArn() == policyArn;
                        })) {
            std::cout << "Policy " << policyArn <<
                      " is already attached to role " << roleName << std::endl;
            return true;
        }

        done = !list_outcome.GetResult().GetIsTruncated();
        list_request.SetMarker(list_outcome.GetResult().GetMarker());
    }

    Aws::IAM::Model::AttachRolePolicyRequest request;
    request.SetRoleName(roleName);
    request.SetPolicyArn(policyArn);

    Aws::IAM::Model::AttachRolePolicyOutcome outcome =
 iam.AttachRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to attach policy " << policyArn << " to role " <<
                  roleName << ": " << outcome.GetError().GetMessage() <<
 std::endl;
    }
    else {
        std::cout << "Successfully attached policy " << policyArn << " to role "
 <<
                  roleName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 AttachRolePolicy를 참조하세요.

CLI

AWS CLI

IAM 역할에 관리형 정책 연결

다음 attach-role-policy 명령은 이름이 ReadOnlyRole인 IAM 역할에 이름이 ReadOnlyAccess인 AWS 관리형 정책을 연결합니다.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \
    --role-name ReadOnlyRole
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 관리형 정책과 인라인 정책을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 AttachRolePolicy를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(policyArn string, roleName string)
 error {
 _, err := wrapper.IamClient.AttachRolePolicy(context.TODO(),
 &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
```

```
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 AttachRolePolicy를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import
 software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AttachRolePolicy {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        attachIAMRolePolicy(iam, roleName, policyArn);
        iam.close();
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

            // Ensure that the policy is not attached to this role
            String polArn = "";
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
```

```
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
 this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);

            System.out.println("Successfully attached policy " + policyArn +
                    " to role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AttachRolePolicy를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

정책을 연결합니다.

```
import { AttachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const attachRolePolicy = (policyArn, roleName) => {
  const command = new AttachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 AttachRolePolicy를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
```

```
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        console.log(
          "AmazonDynamoDBFullAccess is already attached to this role."
        );
        process.exit();
      }
    });
    var params = {
      PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
      RoleName: process.argv[2],
    };
    iam.attachRolePolicy(params, function (err, data) {
      if (err) {
        console.log("Unable to attach policy to role", err);
      } else {
        console.log("Role attached successfully");
      }
    });
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 AttachRolePolicy를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun attachIAMRolePolicy(
```

```
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role
 $roleNameVal")
    }
}

fun checkList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
```

```
    return 0
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 AttachRolePolicy를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";
```

```
$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

    public function attachRolePolicy($roleName, $policyArn)
    {
        return $this->customWaiter(function () use ($roleName, $policyArn) {
            $this->iamClient->attachRolePolicy([
                'PolicyArn' => $policyArn,
                'RoleName' => $roleName,
            ]);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 AttachRolePolicy를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **SecurityAudit**라는 AWS 관리형 정책을 IAM 그룹 **CoSecurityAuditors**에 연결합니다. 해당 역할을 수임하는 사용자는 해당 정책의 최신 버전에 정의된 권한의 영향을 즉시 받습니다.

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 AttachRolePolicy를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

Boto3 정책 객체를 사용하여 역할에 정책을 연결합니다.

```
def attach_to_role(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).attach_role(RoleName=role_name)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise
```

Boto3 역할 객체를 사용하여 역할에 정책을 연결합니다.

```
def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 AttachRolePolicy를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 역할 정책을 나열, 생성, 연결 및 분리합니다.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
```

```
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
```

```
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 AttachRolePolicy를 참조하세요.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn attach_role_policy(
    client: &iamClient,
    role: &Role,
    policy: &Policy,
) -> Result<AttachRolePolicyOutput, SdkError<AttachRolePolicyError>> {
    client
        .attach_role_policy()
        .role_name(role.role_name())
        .policy_arn(policy.arn().unwrap_or_default())
        .send()
        .await
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 AttachRolePolicy을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public func attachRolePolicy(role: String, policyArn: String) async throws {
    let input = AttachRolePolicyInput(
        policyArn: policyArn,
        roleName: role
    )
    do {
        _ = try await client.attachRolePolicy(input: input)
    } catch {
        throw error
    }
}
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 AttachRolePolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **AttachUserPolicy** 사용

다음 코드 예제는 AttachUserPolicy의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 읽기 전용 및 읽기-쓰기 사용자 생성

CLI

AWS CLI

IAM 사용자에게 관리형 정책 연결

다음 attach-user-policy 명령은 이름이 Alice인 IAM 사용자에게 이름이 AdministratorAccess인 AWS 관리형 정책을 연결합니다.

```
aws iam attach-user-policy \
    --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
    --user-name Alice
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 관리형 정책과 인라인 정책을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 AttachUserPolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **AmazonCognitoPowerUser**라는 AWS 관리형 정책을 IAM 사용자 **Bob**에게 연결합니다. 사용자는 해당 정책의 최신 버전에 정의된 권한의 영향을 즉시 받습니다.

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 AttachUserPolicy를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 AttachUserPolicy를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  # Attaches a policy to a user
  #
  # @param user_name [String] The name of the user
  # @param policy_arn [String] The Amazon Resource Name (ARN) of the policy
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_user(user_name, policy_arn)
    @iam_client.attach_user_policy(
      user_name: user_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to user: #{e.message}")
    false
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 AttachUserPolicy를 참조하세요.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn attach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .attach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;
```

```
    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 AttachUserPolicy을 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ChangePassword** 사용

다음 코드 예제는 ChangePassword의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자의 암호 변경

IAM 사용자의 암호를 변경하려면 --cli-input-json 파라미터를 사용하여 기존 암호와 새 암호가 포함된 JSON 파일을 전달하는 것이 좋습니다. 이 방법을 사용하면 영숫자가 아닌 문자가 포함된 강력한 암호를 사용할 수 있습니다. 명령줄 파라미터로 전달할 때 영숫자가 아닌 문자가 포함된 암호는 사용하기 어려울 수 있습니다. --cli-input-json 파라미터를 사용하려면 다음 예와 같이 --generate-cli-skeleton 파라미터와 함께 change-password 명령을 사용하는 것으로 시작합니다.

```
aws iam change-password \
    --generate-cli-skeleton > change-password.json
```

이전 명령은 이전 암호와 새 암호를 입력하는 데 사용할 수 있는 change-password.json이라는 JSON 파일을 생성합니다. 예를 들어 파일은 다음과 같을 수 있습니다.

```
{
    "OldPassword": "3s0K_;xh4~8XXI",
    "NewPassword": "]35d/{pB9Fo9wJ"
}
```

다음으로 암호를 변경하려면 change-password 명령을 다시 사용하되 이번에는 --cli-input-json 파라미터를 전달하여 JSON 파일을 지정합니다. 다음 change-password 명령

은 change-password.json이라는 JSON 파일과 함께 `--cli-input-json` 파라미터를 사용합니다.

```
aws iam change-password \
    --cli-input-json file://change-password.json
```

이 명령은 출력을 생성하지 않습니다.

이 명령은 IAM 사용자만 호출할 수 있습니다. AWS 계정(루트) 자격 증명을 사용하여 이 명령을 호출하면 `InvalidUserType` 오류가 반환됩니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자가 자신의 암호를 변경하는 방법을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ChangePassword를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 명령은 명령을 실행하는 사용자의 암호를 변경합니다. 이 명령은 IAM 사용자만 호출할 수 있습니다. AWS 계정(루트) 자격 증명으로 로그인한 상태에서 이 명령을 호출하면 **`InvalidUserType`** 오류가 반환됩니다.

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ChangePassword를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **`CreateAccessKey`** 사용

다음 코드 예제는 `CreateAccessKey`의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 그룹 생성 및 사용자 추가

- 사용자 생성 및 역할 수임
- 읽기 전용 및 읽기-쓰기 사용자 생성
- 액세스 키 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateAccessKey를 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
```

```
  echo "  -u user_name   The name of the IAM user."
  echo "  [-f file_name]   Optional file name for the access key output."
  echo ""
}

# Retrieve the calling parameters.
while getopts "u:f:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    f) file_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

response=$(aws iam create-access-key \
  --user-name "$user_name" \
  --output text)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-access-key operation failed.$response"
  return 1
fi

if [[ -n "$file_name" ]]; then
  echo "$response" >"$file_name"
fi
```

```
  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}
```

- API 세부 정보는 AWS CLI 명령 참조의 CreateAccessKey를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::String AwsDoc::IAM::createAccessKey(const Aws::String &userName,
                                         const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateAccessKeyRequest request;
    request.SetUserName(userName);

    Aws::String result;
    Aws::IAM::Model::CreateAccessKeyOutcome outcome =
 iam.CreateAccessKey(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating access key for IAM user " << userName
                  << ":" << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &accessKey = outcome.GetResult().GetAccessKey();
```

```
        std::cout << "Successfully created access key for IAM user " <<
                  userName << std::endl << "  aws_access_key_id = " <<
                  accessKey.GetAccessKeyId() << std::endl <<
                  " aws_secret_access_key = " << accessKey.GetSecretAccessKey()
 <<
                  std::endl;
        result = accessKey.GetAccessKeyId();
    }

    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 CreateAccessKey를 참조하세요.

CLI

AWS CLI

IAM 사용자의 액세스 키 생성

다음 `create-access-key` 명령은 이름이 Bob인 IAM 사용자의 액세스 키(액세스 키 ID 및 비밀 액세스 키)를 생성합니다.

```
aws iam create-access-key \
    --user-name Bob
```

출력:

```
{
    "AccessKey": {
        "UserName": "Bob",
        "Status": "Active",
        "CreateDate": "2015-03-09T18:39:23.411Z",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
    }
}
```

비밀 액세스 키를 안전한 위치에 저장합니다. 손실된 경우 복구할 수 없으며, 새로운 액세스 키를 생성해야 합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateAccessKey를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(userName string)
 (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(context.TODO(),
 &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 CreateAccessKey를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <user>\s

                Where:
                   user - An AWS IAM user that you can obtain from the AWS
 Management Console.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String user = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String keyId = createIAMAccessKey(iam, user);
        System.out.println("The Key Id is " + keyId);
        iam.close();
    }

    public static String createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey().accessKeyId();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateAccessKey를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

액세스 키를 생성합니다.

```
import { CreateAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 */
export const createAccessKey = (userName) => {
  const command = new CreateAccessKeyCommand({ UserName: userName });
  return client.send(command);
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateAccessKey를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccessKey({ UserName: "IAM_USER_NAME" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKey);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateAccessKey를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createIAMAccessKey(user: String?): String {
    val request =
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateAccessKey를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 새 액세스 키와 비밀 액세스 키 페어를 생성하여 사용자 **David**에게 할당합니다. 이때만 **SecretAccessKey**를 얻을 수 있으므로 **AccessKeyId** 및 **SecretAccessKey** 값을 파일에 저장해야 합니다. 나중에 검색할 수 없습니다. 보안 키를 잃어버린 경우 새로운 액세스 키 페어를 생성해야 합니다.

```
New-IAMAccessKey -UserName David
```

출력:

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- API 세부 정보는 AWS Tools for PowerShell API 참조의 CreateAccessKey를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
```

```
        return key_pair
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 CreateAccessKey를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 액세스 키를 나열, 생성, 비활성화 및 삭제합니다.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
```

```
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded => e
    @logger.error("Error creating access key: limit exceeded. Cannot create
more.")
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: "Inactive"
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
```

```
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateAccessKey를 참조하세요.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn create_access_key(client: &iamClient, user_name: &str) ->
 Result<AccessKey, iamError> {
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<CreateAccessKeyOutput, SdkError<CreateAccessKeyError>> =
 loop {
        match client.create_access_key().user_name(user_name).send().await {
            Ok(inner_response) => {
                break Ok(inner_response);
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
```

```
                sleep(Duration::from_secs(2)).await;
            }
        }
    };

    Ok(response.unwrap().access_key.unwrap())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 CreateAccessKey을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public func createAccessKey(userName: String) async throws ->
IAMClientTypes.AccessKey {
        let input = CreateAccessKeyInput(
            userName: userName
        )
        do {
            let output = try await iamClient.createAccessKey(input: input)
            guard let accessKey = output.accessKey else {
                throw ServiceHandlerError.keyError
            }
            return accessKey
        } catch {
            throw error
        }
```

```
    }
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 CreateAccessKey를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateAccountAlias** 사용

다음 코드 예제는 CreateAccountAlias의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 계정 관리

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::createAccountAlias(const Aws::String &aliasName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::CreateAccountAliasRequest request;
    request.SetAccountAlias(aliasName);

    Aws::IAM::Model::CreateAccountAliasOutcome outcome = iam.CreateAccountAlias(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating account alias " << aliasName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
```

```
        std::cout << "Successfully created account alias " << aliasName <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 CreateAccountAlias를 참조하세요.

CLI

AWS CLI

계정 별칭 생성

다음 `create-account-alias` 명령은 AWS 계정의 별칭 `examplecorp`를 생성합니다.

```
aws iam create-account-alias \
    --account-alias examplecorp
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 AWS 계정 ID 및 별칭을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateAccountAlias를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to create (for example,
 myawsaccount).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        createIAMAccountAlias(iam, alias);
        iam.close();
        System.out.println("Done");
    }

    public static void createIAMAccountAlias(IamClient iam, String alias) {
        try {
            CreateAccountAliasRequest request =
 CreateAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();
```

```
            iam.createAccountAlias(request);
            System.out.println("Successfully created account alias: " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateAccountAlias를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

계정 별칭을 생성합니다.

```
import { CreateAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias - A unique name for the account alias.
 * @returns
 */
export const createAccountAlias = (alias) => {
  const command = new CreateAccountAliasCommand({
    AccountAlias: alias,
  });

  return client.send(command);
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateAccountAlias를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateAccountAlias를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateAccountAlias를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 AWS 계정의 계정 별칭을 **mycompanyaws**로 변경합니다. 사용자 로그온 페이지의 주소가 https://mycompanyaws.signin.aws.amazon.com/console로 변경됩니다. 별칭 대신 계정 ID 번호를 사용하는 원래 URL(<accountidnumber>https://.signin.aws.amazon.com/console)은 계속 작동합니다. 하지만 이전에 정의된 별칭 기반 URL은 모두 작동이 중지됩니다.

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- API 세부 정보는 AWS Tools for PowerShell API 참조의 CreateAccountAlias를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of
 the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 CreateAccountAlias를 참조하세요.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

계정 별칭을 나열하고, 생성하고, 삭제합니다.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateAccountAlias를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateGroup** 사용

다음 코드 예제는 CreateGroup의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 그룹 생성 및 사용자 추가

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Create an IAM group.
    /// </summary>
    /// <param name="groupName">The name to give the IAM group.</param>
    /// <returns>The IAM group that was created.</returns>
    public async Task<Group> CreateGroupAsync(string groupName)
    {
        var response = await _IAMService.CreateGroupAsync(new CreateGroupRequest
 { GroupName = groupName });
        return response.Group;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateGroup을 참조하십시오.

CLI

AWS CLI

IAM 그룹을 생성하려면

다음 `create-group` 명령은 이름이 `Admins`인 IAM 그룹을 생성합니다.

```
aws iam create-group \
    --group-name Admins
```

출력:

```
{
    "Group": {
        "Path": "/",
        "CreateDate": "2015-03-09T20:30:24.940Z",
        "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/Admins",
        "GroupName": "Admins"
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 그룹 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateGroup을 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { CreateGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} groupName
 */
export const createGroup = async (groupName) => {
  const command = new CreateGroupCommand({ GroupName: groupName });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateGroup을 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **Developers**라는 새 IAM 그룹을 생성합니다.

```
New-IAMGroup -GroupName Developers
```

출력:

```
Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
```

```
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- API 세부 정보는 AWS Tools for PowerShell API 참조의 CreateGroup을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateInstanceProfile** 사용

다음 코드 예제는 CreateInstanceProfile의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 복원력이 뛰어난 서비스 구축 및 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
```

```
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                    "\"Version\": \"2012-10-17\"," +
                                    "\"Statement\": [{" +
                                         "\"Effect\": \"Allow\"," +
                                         "\"Principal\": {" +
                                         "\"Service\": [" +
                                             "\"ec2.amazonaws.com\"" +
                                         "]" +
                                         "}," +
                                    "\"Action\": \"sts:AssumeRole\"" +
                                    "}]" +
                                "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
```

```
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
```

```
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateInstanceProfile을 참조하세요.

CLI

AWS CLI

인스턴스 프로파일 생성

다음 `create-instance-profile` 명령은 이름이 `Webserver`인 인스턴스 프로파일을 생성합니다.

```
aws iam create-instance-profile \
    --instance-profile-name Webserver
```

출력:

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AIPAJMBYC7DLSPEXAMPLE",
        "Roles": [],
        "CreateDate": "2015-03-09T20:33:19.626Z",
        "InstanceProfileName": "Webserver",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:instance-profile/Webserver"
    }
}
```

인스턴스 프로파일에 역할을 추가하려면 `add-role-to-instance-profile` 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateInstanceProfile을 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateInstanceProfile을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **ProfileForDevEC2Instance**라는 새 IAM 인스턴스 프로파일을 생성합니다. **Add-IAMRoleToInstanceProfile** 명령을 별도로 실행하여 인스턴스에 권한을 제공하는 기존 IAM 역할과 인스턴스 프로파일을 연결해야 합니다. 마지막으로 EC2 인스턴스를 시작할 때 인스턴스 프로파일을 EC2 인스턴스에 연결합니다. 이를 수행하려면 **InstanceProfile_Arn** 또는 **InstanceProfile_Name** 파라미터와 함께 **New-EC2Instance** cmdlet을 사용합니다.

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

출력:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- API 세부 정보는 AWS Tools for PowerShell API 참조의 CreateInstanceProfile을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예제에서는 정책, 역할, 인스턴스 프로파일을 생성하고 이를 모두 연결합니다.

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is
                          created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
```

```
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    def create_instance_profile(
        self, policy_file, policy_name, role_name, profile_name,
aws_managed_policies=()
    ):
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version": "2012-10-17",
            "Statement": [
                {
```

```
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        with open(policy_file) as file:
            instance_policy_doc = file.read()

        policy_arn = None
        try:
            pol_response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=instance_policy_doc
            )
            policy_arn = pol_response["Policy"]["Arn"]
            log.info("Created policy with ARN %s.", policy_arn)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                log.info("Policy %s already exists, nothing to do.", policy_name)
                list_pol_response = self.iam_client.list_policies(Scope="Local")
                for pol in list_pol_response["Policies"]:
                    if pol["PolicyName"] == policy_name:
                        policy_arn = pol["Arn"]
                        break
            if policy_arn is None:
                raise AutoScalerError(f"Couldn't create policy {policy_name}:
{err}")

        try:
            self.iam_client.create_role(
                RoleName=role_name,
AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            self.iam_client.attach_role_policy(RoleName=role_name,
PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info("Created role %s and attached policy %s.", role_name,
policy_arn)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
```

```
                log.info("Role %s already exists, nothing to do.", role_name)
            else:
                raise AutoScalerError(f"Couldn't create role {role_name}: {err}")

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            else:
                raise AutoScalerError(
                    f"Couldn't create profile {profile_name} and attach it to
role\n"
                    f"{role_name}: {err}"
                )
        return profile_arn
```

- API 세부 정보는 Python용 AWS SDK(Boto3) API 참조의 CreateSecurityGroup을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateLoginProfile** 사용

다음 코드 예제는 CreateLoginProfile의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자의 암호 생성

IAM 사용자의 암호를 생성하려면 --cli-input-json 파라미터를 사용하여 암호가 포함된 JSON 파일을 전달하는 것이 좋습니다. 이 방법을 사용하면 영숫자가 아닌 문자가 포함된 강력한 암호를 생성할 수 있습니다. 명령줄 파라미터로 전달할 때 영숫자가 아닌 문자가 포함된 암호는 만들기 어려울 수 있습니다.

--cli-input-json 파라미터를 사용하려면 다음 예와 같이 --generate-cli-skeleton 파라미터와 함께 create-login-profile 명령을 사용하는 것으로 시작합니다.

```
aws iam create-login-profile \
    --generate-cli-skeleton > create-login-profile.json
```

이전 명령은 후속 create-login-profile 명령에 대한 정보를 입력하는 데 사용할 수 있는 create-login-profile.json이라는 JSON 파일을 생성합니다. 예:

```
{
    "UserName": "Bob",
    "Password": "&1-3a6u:RA0djs",
    "PasswordResetRequired": true
}
```

다음으로 IAM 사용자의 암호를 생성하려면 create-login-profile 명령을 다시 사용하되 이번에는 --cli-input-json 파라미터를 전달하여 JSON 파일을 지정합니다. 다음 create-login-profile 명령은 create-login-profile.json이라는 JSON 파일과 함께 --cli-input-json 파라미터를 사용합니다.

```
aws iam create-login-profile \
    --cli-input-json file://create-login-profile.json
```

출력:

```
{
```

```
    "LoginProfile": {
        "UserName": "Bob",
        "CreateDate": "2015-03-10T20:55:40.274Z",
        "PasswordResetRequired": true
    }
}
```

새 암호가 계정 암호 정책을 위반하는 경우 명령은 `PasswordPolicyViolation` 오류를 반환합니다.

이미 암호가 있는 사용자의 암호를 변경하려면 `update-login-profile`을 사용합니다. 계정의 암호 정책을 설정하려면 `update-account-password-policy` 명령을 사용합니다.

계정 암호 정책에서 허용하는 경우 IAM 사용자는 `change-password` 명령을 사용하여 자신의 암호를 변경할 수 있습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 암호 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateLoginProfile을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 Bob이라는 IAM 사용자에 대한 (임시) 암호를 생성하고 다음에 **Bob**이 로그인할 때 사용자가 암호를 변경하도록 요구하는 플래그를 설정합니다.

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

출력:

```
CreateDate                PasswordResetRequired             UserName
----------                ---------------------             --------
4/14/2015 12:26:30 PM     True                              Bob
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 CreateLoginProfile을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateOpenIdConnectProvider** 사용

다음 코드 예제는 CreateOpenIdConnectProvider의 사용 방법을 보여 줍니다.

CLI

AWS CLI

OIDC(OpenID Connect) 제공업체 생성

OIDC(OpenID Connect) 제공업체를 생성하려면 --cli-input-json 파라미터를 사용하여 필수 파라미터가 포함된 JSON 파일을 전달하는 것이 좋습니다. OIDC 제공업체를 생성할 때는 제공업체의 URL을 전달해야 하며 URL은 https://로 시작해야 합니다. 일부 명령줄 환경에서는 콜론(:)과 슬래시(/) 문자가 특별한 의미를 갖기 때문에 URL을 명령줄 파라미터로 전달하기 어려울 수 있습니다. --cli-input-json 파라미터를 사용하면 이 제한을 피할 수 있습니다.

--cli-input-json 파라미터를 사용하려면 다음 예와 같이 --generate-cli-skeleton 파라미터와 함께 create-open-id-connect-provider 명령을 사용하는 것으로 시작합니다.

```
aws iam create-open-id-connect-provider \
    --generate-cli-skeleton > create-open-id-connect-provider.json
```

이전 명령은 후속 create-open-id-connect-provider 명령에 대한 정보를 입력하는 데 사용할 수 있는 create-open-id-connect-provider.json이라는 JSON 파일을 생성합니다. 예:

```
{
    "Url": "https://server.example.com",
    "ClientIDList": [
        "example-application-ID"
    ],
    "ThumbprintList": [
        "c3768084dfb3d2b68b7897bf5f565da8eEXAMPLE"
    ]
}
```

다음으로, OIDC(OpenID Connect) 제공업체를 생성하려면 create-open-id-connect-provider 명령을 다시 사용하되 이번에는 --cli-input-json 파라미터를 전달하여 JSON 파일을 지정합니다. 다음 create-open-id-connect-provider 명령은 create-open-id-connect-provider.json이라는 JSON 파일과 함께 --cli-input-json 파라미터를 사용합니다.

```
aws iam create-open-id-connect-provider \
    --cli-input-json file://create-open-id-connect-provider.json
```

출력:

```
{
    "OpenIDConnectProviderArn": "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
}
```

OIDC 제공업체에 대한 자세한 내용은 AWS IAM 사용 설명서의 IAM에서 OIDC(OpenID Connect) ID 제공업체 생성을 참조하세요.

OIDC 제공업체의 지문 가져오기에 대한 자세한 내용은 AWS IAM 사용 설명서의 OpenID Connect 자격 증명 제공업체의 지문 얻기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateOpenIdConnectProvider를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 URL **https://example.oidcprovider.com** 및 클라이언트 ID **my-testapp-1**에 있는 OIDC 호환 제공업체 서비스와 연결된 IAM OIDC 제공업체를 생성합니다. OIDC 제공업체가 지문을 제공합니다. 지문을 인증하려면 http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html의 단계를 따르세요.

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList
 my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

출력:

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 CreateOpenIdConnectProvider를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreatePolicy** 사용

다음 코드 예제는 CreatePolicy의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 그룹 생성 및 사용자 추가
- 사용자 생성 및 역할 수임
- 읽기 전용 및 읽기-쓰기 사용자 생성
- 정책 관리
- IAM 정책 빌더 API 작업

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });
```

```
        return response.Policy;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreatePolicy를 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
  echo "function iam_create_policy"
  echo "Creates an AWS Identity and Access Management (IAM) policy."
  echo "  -n policy_name   The name of the IAM policy."
  echo "  -p policy_json -- The policy document."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) policy_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$policy_name" ]]; then
  errecho "ERROR: You must provide a policy name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-policy \
  --policy-name "$policy_name" \
  --policy-document "$policy_document" \
  --output text \
```

```
    --query Policy.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"
}
```

- API 세부 정보는 AWS CLI 명령 참조의 CreatePolicy를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::String AwsDoc::IAM::createPolicy(const Aws::String &policyName,
                                      const Aws::String &rsrcArn,
                                      const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreatePolicyRequest request;
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(BuildSamplePolicyDocument(rsrcArn));

    Aws::IAM::Model::CreatePolicyOutcome outcome = iam.CreatePolicy(request);
    Aws::String result;
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy " << policyName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    else {
        result = outcome.GetResult().GetPolicy().GetArn();
        std::cout << "Successfully created policy " << policyName <<
                  std::endl;
    }

    return result;
}

Aws::String AwsDoc::IAM::BuildSamplePolicyDocument(const Aws::String &rsrc_arn) {
    std::stringstream stringStream;
    stringStream << "{"
                 << "  \"Version\": \"2012-10-17\","
                 << "  \"Statement\": ["
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": \"logs:CreateLogGroup\","
                 << "        \"Resource\": \""
                 << rsrc_arn
                 << "\""
                 << "    },"
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": ["
                 << "            \"dynamodb:DeleteItem\","
                 << "            \"dynamodb:GetItem\","
                 << "            \"dynamodb:PutItem\","
                 << "            \"dynamodb:Scan\","
                 << "            \"dynamodb:UpdateItem\""
                 << "       ],"
                 << "       \"Resource\": \""
                 << rsrc_arn
                 << "\""
                 << "    }"
                 << "   ]"
                 << "}";

    return stringStream.str();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 CreatePolicy를 참조하세요.

CLI

AWS CLI

예제 1: 고객 관리형 정책 생성

다음 명령은 이름이 my-policy인 고객 관리형 정책을 생성합니다.

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy
```

policy 파일은 이름이 my-bucket인 Amazon S3 버킷의 shared 폴더에 대한 읽기 전용 액세스 권한을 부여하는 현재 폴더의 JSON 문서입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/shared/*"
            ]
        }
    ]
}
```

출력:

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "CreateDate": "2015-06-01T19:31:18.620Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",
        "DefaultVersionId": "v1",
        "Path": "/",
```

```
        "Arn": "arn:aws:iam::0123456789012:policy/my-policy",
        "UpdateDate": "2015-06-01T19:31:18.620Z"
    }
}
```

문자열 파라미터의 입력으로 파일 사용에 대한 자세한 내용은 AWS CLI 사용 설명서의 AWS CLI에 파라미터 값 지정을 참조하세요.

예제 2: 설명이 포함된 고객 관리형 정책 생성

다음 명령은 변경 불가능한 설명이 포함된 이름이 `my-policy`인 고객 관리형 정책을 생성합니다.

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy.json \
    --description "This policy grants access to all Put, Get, and List actions
 for my-bucket"
```

`policy.json` 파일은 이름이 `my-bucket`인 Amazon S3 버킷의 모든 Put, List, Get 작업에 대한 액세스 권한을 부여하는 현재 폴더의 JSON 문서입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                 "s3:ListBucket*",
                 "s3:PutBucket*",
                 "s3:GetBucket*"
             ],
             "Resource": [
                 "arn:aws:s3:::my-bucket"
             ]
         }
     ]
 }
```

출력:

```
{
```

```
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "ANPAWGSUGIDPEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2023-05-24T22:38:47+00:00",
        "UpdateDate": "2023-05-24T22:38:47+00:00"
    }
}
```

자격 증명 기반 정책에 대한 자세한 내용은 AWS IAM 사용 설명서의 자격 증명 기반 정책 및 리소스 기반 정책을 참조하세요.

예제 3: 태그가 포함된 고객 관리형 정책 생성

다음 명령은 태그가 포함된 이름이 `my-policy`인 고객 관리형 정책을 생성합니다. 이 예제에서는 다음 JSON 형식의 태그(`'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'`)에 `--tags` 파라미터 플래그를 사용합니다. 또는 `--tags` 플래그를 짧은 형식의 태그(`'Key=Department,Value=Accounting Key=Location,Value=Seattle'`)에 사용할 수도 있습니다.

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy.json \
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

`policy.json` 파일은 이름이 `my-bucket`인 Amazon S3 버킷의 모든 Put, List, Get 작업에 대한 액세스 권한을 부여하는 현재 폴더의 JSON 문서입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                 "s3:ListBucket*",
```

```
                "s3:PutBucket*",
                "s3:GetBucket*"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket"
            ]
        }
    ]
}
```

출력:

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "ANPAWGSUGIDPEXAMPLE",
        "Arn": "arn:aws:iam::12345678012:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2023-05-24T23:16:39+00:00",
        "UpdateDate": "2023-05-24T23:16:39+00:00",
        "Tags": [
            {
                "Key": "Department",
                "Value": "Accounting"
            },
                "Key": "Location",
                "Value": "Seattle"
            {

        ]
    }
}
```

정책 태그 지정에 대한 자세한 내용은 AWS IAM 사용 설명서의 고객 관리형 정책 태깅을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreatePolicy를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(policyName string, actions []string,
  resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Action: actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
```

```
 }
 result, err := wrapper.IamClient.CreatePolicy(context.TODO(),
 &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 CreatePolicy를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreatePolicy {

    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"dynamodb:DeleteItem\"," +
            "            \"dynamodb:GetItem\"," +
            "            \"dynamodb:PutItem\"," +
            "            \"dynamodb:Scan\"," +
            "            \"dynamodb:UpdateItem\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static void main(String[] args) {

        final String usage = """
                Usage:
                    CreatePolicy <policyName>\s

                Where:
                    policyName - A unique policy name.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
```

```
        String result = createIAMPolicy(iam, policyName);
        System.out.println("Successfully created a policy with this ARN value: "
 + result);
        iam.close();
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument)
                    .build();

            CreatePolicyResponse response = iam.createPolicy(request);

            // Wait until the policy is created.
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
 iamWaiter.waitUntilPolicyExists(polRequest);

 waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreatePolicy를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

정책을 생성합니다.

```
import { CreatePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyName
 */
export const createPolicy = (policyName) => {
  const command = new CreatePolicyCommand({
    PolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: "*",
          Resource: "*",
        },
      ],
    }),
    PolicyName: policyName,
  });

  return client.send(command);
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreatePolicy를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var myManagedPolicy = {
  Version: "2012-10-17",
  Statement: [
    {
      Effect: "Allow",
      Action: "logs:CreateLogGroup",
      Resource: "RESOURCE_ARN",
    },
    {
      Effect: "Allow",
      Action: [
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Scan",
        "dynamodb:UpdateItem",
      ],
      Resource: "RESOURCE_ARN",
    },
  ],
};

var params = {
  PolicyDocument: JSON.stringify(myManagedPolicy),
  PolicyName: "myDynamoDBPolicy",
};
```

```
iam.createPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreatePolicy를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createIAMPolicy(policyNameVal: String?): String {
    val policyDocumentVal =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"dynamodb:DeleteItem\"," +
            "            \"dynamodb:GetItem\"," +
            "            \"dynamodb:PutItem\"," +
            "            \"dynamodb:Scan\"," +
            "            \"dynamodb:UpdateItem\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}"
```

```
    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreatePolicy를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

$listAllBucketsPolicyDocument = "{
               \"Version\": \"2012-10-17\",
               \"Statement\": [{
                   \"Effect\": \"Allow\",
                   \"Action\": \"s3:ListAllMyBuckets\",
                   \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

    public function createPolicy(string $policyName, string $policyDocument)
    {
```

```
        $result = $this->customWaiter(function () use ($policyName,
 $policyDocument) {
            return $this->iamClient->createPolicy([
                'PolicyName' => $policyName,
                'PolicyDocument' => $policyDocument,
            ]);
        });
        return $result['Policy'];
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreatePolicy를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **MySamplePolicy**라는 현재 AWS 계정에 새 IAM 정책을 생성합니다. **MySamplePolicy.json** 파일은 정책 콘텐츠를 제공합니다. JSON 정책 파일을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다.

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

출력:

```
Arn              : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 4/14/2015 2:45:59 PM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : LD4KP6HVFE7WGEXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 4/14/2015 2:45:59 PM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 CreatePolicy를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::my-bucket/*' to allow actions on all
 objects
                         in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
```

```
        return policy
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 CreatePolicy를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 역할 정책을 나열, 생성, 연결 및 분리합니다.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
```

```
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
```

```
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreatePolicy를 참조하세요.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn create_policy(
    client: &iamClient,
    policy_name: &str,
    policy_document: &str,
) -> Result<Policy, iamError> {
```

```
    let policy = client
        .create_policy()
        .policy_name(policy_name)
        .policy_document(policy_document)
        .send()
        .await?;
    Ok(policy.policy.unwrap())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 CreatePolicy을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public func createPolicy(name: String, policyDocument: String) async throws -
> IAMClientTypes.Policy {
        let input = CreatePolicyInput(
            policyDocument: policyDocument,
            policyName: name
        )
        do {
            let output = try await iamClient.createPolicy(input: input)
            guard let policy = output.policy else {
                throw ServiceHandlerError.noSuchPolicy
            }
            return policy
```

```
        } catch {
            throw error
        }
    }
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 CreatePolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreatePolicyVersion** 사용

다음 코드 예제는 CreatePolicyVersion의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 정책 관리

CLI

AWS CLI

관리형 정책의 새 버전 생성

이 예제는 ARN이 arn:aws:iam::123456789012:policy/MyPolicy인 IAM 정책의 새 v2 버전을 생성하고 이를 기본 버전으로 만듭니다.

```
aws iam create-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --policy-document file://NewPolicyVersion.json \
    --set-as-default
```

출력:

```
{
    "PolicyVersion": {
        "CreateDate": "2015-06-16T18:56:03.721Z",
        "VersionId": "v2",
        "IsDefaultVersion": true
```

```
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 정책 버전 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreatePolicyVersion을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/MyPolicy**인 IAM 정책의 새 'v2' 버전을 생성하고 이를 기본 버전으로 만듭니다. **NewPolicyVersion.json** 파일은 정책 콘텐츠를 제공합니다. JSON 정책 파일을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다.

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

출력:

```
CreateDate                          Document                 IsDefaultVersion
          VersionId
----------                          --------                 ----------------
          ---------
4/15/2015 10:54:54 AM                                        True
          v2
```

- Cmdlet 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 CreatePolicyVersion을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 CreatePolicyVersion를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateRole** 사용

다음 코드 예제는 CreateRole의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 그룹 생성 및 사용자 추가
- 사용자 생성 및 역할 수임
- 역할 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateRole을 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
#############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
  echo "function iam_create_user_access_key"
  echo "Creates an AWS Identity and Access Management (IAM) role."
  echo "  -n role_name   The name of the IAM role."
  echo "  -p policy_json -- The assume role policy document."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-role \
  --role-name "$role_name" \
  --assume-role-policy-document "$policy_document" \
  --output text \
  --query Role.Arn)
```

```
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API 세부 정보는 AWS CLI 명령 참조의 CreateRole을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::createIamRole(
        const Aws::String &roleName,
        const Aws::String &policy,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::CreateRoleRequest request;

    request.SetRoleName(roleName);
    request.SetAssumeRolePolicyDocument(policy);

    Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    else {
        const Aws::IAM::Model::Role iamRole = outcome.GetResult().GetRole();
        std::cout << "Created role " << iamRole.GetRoleName() << "\n";
        std::cout << "ID: " << iamRole.GetRoleId() << "\n";
        std::cout << "ARN: " << iamRole.GetArn() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 CreateRole을 참조하십시오.

CLI

AWS CLI

예제 1: IAM 역할 생성

다음 `create-role` 명령은 이름이 `Test-Role`인 역할을 생성하고 해당 역할에 신뢰 정책을 연결합니다.

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json
```

출력:

```
{
    "Role": {
        "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "CreateDate": "2013-06-07T20:43:32.821Z",
        "RoleName": "Test-Role",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/Test-Role"
    }
}
```

신뢰 정책은 Test-Role-Trust-Policy.json 파일에 JSON 문서로 정의됩니다. (파일 이름과 확장자는 중요하지 않습니다.) 신뢰 정책에서 보안 주체를 지정해야 합니다.

역할에 권한 정책을 연결하려면 put-role-policy 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 역할 생성을 참조하세요.

예제 2: 지정된 최대 세션 시간을 포함한 IAM 역할 생성

다음 create-role 명령은 이름이 Test-Role인 역할을 생성하고 최대 세션 지속 시간을 7,200초(2시간)로 설정합니다.

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json \
    --max-session-duration 7200
```

출력:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "Test-Role",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:role/Test-Role",
        "CreateDate": "2023-05-24T23:50:25+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::12345678012:root"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 역할 최대 세션 기간 수정(AWS API)을 참조하세요.

예제 3: 태그가 포함된 IAM 역할 생성

다음 명령은 태그가 포함된 IAM 역할 Test-Role을 생성합니다. 이 예제에서는 다음 JSON 형식의 태그('{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}')에 --tags 파라미터 플래그를 사용합니다. 또는 --tags 플래그를 짧은 형식의 태그('Key=Department,Value=Accounting Key=Location,Value=Seattle')에 사용할 수도 있습니다.

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json \
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

출력:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "Test-Role",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:role/Test-Role",
        "CreateDate": "2023-05-25T23:29:41+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::123456789012:root"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Tags": [
            {
                "Key": "Department",
                "Value": "Accounting"
            },
            {
                "Key": "Location",
                "Value": "Seattle"
```

```
            }
        ]
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 역할 태그 지정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateRole을 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(roleName string, trustedUserArn string)
 (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
```

```
   Principal: map[string]string{"AWS": trustedUserArn},
   Action: []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreateRole(context.TODO(),
 &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 CreateRole을 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;

/*
*   This example requires a trust policy document. For more information, see:
*   https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-
roles/
*
*
*  In addition, set up your development environment, including your credentials.
*
*  For information, see this documentation topic:
*
*  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = """
                Usage:
                    <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
                    fileLocation - The location of the JSON document that
 represents the trust policy.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String rolename = args[0];
        String fileLocation = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMRole(iam, rolename, fileLocation);
        System.out.println("Successfully created user: " + result);
```

```
        iam.close();
    }

    public static String createIAMRole(IamClient iam, String rolename, String
 fileLocation) throws Exception {
        try {
            JSONObject jsonObject = (JSONObject)
 readJsonSimpleDemo(fileLocation);
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(jsonObject.toJSONString())
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " +
 response.role().arn());

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static Object readJsonSimpleDemo(String filename) throws Exception {
        FileReader reader = new FileReader(filename);
        JSONParser jsonParser = new JSONParser();
        return jsonParser.parse(reader);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateRole을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

역할을 생성합니다.

```
import { CreateRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const createRole = (roleName) => {
  const command = new CreateRoleCommand({
    AssumeRolePolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
            Service: "lambda.amazonaws.com",
          },
          Action: "sts:AssumeRole",
        },
      ],
    }),
    RoleName: roleName,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateRole을 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

    /**
     * @param string $roleName
     * @param string $rolePolicyDocument
     * @return array
     * @throws AwsException
     */
    public function createRole(string $roleName, string $rolePolicyDocument)
    {
        $result = $this->customWaiter(function () use ($roleName,
 $rolePolicyDocument) {
            return $this->iamClient->createRole([
                'AssumeRolePolicyDocument' => $rolePolicyDocument,
                'RoleName' => $roleName,
            ]);
        });
        return $result['Role'];
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateRole을 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **MyNewRole**이라는 새 역할을 생성하고 여기에 **NewRoleTrustPolicy.json** 파일에 있는 정책을 연결합니다. JSON 정책 파일을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다. 출력에 표시된 정책 문서는 URL로 인코딩됩니다. 이 예제에서는 **UrlDecode** .NET 메서드를 사용하여 디코딩됩니다.

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

출력:

```
Arn                       : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C
%0D%0A%20%20%22Statement%22
                           %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
                           %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow
%22%2C%0D%0A%20%20%20%20%20%20
                           %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                           %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
                           %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D
%0A%20%20%20%20%7D%0D%0A%20
                           %20%5D%0D%0A%7D
CreateDate                : 4/15/2015 11:04:23 AM
Path                      : /
RoleId                    : V5PAJI2KPN4EAEXAMPLE1
RoleName                  : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 CreateRole을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
```

```
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 CreateRole를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
 # Creates a role and attaches policies to it.
 #
 # @param role_name [String] The name of the role.
 # @param assume_role_policy_document [Hash] The trust relationship policy
document.
 # @param policy_arns [Array<String>] The ARNs of the policies to attach.
 # @return [String, nil] The ARN of the new role if successful, or nil if an
error occurred.
 def create_role(role_name, assume_role_policy_document, policy_arns)
   response = @iam_client.create_role(
     role_name: role_name,
     assume_role_policy_document: assume_role_policy_document.to_json
```

```
    )
    role_arn = response.role.arn

    policy_arns.each do |policy_arn|
      @iam_client.attach_role_policy(
        role_name: role_name,
        policy_arn: policy_arn
      )
    end

    role_arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating role: #{e.message}")
    nil
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateRole을 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn create_role(
    client: &iamClient,
    role_name: &str,
    role_policy_document: &str,
) -> Result<Role, iamError> {
    let response: CreateRoleOutput = loop {
        if let Ok(response) = client
            .create_role()
            .role_name(role_name)
            .assume_role_policy_document(role_policy_document)
            .send()
            .await
        {
```

```
            break response;
        }
    };

    Ok(response.role.unwrap())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 CreateRole을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public func createRole(name: String, policyDocument: String) async throws ->
String {
        let input = CreateRoleInput(
            assumeRolePolicyDocument: policyDocument,
            roleName: name
        )
        do {
            let output = try await client.createRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            guard let id = role.roleId else {
                throw ServiceHandlerError.noSuchRole
            }
```

```
            return id
        } catch {
            throw error
        }
    }
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 CreateRole을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateSAMLProvider** 사용

다음 코드 예제는 CreateSAMLProvider의 사용 방법을 보여 줍니다.

CLI

AWS CLI

SAML 공급자 생성

이 예제는 IAM에 이름이 MySAMLProvider인 새 SAML 공급자를 생성합니다. SAMLMetaData.xml 파일에 있는 SAML 메타데이터 문서에 설명되어 있습니다.

```
aws iam create-saml-provider \
    --saml-metadata-document file://SAMLMetaData.xml \
    --name MySAMLProvider
```

출력:

```
{
    "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/MySAMLProvider"
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM SAML 자격 증명 공급자 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateSAMLProvider를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { CreateSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "fs";
import * as path from "path";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";

const client = new IAMClient({});

/**
 * This sample document was generated using Auth0.
 * For more information on generating this document,
 see https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_providers_create_saml.html#samlstep1.
 */
const sampleMetadataDocument = readFileSync(
  path.join(
    dirnameFromMetaUrl(import.meta.url),
    "../../../../resources/sample_files/sample_saml_metadata.xml",
  ),
);

/**
 *
 * @param {*} providerName
 * @returns
 */
export const createSAMLProvider = async (providerName) => {
  const command = new CreateSAMLProviderCommand({
    Name: providerName,
    SAMLMetadataDocument: sampleMetadataDocument.toString(),
  });

  const response = await client.send(command);
```

```
  console.log(response);
  return response;
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateSAMLProvider를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM에 새 SAML 제공업체를 생성합니다. 이름은 **MySAMLProvider**이며 SAML 서비스 제공업체의 웹 사이트에서 별도로 다운로드한 **SAMLMetaData.xml** 파일에 있는 SAML 메타데이터 문서에 설명되어 있습니다.

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

출력:

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 CreateSAMLProvider를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateServiceLinkedRole** 사용

다음 코드 예제는 `CreateServiceLinkedRole`의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateServiceLinkedRole을 참조하십시오.

CLI

AWS CLI

서비스 연결 역할 생성

다음 `create-service-linked-role` 예제에서는 지정된 AWS 서비스에 대한 서비스 연결 역할을 생성하고 지정된 설명을 첨부합니다.

```
aws iam create-service-linked-role \
    --aws-service-name lex.amazonaws.com \
    --description "My service-linked role to support Lex"
```

출력:

```
{
    "Role": {
        "Path": "/aws-service-role/lex.amazonaws.com/",
```

```
        "RoleName": "AWSServiceRoleForLexBots",
        "RoleId": "AROA1234567890EXAMPLE",
        "Arn": "arn:aws:iam::1234567890:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
        "CreateDate": "2019-04-17T20:34:14+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": [
                        "sts:AssumeRole"
                    ],
                    "Effect": "Allow",
                    "Principal": {
                        "Service": [
                            "lex.amazonaws.com"
                        ]
                    }
                }
            ]
        }
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 서비스 연결 역할 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateServiceLinkedRole을 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
```

```
type RoleWrapper struct {
 IamClient *iam.Client
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(serviceName string,
 description string) (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(context.TODO(),
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 CreateServiceLinkedRole을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

서비스 연결 역할을 생성합니다.

```
import {
```

```
  CreateServiceLinkedRoleCommand,
  GetRoleCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} serviceName
 */
export const createServiceLinkedRole = async (serviceName) => {
  const command = new CreateServiceLinkedRoleCommand({
    // For a list of AWS services that support service-linked roles,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-
services-that-work-with-iam.html.
    //
    // For a list of AWS service endpoints, see https://docs.aws.amazon.com/
general/latest/gr/aws-service-information.html.
    AWSServiceName: serviceName,
  });
  try {
    const response = await client.send(command);
    console.log(response);
    return response;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInputException" &&
      caught.message.includes(
        "Service role name AWSServiceRoleForElasticBeanstalk has been taken in
 this account",
      )
    ) {
      console.warn(caught.message);
      return client.send(
        new GetRoleCommand({ RoleName: "AWSServiceRoleForElasticBeanstalk" }),
      );
    } else {
      throw caught;
    }
  }
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateServiceLinkedRole을 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
 $description = "")
    {
        $createServiceLinkedRoleArguments = ['AWSServiceName' =>
 $awsServiceName];
        if ($customSuffix) {
            $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
        }
        if ($description) {
            $createServiceLinkedRoleArguments['Description'] = $description;
        }
        return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateServiceLinkedRole을 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 AutoScaling 서비스에 대한 서비스 연결 역할을 생성합니다.

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support
 autoscaling"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 CreateServiceLinkedRole을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def create_service_linked_role(service_name, description):
    """
    Creates a service-linked role.

    :param service_name: The name of the service that owns the role.
    :param description: A description to give the role.
    :return: The newly created role.
    """
    try:
        response = iam.meta.client.create_service_linked_role(
            AWSServiceName=service_name, Description=description
        )
        role = iam.Role(response["Role"]["RoleName"])
        logger.info("Created service-linked role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create service-linked role for %s.",
 service_name)
        raise
```

```
    else:
        return role
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 CreateServiceLinkedRole를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
 # Creates a service-linked role
 #
 # @param service_name [String] The service name to create the role for.
 # @param description [String] The description of the service-linked role.
 # @param suffix [String] Suffix for customizing role name.
 # @return [String] The name of the created role
 def create_service_linked_role(service_name, description, suffix)
   response = @iam_client.create_service_linked_role(
     aws_service_name: service_name, description: description, custom_suffix:
suffix,)
   role_name = response.role.role_name
   @logger.info("Created service-linked role #{role_name}.")
   role_name
 rescue Aws::Errors::ServiceError => e
   @logger.error("Couldn't create service-linked role for #{service_name}.
Here's why:")
   @logger.error("\t#{e.code}: #{e.message}")
   raise
 end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateServiceLinkedRole을 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn create_service_linked_role(
    client: &iamClient,
    aws_service_name: String,
    custom_suffix: Option<String>,
    description: Option<String>,
) -> Result<CreateServiceLinkedRoleOutput,
 SdkError<CreateServiceLinkedRoleError>> {
    let response = client
        .create_service_linked_role()
        .aws_service_name(aws_service_name)
        .set_custom_suffix(custom_suffix)
        .set_description(description)
        .send()
        .await?;

    Ok(response)
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 CreateServiceLinkedRole을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public func createServiceLinkedRole(service: String, suffix: String? = nil,
description: String?)
                   async throws -> IAMClientTypes.Role {
       let input = CreateServiceLinkedRoleInput(
           awsServiceName: service,
           customSuffix: suffix,
           description: description
       )
       do {
           let output = try await client.createServiceLinkedRole(input: input)
           guard let role = output.role else {
               throw ServiceHandlerError.noSuchRole
           }
           return role
       } catch {
           throw error
       }
   }
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 CreateServiceLinkedRole을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateUser** 사용

다음 코드 예제는 CreateUser의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 그룹 생성 및 사용자 추가
- 사용자 생성 및 역할 수임

- 읽기 전용 및 읽기-쓰기 사용자 생성

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
        return response.User;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 CreateUser를 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
##############################################################################
```

```
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an WS Identity and Access Management (IAM) user. You must
 supply a username:"
```

```
    echo "  -u user_name    The name of the user. It must be unique within the
account."
    echo ""
 }

 # Retrieve the calling parameters.
 while getopts "u:h" option; do
   case "${option}" in
     u) user_name="${OPTARG}" ;;
     h)
       usage
       return 0
       ;;
     \?)
       echo "Invalid parameter"
       usage
       return 1
       ;;
   esac
 done
 export OPTIND=1

 if [[ -z "$user_name" ]]; then
   errecho "ERROR: You must provide a username with the -u parameter."
   usage
   return 1
 fi

 iecho "Parameters:\n"
 iecho "    User name:   $user_name"
 iecho ""

 # If the user already exists, we don't want to try to create it.
 if (iam_user_exists "$user_name"); then
   errecho "ERROR: A user with that name already exists in the account."
   return 1
 fi

 response=$(aws iam create-user --user-name "$user_name" \
   --output text \
   --query 'User.Arn')

 local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API 세부 정보는 AWS CLI 명령 참조의 CreateUser를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateUserRequest create_request;
    create_request.SetUserName(userName);

    auto create_outcome = iam.CreateUser(create_request);
    if (!create_outcome.IsSuccess()) {
        std::cerr << "Error creating IAM user " << userName << ":" <<
                  create_outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created IAM user " << userName << std::endl;
    }

    return create_outcome.IsSuccess();
```

- API 세부 정보는 AWS SDK for C++ API 참조의 CreateUser를 참조하십시오.

CLI

AWS CLI

예제 1: IAM 사용자 생성

다음 `create-user` 명령은 현재 계정에서 이름이 Bob인 IAM 사용자를 생성합니다.

```
aws iam create-user \
    --user-name Bob
```

출력:

```
{
    "User": {
        "UserName": "Bob",
        "Path": "/",
        "CreateDate": "2023-06-08T03:20:41.270Z",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:user/Bob"
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하세요.

예제 2: 지정된 경로에 IAM 사용자 생성

다음 `create-user` 명령은 지정된 경로에서 이름이 Bob인 IAM 사용자를 생성합니다.

```
aws iam create-user \
    --user-name Bob \
    --path /division_abc/subdivision_xyz/
```

출력:

```
{
    "User": {
        "Path": "/division_abc/subdivision_xyz/",
```

```
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/division_abc/subdivision_xyz/Bob",
        "CreateDate": "2023-05-24T18:20:17+00:00"
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 식별자를 참조하세요.

예제 3: 태그가 포함된 IAM 사용자 생성

다음 `create-user` 명령은 태그가 포함된 이름이 Bob인 IAM 사용자를 생성합니다. 이 예제에서는 다음 JSON 형식의 태그(`'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'`)에 `--tags` 파라미터 플래그를 사용합니다. 또는 `--tags` 플래그를 짧은 형식의 태그(`'Key=Department,Value=Accounting Key=Location,Value=Seattle'`)에 사용할 수도 있습니다.

```
aws iam create-user \
    --user-name Bob \
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

출력:

```
{
    "User": {
        "Path": "/",
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/Bob",
        "CreateDate": "2023-05-25T17:14:21+00:00",
        "Tags": [
            {
                "Key": "Department",
                "Value": "Accounting"
            },
            {
                "Key": "Location",
                "Value": "Seattle"
            }
        ]
    }
```

```
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 태깅을 참조하세요.

예제 3: 권한 경계가 설정된 IAM 사용자 생성

다음 `create-user` 명령은 Amazons3FullAccess의 권한 경계가 포함되어 있으며 이름이 Bob인 IAM 사용자를 생성합니다.

```
aws iam create-user \
    --user-name Bob \
    --permissions-boundary arn:aws:iam::aws:policy/AmazonS3FullAccess
```

출력:

```
{
    "User": {
        "Path": "/",
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/Bob",
        "CreateDate": "2023-05-24T17:50:53+00:00",
        "PermissionsBoundary": {
        "PermissionsBoundaryType": "Policy",
        "PermissionsBoundaryArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
        }
    }
}
```

자세한 정보는 AWS IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateUser를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(userName string) (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(context.TODO(),
 &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 CreateUser를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
```

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username>\s

                Where:
                    username - The name of the user to create.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMUser(iam, username);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMUser(IamClient iam, String username) {
```

```
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            CreateUserResponse response = iam.createUser(request);

            // Wait until the user is created.
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
 iamWaiter.waitUntilUserExists(userRequest);

 waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user().userName();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateUser를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

사용자를 생성합니다.

```
import { CreateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const createUser = (name) => {
  const command = new CreateUserCommand({ UserName: name });
  return client.send(command);
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateUser를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    iam.createUser(params, function (err, data) {
```

```
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  } else {
    console.log(
      "User " + process.argv[2] + " already exists",
      data.User.UserId
    );
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 CreateUser를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 CreateUser를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";


    /**
     * @param string $name
     * @return array
     * @throws AwsException
     */
    public function createUser(string $name): array
    {
        $result = $this->iamClient->createUser([
            'UserName' => $name,
        ]);

        return $result['User'];
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 CreateUser를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **Bob**이라는 IAM 사용자를 생성합니다. Bob이 AWS 콘솔에 로그인해야 하는 경우 **New-IAMLoginProfile** 명령을 별도로 실행하여 암호가 포함된 로그인 프로파일을 생성해야 합니다. Bob이 AWS PowerShell 또는 크로스 플랫폼 CLI 명령을 실행하거나 AWS API 직접 호출을 수행해야 하는 경우 별도로 **New-IAMAccessKey** 명령을 실행하여 액세스 키를 생성해야 합니다.

```
New-IAMUser -UserName Bob
```

출력:

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 4/22/2015 12:02:11 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : AIDAJWGEFDMEMEXAMPLE1
UserName         : Bob
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 CreateUser를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
```

```
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 CreateUser를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
 # Creates a user and their login profile
 #
 # @param user_name [String] The name of the user
 # @param initial_password [String] The initial password for the user
 # @return [String, nil] The ID of the user if created, or nil if an error
occurred
 def create_user(user_name, initial_password)
   response = @iam_client.create_user(user_name: user_name)
   @iam_client.wait_until(:user_exists, user_name: user_name)
   @iam_client.create_login_profile(
     user_name: user_name,
     password: initial_password,
     password_reset_required: true
   )
   @logger.info("User '#{user_name}' created successfully.")
   response.user.user_id
 rescue Aws::IAM::Errors::EntityAlreadyExists
```

```
    @logger.error("Error creating user '#{user_name}': user already exists.")
    nil
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating user '#{user_name}': #{e.message}")
    nil
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateUser를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn create_user(client: &iamClient, user_name: &str) -> Result<User,
 iamError> {
    let response = client.create_user().user_name(user_name).send().await?;

    Ok(response.user.unwrap())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 CreateUser을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public func createUser(name: String) async throws -> String {
    let input = CreateUserInput(
        userName: name
    )
    do {
        let output = try await client.createUser(input: input)
        guard let user = output.user else {
            throw ServiceHandlerError.noSuchUser
        }
        guard let id = user.userId else {
            throw ServiceHandlerError.noSuchUser
        }
        return id
    } catch {
        throw error
    }
}
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 CreateUser를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **CreateVirtualMfaDevice** 사용

다음 코드 예제는 `CreateVirtualMfaDevice`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

가상 MFA 디바이스 생성

이 예제는 BobsMFADevice라는 새 가상 MFA 디바이스를 생성합니다. 부트스트랩 정보가 포함된 QRCode.png라는 파일을 생성하여 C:/ 디렉터리에 배치합니다. 이 예제에서 사용된 부트스트랩 방법은 QRCodePNG입니다.

```
aws iam create-virtual-mfa-device \
    --virtual-mfa-device-name BobsMFADevice \
    --outfile C:/QRCode.png \
    --bootstrap-method QRCodePNG
```

출력:

```
{
    "VirtualMFADevice": {
        "SerialNumber": "arn:aws:iam::210987654321:mfa/BobsMFADevice"
}
```

자세한 내용은 AWS IAM 사용 설명서의 AWS에서 멀티 팩터 인증(MFA) 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 CreateVirtualMfaDevice를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 새 가상 MFA 디바이스를 생성합니다. 줄 2와 3은 가상 MFA 소프트웨어 프로그램에서 계정을 생성하는 데 필요한 **Base32StringSeed** 값을 QR 코드의 대안으로 추출합니다. 값으로 프로그램을 구성한 후 프로그램에서 두 개의 순차적 인증 코드를 받습니다. 끝으로 마지막 명령을 사용하여 가상 MFA 디바이스를 IAM 사용자 **Bob**에게 연결하고 계정을 두 개의 인증 코드와 동기화합니다.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

출력:

```
-- Pause here to enter base-32 string seed code into virtual MFA program to
 register account. --
```

```
Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

예제 2: 이 예제는 새 가상 MFA 디바이스를 생성합니다. 줄 2와 3은 **QRCodePNG** 값을 추출하여 파일에 씁니다. Base32StringSeed 값을 수동으로 입력하는 대신 가상 MFA 소프트웨어 프로그램에서 이 이미지를 스캔하여 계정을 생성할 수 있습니다. 가상 MFA 프로그램에서 계정을 생성한 후 두 개의 순차적 인증 코드를 받아 마지막 명령에 입력하여 가상 MFA 디바이스를 IAM 사용자 **Bob**에게 연결하고 계정을 동기화합니다.

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path
 QRCode.png
```

출력:

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 CreateVirtualMfaDevice를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeactivateMfaDevice** 사용

다음 코드 예제는 DeactivateMfaDevice의 사용 방법을 보여 줍니다.

CLI

AWS CLI

MFA 디바이스 비활성화

이 명령은 사용자 Bob과 연결된 ARN arn:aws:iam::210987654321:mfa/BobsMFADevice의 가상 MFA 디바이스를 비활성화합니다.

```
aws iam deactivate-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 AWS에서 멀티 팩터 인증(MFA) 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeactivateMfaDevice를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 명령은 일련 번호 **123456789012**를 가진 사용자 **Bob**과 연결된 하드웨어 MFA 디바이스를 비활성화합니다.

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

예제 2: 이 명령은 ARN **arn:aws:iam::210987654321:mfa/David**를 가진 사용자 **David**과 연결된 가상 MFA 디바이스를 비활성화합니다. 단, 가상 MFA 디바이스는 계정에서 삭제되지 않습니다. 가상 디바이스는 여전히 존재하며 **Get-IAMVirtualMFADevice** 명령 출력에 나타납니다. 동일한 사용자를 위한 새 가상 MFA 디바이스를 생성하려면 먼저 **Remove-IAMVirtualMFADevice** 명령을 사용하여 이전 디바이스를 삭제해야 합니다.

```
Disable-IAMMFADevice -UserName "David" -SerialNumber
 "arn:aws:iam::210987654321:mfa/David"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeactivateMfaDevice를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteAccessKey** 사용

다음 코드 예제는 DeleteAccessKey의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 그룹 생성 및 사용자 추가
- 사용자 생성 및 역할 수임
- 읽기 전용 및 읽기-쓰기 사용자 생성
- 액세스 키 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteAccessKey를 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an WS Identity and Access Management (IAM) access key for the
 specified IAM user"
    echo "  -u user_name    The name of the user."
```

```
  echo "  -k access_key   The access key to delete."
  echo ""
}

# Retrieve the calling parameters.
while getopts "u:k:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    k) access_key="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

if [[ -z "$access_key" ]]; then
  errecho "ERROR: You must provide an access key with the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Username:   $user_name"
iecho "    Access key:   $access_key"
iecho ""

response=$(aws iam delete-access-key \
  --user-name "$user_name" \
  --access-key-id "$access_key")

local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
  fi

  iecho "delete-access-key response:$response"
  iecho

  return 0
}
```

- API 세부 정보는 AWS CLI 명령 참조의 DeleteAccessKey를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::deleteAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);

    auto outcome = iam.DeleteAccessKey(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting access key " << accessKeyID << " from user "
                  << userName << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
```

```
    }
    else {
        std::cout << "Successfully deleted access key " << accessKeyID
                  << " for IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DeleteAccessKey를 참조하세요.

CLI

AWS CLI

IAM 사용자의 액세스 키 삭제

다음 `delete-access-key` 명령은 이름이 Bob인 IAM 사용자의 지정된 액세스 키(액세스 키 ID 및 비밀 액세스 키)를 삭제합니다.

```
aws iam delete-access-key \
    --access-key-id AKIDPMS9RO4H3FEXAMPLE \
    --user-name Bob
```

이 명령은 출력을 생성하지 않습니다.

IAM 사용자에 대해 정의된 액세스 키를 나열하려면 `list-access-keys` 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteAccessKey를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(userName string, keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(context.TODO(),
 &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DeleteAccessKey를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username> <accessKey>\s

                Where:
                    username - The name of the user.\s
                    accessKey - The access key ID for the secret access key you
 want to delete.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessKey = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        deleteKey(iam, username, accessKey);
        iam.close();
    }

    public static void deleteKey(IamClient iam, String username, String
 accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();
```

```
            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteAccessKey를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

액세스 키를 삭제합니다.

```
import { DeleteAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const deleteAccessKey = (userName, accessKeyId) => {
  const command = new DeleteAccessKeyCommand({
    AccessKeyId: accessKeyId,
    UserName: userName,
  });
```

```
  return client.send(command);
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteAccessKey를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  UserName: "USER_NAME",
};

iam.deleteAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteAccessKey를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteKey(
    userNameVal: String,
    accessKey: String,
) {
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteAccessKey를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **Bob**이라는 사용자로부터 키 ID가 **AKIAIOSFODNN7EXAMPLE**인 AWS 액세스 키 페어를 삭제합니다.

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteAccessKey를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DeleteAccessKey를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 액세스 키를 나열, 생성, 비활성화 및 삭제합니다.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
 #{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded => e
    @logger.error("Error creating access key: limit exceeded. Cannot create
 more.")
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
```

```
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: "Inactive"
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteAccessKey를 참조하세요.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn delete_access_key(
    client: &iamClient,
    user: &User,
    key: &AccessKey,
) -> Result<(), iamError> {
    loop {
        match client
            .delete_access_key()
            .user_name(user.user_name())
            .access_key_id(key.access_key_id())
            .send()
            .await
        {
            Ok(_) => {
                break;
            }
            Err(e) => {
                println!("Can't delete the access key: {:?}", e);
                sleep(Duration::from_secs(2)).await;
            }
        }
    }
    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 DeleteAccessKey을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public func deleteAccessKey(user: IAMClientTypes.User? = nil,
                            key: IAMClientTypes.AccessKey) async throws {
    let userName: String?

    if user != nil {
        userName = user!.userName
    } else {
        userName = nil
    }

    let input = DeleteAccessKeyInput(
        accessKeyId: key.accessKeyId,
        userName: userName
    )
    do {
        _ = try await iamClient.deleteAccessKey(input: input)
    } catch {
        throw error
    }
}
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 DeleteAccessKey를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteAccountAlias** 사용

다음 코드 예제는 DeleteAccountAlias의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 계정 관리

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::deleteAccountAlias(const Aws::String &accountAlias,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccountAliasRequest request;
    request.SetAccountAlias(accountAlias);

    const auto outcome = iam.DeleteAccountAlias(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting account alias " << accountAlias << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted account alias " << accountAlias <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DeleteAccountAlias를 참조하세요.

CLI

AWS CLI

계정 별칭 삭제

다음 `delete-account-alias` 명령은 현재 계정의 별칭 `mycompany`를 제거합니다.

```
aws iam delete-account-alias \
    --account-alias mycompany
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 AWS 계정 ID 및 별칭을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteAccountAlias를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMAccountAlias(iam, alias);
        iam.close();
    }

    public static void deleteIAMAccountAlias(IamClient iam, String alias) {
        try {
            DeleteAccountAliasRequest request =
 DeleteAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();

            iam.deleteAccountAlias(request);
            System.out.println("Successfully deleted account alias " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
        System.out.println("Done");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteAccountAlias를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

계정 별칭을 삭제합니다.

```
import { DeleteAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias
 */
export const deleteAccountAlias = (alias) => {
  const command = new DeleteAccountAliasCommand({ AccountAlias: alias });

  return client.send(command);
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteAccountAlias를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteAccountAlias를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteIAMAccountAlias(alias: String) {
```

```
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteAccountAlias를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 AWS 계정에서 계정 별칭을 제거합니다. https://mycompanyaws.signin.aws.amazon.com/console에서 별칭을 사용하는 사용자 로그인 페이지가 더 이상 작동하지 않습니다. https://<accountidnumber>.signin.aws.amazon.com/console에서 AWS 계정 ID 번호와 함께 원래 URL을 사용해야 합니다.

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteAccountAlias를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def delete_alias(alias):
```

```
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DeleteAccountAlias를 참조하세요.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

계정 별칭을 나열하고, 생성하고, 삭제합니다.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
```

```
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteAccountAlias를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteAccountPasswordPolicy** 사용

다음 코드 예제는 DeleteAccountPasswordPolicy의 사용 방법을 보여 줍니다.

CLI

AWS CLI

현재 계정 암호 정책 삭제

다음 delete-account-password-policy 명령은 현재 계정의 암호 정책을 제거합니다.

```
aws iam delete-account-password-policy
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자의 계정 암호 정책 설정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteAccountPasswordPolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 AWS 계정의 암호 정책을 삭제하고 모든 값을 원래 기본값으로 재설정합니다. 암호 정책이 현재 존재하지 않는 경우 다음과 같은 오류 메시지가 나타납니다. PasswordPolicy 라는 계정 정책을 찾을 수 없습니다.

```
Remove-IAMAccountPasswordPolicy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteAccountPasswordPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteGroup** 사용

다음 코드 예제는 DeleteGroup의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 그룹 생성 및 사용자 추가

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupAsync(string groupName)
    {
        var response = await _IAMService.DeleteGroupAsync(new DeleteGroupRequest
{ GroupName = groupName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteGroup을 참조하십시오.

CLI

AWS CLI

IAM 그룹 삭제

다음 `delete-group` 명령은 이름이 `MyTestGroup`인 IAM 그룹을 삭제합니다.

```
aws iam delete-group \
    --group-name MyTestGroup
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 그룹 삭제를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteGroup을 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { DeleteGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} groupName
 */
export const deleteGroup = async (groupName) => {
  const command = new DeleteGroupCommand({
    GroupName: groupName,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteGroup을 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **MyTestGroup**이라는 IAM 그룹을 삭제합니다. 첫 번째 명령은 그룹 멤버인 IAM 사용자를 모두 제거하고, 두 번째 명령은 IAM 그룹을 삭제합니다. 두 명령 모두 확인 메시지 없이 작동합니다.

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteGroup을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteGroupPolicy** 사용

다음 코드 예제는 DeleteGroupPolicy의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 그룹 생성 및 사용자 추가

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete an IAM policy associated with an IAM group.
    /// </summary>
```

```
    /// <param name="groupName">The name of the IAM group associated with the
    /// policy.</param>
    /// <param name="policyName">The name of the policy to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupPolicyAsync(string groupName, string
 policyName)
    {
        var request = new DeleteGroupPolicyRequest()
        {
            GroupName = groupName,
            PolicyName = policyName,
        };

        var response = await _IAMService.DeleteGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteGroupPolicy를 참조하십시오.

CLI

AWS CLI

IAM 그룹에서 정책 삭제

다음 `delete-group-policy` 명령은 이름이 `Admins`인 그룹에서 이름이 `ExamplePolicy`인 정책을 삭제합니다.

```
aws iam delete-group-policy \
    --group-name Admins \
    --policy-name ExamplePolicy
```

이 명령은 출력을 생성하지 않습니다.

그룹에 연결된 정책을 보려면 `list-group-policies` 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 정책 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteGroupPolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM 그룹 **Testers**에서 **TesterPolicy**라는 인라인 정책을 제거합니다. 해당 그룹의 사용자는 해당 정책에 정의된 권한을 즉시 잃게 됩니다.

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteGroupPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteInstanceProfile** 사용

다음 코드 예제는 DeleteInstanceProfile의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 복원력이 뛰어난 서비스 구축 및 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
```

```
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
```

```
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteInstanceProfile을 참조하세요.

CLI

AWS CLI

인스턴스 프로파일 삭제

다음 `delete-instance-profile` 명령은 이름이 `ExampleInstanceProfile`인 인스턴스 프로파일을 삭제합니다.

```
aws iam delete-instance-profile \
    --instance-profile-name ExampleInstanceProfile
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 인스턴스 프로파일 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteInstanceProfile을 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  const client = new IAMClient({});
  await client.send(
    new DeleteInstanceProfileCommand({
      InstanceProfileName: NAMES.instanceProfileName,
    }),
  );
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteInstanceProfile을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **MyAppInstanceProfile**이라는 EC2 인스턴스 프로파일을 삭제합니다. 첫 번째 명령은 인스턴스 프로파일에서 모든 역할을 분리하고, 두 번째 명령은 인스턴스 프로파일을 삭제합니다.

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles |
 Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteInstanceProfile을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예제에서는 인스턴스 프로파일에서 역할을 제거하고 역할에 연결된 모든 정책을 분리하며 모든 리소스를 삭제합니다.

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
```

```
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is
                          created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    def delete_instance_profile(self, profile_name, role_name):
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
```

```
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )
            else:
                raise AutoScalerError(
                    f"Couldn't delete instance profile {profile_name} or detach "
                    f"policies and delete role {role_name}: {err}"
                )
```

- API에 대한 세부 정보는 Python용 AWS SDK(Boto3) API 참조의 DeleteInstanceProfile을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteLoginProfile** 사용

다음 코드 예제는 DeleteLoginProfile의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자의 암호 삭제

다음 `delete-login-profile` 명령은 이름이 Bob인 IAM 사용자의 암호를 삭제합니다.

```
aws iam delete-login-profile \
    --user-name Bob
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 암호 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteLoginProfile을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **Bob**이라는 IAM 사용자로부터 로그인 프로파일을 삭제합니다. 이렇게 하면 사용자가 AWS 콘솔에 로그인할 수 없습니다. 사용자 계정에 여전히 연결되어 있을 수 있는 AWS 액세스 키를 사용하여 사용자가 AWS CLI, PowerShell 또는 API 직접 호출을 실행하는 것을 막지는 못합니다.

```
Remove-IAMLoginProfile -UserName Bob
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteLoginProfile을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteOpenIdConnectProvider** 사용

다음 코드 예제는 `DeleteOpenIdConnectProvider`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM OpenID Connect ID 제공업체 삭제

이 예제는 제공업체 `example.oidcprovider.com`에 연결되는 IAM OIDC 제공업체를 삭제합니다.

```
aws iam delete-open-id-connect-provider \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM에서 OIDC(OpenID Connect) ID 제공업체 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteOpenIdConnectProvider를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 제공업체 **example.oidcprovider.com**에 연결되는 IAM OIDC 제공업체를 삭제합니다. 역할 신뢰 정책의 **Principal** 요소에서 이 제공업체를 참조하는 모든 역할을 업데이트하거나 삭제해야 합니다.

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteOpenIdConnectProvider를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeletePolicy** 사용

다음 코드 예제는 `DeletePolicy`의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 사용자 생성 및 역할 수임

- 읽기 전용 및 읽기-쓰기 사용자 생성
- 정책 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeletePolicy를 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
########################################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
########################################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

########################################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
########################################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

########################################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
########################################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an WS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
```

```
}

# Retrieve the calling parameters.
while getopts "n:h" option; do
  case "${option}" in
    n) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy arn with the -n parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Policy arn:  $policy_arn"
iecho ""

response=$(aws iam delete-policy \
  --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
  return 1
fi

iecho "delete-policy response:$response"
iecho

return 0
```

```
}
```

- API 세부 정보는 AWS CLI 명령 참조의 DeletePolicy를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::deletePolicy(const Aws::String &policyArn,
                               const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeletePolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.DeletePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting policy with arn " << policyArn << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted policy with arn " << policyArn
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DeletePolicy를 참조하십시오.

CLI

AWS CLI

IAM 정책 삭제

이 예제에서는 ARN이 arn:aws:iam::123456789012:policy/MySamplePolicy인 정책을 삭제합니다.

```
aws iam delete-policy \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeletePolicy를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(policyArn string) error {
```

```
 _, err := wrapper.IamClient.DeletePolicy(context.TODO(), &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DeletePolicy를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeletePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <policyARN>\s

                Where:
                    policyARN - A policy ARN value to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    public static void deleteIAMPolicy(IamClient iam, String policyARN) {
        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(policyARN)
                    .build();

            iam.deletePolicy(request);
            System.out.println("Successfully deleted the policy");

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeletePolicy를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

정책을 삭제합니다.

```
import { DeletePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const deletePolicy = (policyArn) => {
  const command = new DeletePolicyCommand({ PolicyArn: policyArn });
  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeletePolicy를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
```

```
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeletePolicy를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/MySamplePolicy**인 정책을 삭제합니다. 정책을 삭제하려면 먼저 **Remove-IAMPolicyVersion**을 실행하여 기본값을 제외한 모든 버전을 삭제해야 합니다. 또한 모든 IAM 사용자, 그룹 또는 역할에서 정책을 분리해야 합니다.

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

예제 2: 이 예제는 먼저 기본이 아닌 모든 정책 버전을 삭제하고 연결된 모든 IAM 엔티티에서 분리한 다음 마지막으로 정책 자체를 삭제하여 정책을 삭제합니다. 줄 1은 정책 객체를 검색합니다. 줄 2는 기본값으로 플래그가 지정되지 않은 모든 정책 버전을 컬렉션으로 검색한 다음 컬렉션의 각 정책을 삭제합니다. 줄 3은 정책이 연결된 모든 IAM 사용자, 그룹 및 역할을 검색합니다. 줄 4~6은 연결된 각 엔터티에서 정책을 분리합니다. 마지막 줄은 이 명령을 사용하여 관리형 정책과 나머지 기본 버전을 제거합니다. 이 예제에는 확인 프롬프트를 표시하지 않는 데 필요한 모든 줄에 **-Force** 스위치 파라미터가 포함되어 있습니다.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn
Remove-IAMPolicy $pol.Arn -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeletePolicy를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DeletePolicy를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn delete_policy(client: &iamClient, policy: Policy) -> Result<(),
 iamError> {
    client
        .delete_policy()
        .policy_arn(policy.arn.unwrap())
        .send()
        .await?;
    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 DeletePolicy을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public func deletePolicy(policy: IAMClientTypes.Policy) async throws {
        let input = DeletePolicyInput(
            policyArn: policy.arn
        )
        do {
            _ = try await iamClient.deletePolicy(input: input)
        } catch {
            throw error
        }
    }
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 DeletePolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeletePolicyVersion** 사용

다음 코드 예제는 DeletePolicyVersion의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 정책 관리
- 정책 버전 롤백

CLI

AWS CLI

관리형 정책의 버전 삭제

이 예제는 ARN이 arn:aws:iam::123456789012:policy/MySamplePolicy인 정책에서 v2로 식별된 버전을 삭제합니다.

```
aws iam delete-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeletePolicyVersion을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/MySamplePolicy**인 정책에서 **v2**로 식별된 버전을 삭제합니다.

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MySamplePolicy -VersionID v2
```

예제 2: 이 예제는 먼저 기본이 아닌 모든 정책 버전을 삭제한 다음 정책 자체를 삭제하여 정책을 삭제합니다. 줄 1은 정책 객체를 검색합니다. 줄 2는 기본값으로 플래그가 지정되지 않은 모든 정책 버전을 컬렉션으로 검색한 다음 이 명령을 사용하여 컬렉션의 각 정책을 삭제합니다. 마지막 줄은 나머지 기본 버전뿐만 아니라 정책 자체도 제거합니다. 관리형 정책을 성공적으로 삭제하려면 **Unregister-IAMUserPolicy**, **Unregister-IAMGroupPolicy** 및 **Unregister-IAMRolePolicy** 명령을 사용하여 모든 사용자, 그룹 또는 역할에서 정책을 분리해야 합니다. **Remove-IAMPolicy** cmdlet의 예를 참조하세요.

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeletePolicyVersion을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteRole** 사용

다음 코드 예제는 DeleteRole의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 사용자 생성 및 역할 수임
- 역할 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
{ RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteRole을 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
```

```
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an WS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
```

```
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}
```

- API 세부 정보는 AWS CLI 명령 참조의 DeleteRole을 참조하십시오.

CLI

AWS CLI

IAM 역할 삭제

다음 `delete-role` 명령은 이름이 `Test-Role`인 역할을 제거합니다.

```
aws iam delete-role \
    --role-name Test-Role
```

이 명령은 출력을 생성하지 않습니다.

역할을 삭제하려면 먼저 인스턴스 프로파일에서 역할을 제거하고 (`remove-role-from-instance-profile`), 관리형 정책을 모두 분리하고(`detach-role-policy`), 역할에 연결된 인라인 정책을 모두 삭제해야 합니다(`delete-role-policy`).

자세한 내용은 AWS IAM 사용 설명서의 IAM 역할 생성 및 인스턴스 프로파일 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteRole을 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}
```

```
// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(roleName string) error {
 _, err := wrapper.IamClient.DeleteRole(context.TODO(), &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DeleteRole을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

역할을 삭제합니다.

```
import { DeleteRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteRole = (roleName) => {
  const command = new DeleteRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteRole을 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 현재 IAM 계정에서 **MyNewRole**이라는 역할을 삭제합니다. 역할을 삭제하려면 먼저 **Unregister-IAMRolePolicy** 명령을 사용하여 관리형 정책을 분리해야 합니다. 인라인 정책은 역할과 함께 삭제됩니다.

```
Remove-IAMRole -RoleName MyNewRole
```

예제 2: 이 예제는 **MyNewRole**이라는 역할에서 관리형 정책을 분리한 다음 역할을 삭제합니다. 줄 1은 역할에 연결된 모든 관리형 정책을 컬렉션으로 검색한 다음 컬렉션의 각 정책을 역할에서 분리합니다. 줄 2는 역할 자체를 삭제합니다. 인라인 정책은 역할과 함께 삭제됩니다.

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteRole을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
```

```
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DeleteRole를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  # Deletes a role and its attached policies.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    begin
      # Detach and delete attached policies
      @iam_client.list_attached_role_policies(role_name: role_name).each do |
response|
        response.attached_policies.each do |policy|
          @iam_client.detach_role_policy({
                                           role_name: role_name,
                                           policy_arn: policy.policy_arn
                                         })
          # Check if the policy is a customer managed policy (not AWS managed)
          unless policy.policy_arn.include?("aws:policy/")
            @iam_client.delete_policy({ policy_arn: policy.policy_arn })
            @logger.info("Deleted customer managed policy
 #{policy.policy_name}.")
          end
        end
```

```
      end

      # Delete the role
      @iam_client.delete_role({ role_name: role_name })
      @logger.info("Deleted role #{role_name}.")
    rescue Aws::IAM::Errors::ServiceError => e
      @logger.error("Couldn't detach policies and delete role #{role_name}.
 Here's why:")
      @logger.error("\t#{e.code}: #{e.message}")
      raise
    end
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteRole을 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn delete_role(client: &iamClient, role: &Role) -> Result<(), iamError>
 {
    let role = role.clone();
    while client
        .delete_role()
        .role_name(role.role_name())
        .send()
        .await
        .is_err()
    {
        sleep(Duration::from_secs(2)).await;
    }
    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 DeleteRole을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public func deleteRole(role: IAMClientTypes.Role) async throws {
    let input = DeleteRoleInput(
        roleName: role.roleName
    )
    do {
        _ = try await iamClient.deleteRole(input: input)
    } catch {
        throw error
    }
}
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 DeleteRole을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **`DeleteRolePermissionsBoundary`** 사용

다음 코드 예제는 `DeleteRolePermissionsBoundary`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 역할에서 권한 경계 삭제

다음 `delete-role-permissions-boundary` 예제는 지정된 IAM 역할의 권한 경계를 삭제합니다. 역할에 권한 경계를 적용하려면 `put-role-permissions-boundary` 명령을 사용합니다.

```
aws iam delete-role-permissions-boundary \
    --role-name lambda-application-role
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteRolePermissionsBoundary를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM 역할에 연결된 권한 경계를 제거하는 방법을 보여줍니다.

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteRolePermissionsBoundary를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteRolePolicy** 사용

다음 코드 예제는 `DeleteRolePolicy`의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteRolePolicy를 참조하십시오.

CLI

AWS CLI

IAM 역할에서 정책 제거

다음 delete-role-policy 명령은 이름이 Test-Role인 역할에서 이름이 ExamplePolicy인 정책을 제거합니다.

```
aws iam delete-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 역할 변경을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteRolePolicy를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { DeleteRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 */
export const deleteRolePolicy = (roleName, policyName) => {
  const command = new DeleteRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
  });
  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteRolePolicy를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM 역할 **S3BackupRole**에 포함된 인라인 정책 **S3AccessPolicy**를 삭제합니다.

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteRolePolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteSAMLProvider** 사용

다음 코드 예제는 DeleteSAMLProvider의 사용 방법을 보여 줍니다.

CLI

AWS CLI

SAML 공급자 삭제

이 예제에서는 ARN이 arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider인 IAM SAML 2.0 공급자를 삭제합니다.

```
aws iam delete-saml-provider \
--saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM SAML 자격 증명 공급자 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteSAMLProvider를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { DeleteSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} providerArn
 * @returns
 */
export const deleteSAMLProvider = async (providerArn) => {
  const command = new DeleteSAMLProviderCommand({
    SAMLProviderArn: providerArn,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteSAMLProvider를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider**인 IAM SAML 2.0 제공업체를 삭제합니다.

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteSAMLProvider를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteServerCertificate** 사용

다음 코드 예제는 DeleteServerCertificate의 사용 방법을 보여 줍니다.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::deleteServerCertificate(const Aws::String &certificateName,
                                          const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeleteServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    const auto outcome = iam.DeleteServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error deleting server certificate " << certificateName
 <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
```

```
    else {
        std::cout << "Successfully deleted server certificate " <<
 certificateName
                  << std::endl;
    }

    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DeleteServerCertificate를 참조하세요.

CLI

AWS CLI

AWS 계정에서 서버 인증서 삭제

다음 `delete-server-certificate` 명령은 AWS 계정에서 지정된 서버 인증서를 제거합니다.

```
aws iam delete-server-certificate \
    --server-certificate-name myUpdatedServerCertificate
```

이 명령은 출력을 생성하지 않습니다.

AWS 계정에서 사용 가능한 서버 인증서를 나열하려면 `list-server-certificates` 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM에서 서버 인증서 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteServerCertificate를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

서버 인증서를 삭제합니다.

```
import { DeleteServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
 */
export const deleteServerCertificate = (certName) => {
  const command = new DeleteServerCertificateCommand({
    ServerCertificateName: certName,
  });

  return client.send(command);
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteServerCertificate를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
```

```
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteServerCertificate를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **MyServerCert**라는 서버 인증서를 삭제합니다.

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteServerCertificate를 참조하세요.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

서버 인증서를 나열하고, 업데이트하고, 삭제합니다.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
```

```
    @logger.progname = "ServerCertificateManager"
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key,
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info("No server certificates found.")
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
```

```
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteServerCertificate를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteServiceLinkedRole** 사용

다음 코드 예제는 DeleteServiceLinkedRole의 사용 방법을 보여 줍니다.

CLI

AWS CLI

서비스 연결 역할 삭제

다음 delete-service-linked-role 예제에서는 더 이상 필요하지 않은 지정된 서비스 연결 역할을 삭제합니다. 삭제는 비동기식으로 이루어집니다. get-service-linked-role-deletion-status 명령을 사용하여 삭제 상태를 확인하고 언제 삭제되는지 확인할 수 있습니다.

```
aws iam delete-service-linked-role \
    --role-name AWSServiceRoleForLexBots
```

출력:

```
{
    "DeletionTaskId": "task/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE"
}
```

자세한 내용은 AWS IAM 사용 설명서의 서비스 연결 역할 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteServiceLinkedRole을 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(roleName string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(context.TODO(),
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
 roleName, err)
 }
```

```
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DeleteServiceLinkedRole을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { DeleteServiceLinkedRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteServiceLinkedRole = (roleName) => {
  const command = new DeleteServiceLinkedRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteServiceLinkedRole을 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 서비스 연결 역할을 삭제했습니다. 서비스에서 여전히 이 역할을 사용하고 있는 경우 이 명령을 실행하면 실패합니다.

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteServiceLinkedRole을 참조하세요.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Deletes a service-linked role.
#
# @param role_name [String] The name of the role to delete.
def delete_service_linked_role(role_name)
  response = @iam_client.delete_service_linked_role(role_name: role_name)
  task_id = response.deletion_task_id
  check_deletion_status(role_name, task_id)
rescue Aws::Errors::ServiceError => e
  handle_deletion_error(e, role_name)
end

private

# Checks the deletion status of a service-linked role
#
# @param role_name [String] The name of the role being deleted
# @param task_id [String] The task ID for the deletion process
def check_deletion_status(role_name, task_id)
  loop do
    response = @iam_client.get_service_linked_role_deletion_status(
      deletion_task_id: task_id)
    status = response.status
    @logger.info("Deletion of #{role_name} #{status}.")
    break if %w[SUCCEEDED FAILED].include?(status)
    sleep(3)
```

```
    end
  end

  # Handles deletion error
  #
  # @param e [Aws::Errors::ServiceError] The error encountered during deletion
  # @param role_name [String] The name of the role attempted to delete
  def handle_deletion_error(e, role_name)
    unless e.code == "NoSuchEntity"
      @logger.error("Couldn't delete #{role_name}. Here's why:")
      @logger.error("\t#{e.code}: #{e.message}")
      raise
    end
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteServiceLinkedRole을 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn delete_service_linked_role(
    client: &iamClient,
    role_name: &str,
) -> Result<(), iamError> {
    client
        .delete_service_linked_role()
        .role_name(role_name)
        .send()
        .await?;

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 DeleteServiceLinkedRole을 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteSigningCertificate** 사용

다음 코드 예제는 DeleteSigningCertificate의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자의 서명 인증서 삭제

다음 delete-signing-certificate 명령은 Bob이라는 IAM 사용자에 대해 지정된 서명 인증서를 삭제합니다.

```
aws iam delete-signing-certificate \
    --user-name Bob \
    --certificate-id TA7SMP42TDN5Z26OBPJE7EXAMPLE
```

이 명령은 출력을 생성하지 않습니다.

서명 인증서의 ID를 가져오려면 list-signing-certificates 명령을 사용합니다.

자세한 내용은 Amazon EC2 사용 설명서의 서명 인증서 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteSigningCertificate를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **Bob**이라는 IAM 사용자로부터 ID가 **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU**인 서명 인증서를 삭제합니다.

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteSigningCertificate를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteUser** 사용

다음 코드 예제는 DeleteUser의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 그룹 생성 및 사용자 추가
- 사용자 생성 및 역할 수임
- 읽기 전용 및 읽기-쓰기 사용자 생성

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteUser를 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
```

```
########################################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an WS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
```

```
    return 1
  fi

  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- API 세부 정보는 AWS CLI 명령 참조의 DeleteUser를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteUserRequest request;
    request.SetUserName(userName);
    auto outcome = iam.DeleteUser(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting IAM user " << userName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;;
    }
```

```
    else {
        std::cout << "Successfully deleted IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DeleteUser를 참조하십시오.

CLI

AWS CLI

IAM 사용자 삭제

다음 `delete-user` 명령은 현재 계정에서 이름이 Bob인 IAM 사용자를 제거합니다.

```
aws iam delete-user \
    --user-name Bob
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 삭제를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteUser를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
```

```
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(userName string) error {
 _, err := wrapper.IamClient.DeleteUser(context.TODO(), &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DeleteUser를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class DeleteUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
                    userName - The name of the user to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteUser를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

사용자를 삭제합니다.

```
import { DeleteUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const deleteUser = (name) => {
  const command = new DeleteUserCommand({ UserName: name });
  return client.send(command);
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteUser를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    console.log("User " + process.argv[2] + " does not exist.");
  } else {
    iam.deleteUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DeleteUser를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteIAMUser(userNameVal: String) {
```

```
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    // To delete a user, ensure that the user's access keys are deleted first.
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DeleteUser를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **Bob**이라는 IAM 사용자를 삭제합니다.

```
Remove-IAMUser -UserName Bob
```

예제 2: 이 예제는 먼저 삭제해야 하는 모든 요소와 함께 **Theresa**라는 IAM 사용자를 삭제합니다.

```
$name = "Theresa"

# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName $name -Force }

# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 $name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn
 $pol.PolicyArn -UserName $name }
```

```
# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
 $cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
 suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}

# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteUser를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
```

```
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DeleteUser를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
 # Deletes a user and their associated resources
 #
 # @param user_name [String] The name of the user to delete
 def delete_user(user_name)
   user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
   user.each do |key|
     @iam_client.delete_access_key({ access_key_id: key.access_key_id,
user_name: user_name })
     @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
   end

   @iam_client.delete_user(user_name: user_name)
   @logger.info("Deleted user '#{user_name}'.")
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error deleting user '#{user_name}': #{e.message}")
```

```
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteUser를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn delete_user(client: &iamClient, user: &User) -> Result<(),
 SdkError<DeleteUserError>> {
    let user = user.clone();
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<(), SdkError<DeleteUserError>> = loop {
        match client
            .delete_user()
            .user_name(user.user_name())
            .send()
            .await
        {
            Ok(_) => {
                break Ok(());
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
        }
    };

    response
```

```
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 DeleteUser을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public func deleteUser(user: IAMClientTypes.User) async throws {
    let input = DeleteUserInput(
        userName: user.userName
    )
    do {
        _ = try await iamClient.deleteUser(input: input)
    } catch {
        throw error
    }
}
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 DeleteUser를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteUserPermissionsBoundary** 사용

다음 코드 예제는 DeleteUserPermissionsBoundary의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자에게서 권한 경계 삭제

다음 delete-user-permissions-boundary 예제는 intern이라는 IAM 사용자에게 연결된 권한 경계를 삭제합니다. 사용자에게 권한 경계를 적용하려면 put-user-permissions-boundary 명령을 사용합니다.

```
aws iam delete-user-permissions-boundary \
    --user-name intern
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteUserPermissionsBoundary를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM 사용자에게 연결된 권한 경계를 제거하는 방법을 보여줍니다.

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteUserPermissionsBoundary를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteUserPolicy** 사용

다음 코드 예제는 DeleteUserPolicy의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 사용자 생성 및 역할 수임

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DeleteUserPolicy를 참조하십시오.

CLI

AWS CLI

IAM 사용자에서 정책 제거

다음 delete-user-policy 명령은 이름이 Bob인 IAM 사용자에서 지정된 정책을 제거합니다.

```
aws iam delete-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy
```

이 명령은 출력을 생성하지 않습니다.

IAM 사용자의 정책 목록을 가져오려면 list-user-policies 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteUserPolicy를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(userName string, policyName string)
 error {
 _, err := wrapper.IamClient.DeleteUserPolicy(context.TODO(),
 &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
```

```
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DeleteUserPolicy를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **Bob**이라는 IAM 사용자에게 포함된 **AccessToEC2Policy**라는 인라인 정책을 삭제합니다.

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

예제 2: 이 예제는 **Theresa**라는 IAM 사용자에게 포함된 모든 인라인 정책을 찾아 삭제합니다.

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteUserPolicy를 참조하세요.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
 # Deletes a user and their associated resources
 #
```

```
  # @param user_name [String] The name of the user to delete
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id,
 user_name: user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteUserPolicy를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn delete_user_policy(
    client: &iamClient,
    user: &User,
    policy_name: &str,
) -> Result<(), SdkError<DeleteUserPolicyError>> {
    client
        .delete_user_policy()
        .user_name(user.user_name())
        .policy_name(policy_name)
        .send()
        .await?;

    Ok(())
```

```
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 DeleteUserPolicy을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    func deleteUserPolicy(user: IAMClientTypes.User, policyName: String) async
throws {
        let input = DeleteUserPolicyInput(
            policyName: policyName,
            userName: user.userName
        )
        do {
            _ = try await iamClient.deleteUserPolicy(input: input)
        } catch {
            throw error
        }
    }
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 DeleteUserPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DeleteVirtualMfaDevice** 사용

다음 코드 예제는 DeleteVirtualMfaDevice의 사용 방법을 보여 줍니다.

CLI

AWS CLI

가상 MFA 디바이스 제거

다음 delete-virtual-mfa-device 명령은 현재 계정에서 지정된 MFA 디바이스를 제거합니다.

```
aws iam delete-virtual-mfa-device \
    --serial-number arn:aws:iam::123456789012:mfa/MFATest
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 MFA 디바이스 비활성화를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DeleteVirtualMfaDevice를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:mfa/bob**인 IAM 가상 MFA 디바이스를 삭제합니다.

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

예제 2: 이 예제는 IAM 사용자 Theresa에게 MFA 디바이스가 할당되었는지 확인합니다. 발견된 디바이스는 IAM 사용자에 대해 비활성화됩니다. 디바이스가 가상이면 삭제됩니다.

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DeleteVirtualMfaDevice를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DetachGroupPolicy** 사용

다음 코드 예제는 DetachGroupPolicy의 사용 방법을 보여 줍니다.

CLI

AWS CLI

그룹에서 정책 분리

이 예제는 Testers라는 그룹에서 ARN arn:aws:iam::123456789012:policy/TesterAccessPolicy가 있는 관리형 정책을 제거합니다.

```
aws iam detach-group-policy \
    --group-name Testers \
    --policy-arn arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 그룹 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DetachGroupPolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/TesterAccessPolicy**인 관리형 그룹 정책을 **Testers**라는 그룹에서 분리합니다.

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

예제 2: 이 예제는 **Testers**라는 그룹에 연결된 모든 관리형 정책을 찾아 그룹에서 분리합니다.

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DetachGroupPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DetachRolePolicy** 사용

다음 코드 예제는 DetachRolePolicy의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 사용자 생성 및 역할 수임
- 역할 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
    {
```

```
        var response = await _IAMService.DetachRolePolicyAsync(new
 DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 DetachRolePolicy를 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
```

```
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
```

```
    usage
    return 1
  fi

  response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API 세부 정보는 AWS CLI 명령 참조의 DetachRolePolicy를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DetachRolePolicyRequest detachRequest;
    detachRequest.SetRoleName(roleName);
    detachRequest.SetPolicyArn(policyArn);

    auto detachOutcome = iam.DetachRolePolicy(detachRequest);
    if (!detachOutcome.IsSuccess()) {
```

```
        std::cerr << "Failed to detach policy " << policyArn << " from role "
                  << roleName << ": " << detachOutcome.GetError().GetMessage() <<
                  std::endl;
    }
    else {
        std::cout << "Successfully detached policy " << policyArn << " from role
"
                  << roleName << std::endl;
    }

    return detachOutcome.IsSuccess();
```

- API 세부 정보는 AWS SDK for C++ API 참조의 DetachRolePolicy를 참조하세요.

CLI

AWS CLI

역할에서 정책 분리

이 예제에서는 FedTesterRole을 호출한 역할에서 ARN이 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy인 관리형 정책을 제거합니다.

```
aws iam detach-role-policy \
    --role-name FedTesterRole \
    --policy-arn arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 역할 변경을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DetachRolePolicy를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(roleName string, policyArn string)
 error {
 _, err := wrapper.IamClient.DetachRolePolicy(context.TODO(),
 &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 DetachRolePolicy를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
 Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
 Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String roleName = args[0];
        String policyArn = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        detachPolicy(iam, roleName, policyArn);
        System.out.println("Done");
        iam.close();
    }

    public static void detachPolicy(IamClient iam, String roleName, String
 policyArn) {
        try {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                    " from role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DetachRolePolicy를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

태그를 분리합니다.

```
import { DetachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const detachRolePolicy = (policyArn, roleName) => {
  const command = new DetachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DetachRolePolicy를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
```

```
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        var params = {
          PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
          RoleName: process.argv[2],
        };
        iam.detachRolePolicy(params, function (err, data) {
          if (err) {
            console.log("Unable to detach policy from role", err);
          } else {
            console.log("Policy detached from role successfully");
            process.exit();
          }
        });
      }
    });
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 DetachRolePolicy를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun detachPolicy(
```

```
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role
 $roleNameVal")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 DetachRolePolicy를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy**인 관리형 그룹 정책을 **FedTesterRole**라는 역할에서 분리합니다.

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

예제 2: 이 예제는 **FedTesterRole**라는 역할에 연결된 모든 관리형 정책을 찾아 역할에서 분리합니다.

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy
 -Rolename FedTesterRole
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DetachRolePolicy를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

Boto3 정책 객체를 사용하여 역할에서 정책을 분리합니다.

```
def detach_from_role(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).detach_role(RoleName=role_name)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

Boto3 역할 객체를 사용하여 역할에서 정책을 분리합니다.

```
def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
```

```
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 DetachRolePolicy를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 역할 정책을 나열, 생성, 연결 및 분리합니다.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
```

```
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
```

```
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DetachRolePolicy를 참조하세요.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn detach_role_policy(
    client: &iamClient,
    role_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_role_policy()
        .role_name(role_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 DetachRolePolicy을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public func detachRolePolicy(policy: IAMClientTypes.Policy, role:
IAMClientTypes.Role) async throws {
        let input = DetachRolePolicyInput(
            policyArn: policy.arn,
            roleName: role.roleName
        )

        do {
            _ = try await iamClient.detachRolePolicy(input: input)
        } catch {
            throw error
        }
    }
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 DetachRolePolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **`DetachUserPolicy`** 사용

다음 코드 예제는 `DetachUserPolicy`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 읽기 전용 및 읽기-쓰기 사용자 생성

CLI

AWS CLI

사용자에서 정책 분리

이 예제에서는 사용자 Bob에서 ARN이 arn:aws:iam::123456789012:policy/TesterPolicy인 관리형 정책을 제거합니다.

```
aws iam detach-user-policy \
    --user-name Bob \
    --policy-arn arn:aws:iam::123456789012:policy/TesterPolicy
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자의 권한 변경을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DetachUserPolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/TesterPolicy**인 관리형 정책을 **Bob**라는 IAM 사용자에게서 분리합니다.

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

예제 2: 이 예제는 **Theresa**라는 IAM 사용자에게 연결된 모든 관리형 정책을 찾아 사용자에게서 분리합니다.

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -
Username Theresa
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DetachUserPolicy를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 DetachUserPolicy를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
 # Detaches a policy from a user
 #
 # @param user_name [String] The name of the user
 # @param policy_arn [String] The ARN of the policy to detach
 # @return [Boolean] true if the policy was successfully detached, false
otherwise
 def detach_user_policy(user_name, policy_arn)
   @iam_client.detach_user_policy(
     user_name: user_name,
```

```
      policy_arn: policy_arn
    )
    @logger.info("Policy '#{policy_arn}' detached from user '#{user_name}'
 successfully.")
    true
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error detaching policy: Policy or user does not exist.")
    false
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from user '#{user_name}':
 #{e.message}")
    false
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DetachUserPolicy를 참조하세요.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn detach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 DetachUserPolicy을 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **`EnableMfaDevice`** 사용

다음 코드 예제는 `EnableMfaDevice`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

MFA 디바이스 활성화

`create-virtual-mfa-device` 명령을 사용하여 새 가상 MFA 디바이스를 생성한 후 사용자에게 MFA 디바이스를 할당할 수 있습니다. 다음 `enable-mfa-device` 예제는 일련 번호가 `arn:aws:iam::210987654321:mfa/BobsMFADevice`인 MFA 디바이스를 사용자 Bob에게 할당합니다. 또한 이 명령은 가상 MFA 디바이스의 처음 두 코드를 순서대로 포함하여 디바이스를 AWS와 동기화합니다.

```
aws iam enable-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \
    --authentication-code1 123456 \
    --authentication-code2 789012
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 가상 멀티 팩터 인증(MFA) 디바이스 활성화를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 EnableMfaDevice를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 명령은 일련 번호 **987654321098**을 사용하여 하드웨어 MFA 디바이스를 활성화하고 디바이스를 사용자 **Bob**과 연결합니다. 여기에는 디바이스의 처음 두 코드가 순서대로 포함됩니다.

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

예제 2: 이 예제는 가상 MFA 디바이스를 생성하고 활성화합니다. 첫 번째 명령은 가상 디바이스를 생성하고 **$MFADevice** 변수에 디바이스의 객체 표현을 반환합니다. **.Base32StringSeed** 또는 **QRCodePng** 속성을 사용하여 사용자의 소프트웨어 애플리케이션을 구성할 수 있습니다. 마지막 명령은 사용자 **David**에게 디바이스를 할당하고 일련 번호로 디바이스를 식별합니다. 또한 이 명령은 가상 MFA 디바이스의 처음 두 코드를 순서대로 포함하여 디바이스를 AWS와 동기화합니다.

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"
# see example for New-IAMVirtualMFADevice to see how to configure the software
 program with PNG or base32 seed code
Enable-IAMMFADevice -UserName "David" -SerialNumber -SerialNumber
 $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2
 "13572468"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 EnableMfaDevice를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GenerateCredentialReport** 사용

다음 코드 예제는 GenerateCredentialReport의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 계정 관리

CLI

AWS CLI

보안 인증 보고서 생성

다음 예제에서는 AWS 계정에 대해 보안 인증 보고서 생성을 시도합니다.

```
aws iam generate-credential-report
```

출력:

```
{
    "State":  "STARTED",
    "Description": "No report exists. Starting a new report generation task"
}
```

자세한 내용은 AWS IAM 사용 설명서의 AWS 계정의 보안 인증 보고서 가져오기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GenerateCredentialReport를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 4시간마다 수행할 수 있는 새 보고서 생성을 요청합니다. 마지막 보고서가 아직 최신인 경우 State 필드에 **COMPLETE**가 표시됩니다. 완성된 보고서를 보려면 **Get-IAMCredentialReport**를 사용합니다.

```
Request-IAMCredentialReport
```

출력:

```
Description                                                    State
-----------                                                    -----
No report exists. Starting a new report generation task        STARTED
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GenerateCredentialReport를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four
 hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
 account.")
        raise
    else:
        return response
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 GenerateCredentialReport를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GenerateServiceLastAccessedDetails** 사용

다음 코드 예제는 GenerateServiceLastAccessedDetails의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: 사용자 지정 정책에 대한 서비스 액세스 보고서 생성

다음 generate-service-last-accessed-details 예제는 백그라운드 작업을 시작하여 intern-boundary라는 사용자 지정 정책으로 IAM 사용자와 기타 엔터티가 액세스하는 서비스를 나열하는 보고서를 생성합니다. 보고서가 생성된 후에는 get-service-last-accessed-details 명령을 실행하여 보고서를 표시할 수 있습니다.

```
aws iam generate-service-last-accessed-details \
    --arn arn:aws:iam::123456789012:policy/intern-boundary
```

출력:

```
{
    "JobId": "2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc"
}
```

예제 2: AWS 관리형 AdministratorAccess 정책에 대한 서비스 액세스 보고서 생성

다음 generate-service-last-accessed-details 예제는 백그라운드 작업을 시작하여 AWS 관리형 AdministratorAccess 정책으로 IAM 사용자와 기타 엔터티가 액세스하는 서비스를 나열하는 보고서를 생성합니다. 보고서가 생성된 후에는 get-service-last-accessed-details 명령을 실행하여 보고서를 표시할 수 있습니다.

```
aws iam generate-service-last-accessed-details \
    --arn arn:aws:iam::aws:policy/AdministratorAccess
```

출력:

```
{
    "JobId": "78b6c2ba-d09e-6xmp-7039-ecde30b26916"
}
```

자세한 내용은 AWS IAM 사용 설명서의 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GenerateServiceLastAccessedDetails를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 GenerateServiceLastAccessedDetails API와 동등한 cmdlet입니다. 이는 Get-IAMServiceLastAccessedDetail과 Get-IAMServiceLastAccessedDetailWithEntity에서 사용할 수 있는 작업 ID를 제공합니다.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GenerateServiceLastAccessedDetails를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetAccessKeyLastUsed** 사용

다음 코드 예제는 GetAccessKeyLastUsed의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 액세스 키 관리

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::accessKeyLastUsed(const Aws::String &secretKeyID,
                                    const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetAccessKeyLastUsedRequest request;

    request.SetAccessKeyId(secretKeyID);

    Aws::IAM::Model::GetAccessKeyLastUsedOutcome outcome =
 iam.GetAccessKeyLastUsed(
            request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error querying last used time for access key " <<
                  secretKeyID << ":" << outcome.GetError().GetMessage() <<
 std::endl;
    }
    else {
        Aws::String lastUsedTimeString =
                outcome.GetResult()
                        .GetAccessKeyLastUsed()
                        .GetLastUsedDate()
                        .ToGmtString(Aws::Utils::DateFormat::ISO_8601);
        std::cout << "Access key " << secretKeyID << " last used at time " <<
                  lastUsedTimeString << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 GetAccessKeyLastUsed를 참조하십시오.

CLI

AWS CLI

지정된 액세스 키가 마지막으로 사용된 경우에 대한 정보 검색

다음 예제에서는 액세스 키 ABCDEXAMPLE이 마지막으로 사용된 시간에 대한 정보를 검색합니다.

```
aws iam get-access-key-last-used \
    --access-key-id ABCDEXAMPLE
```

출력:

```
{
    "UserName":  "Bob",
    "AccessKeyLastUsed": {
        "Region": "us-east-1",
        "ServiceName": "iam",
        "LastUsedDate": "2015-06-16T22:45:00Z"
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetAccessKeyLastUsed를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

액세스 키를 가져옵니다.

```
import { GetAccessKeyLastUsedCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} accessKeyId
 */
export const getAccessKeyLastUsed = async (accessKeyId) => {
  const command = new GetAccessKeyLastUsedCommand({
    AccessKeyId: accessKeyId,
```

```
  });

  const response = await client.send(command);

  if (response.AccessKeyLastUsed?.LastUsedDate) {
    console.log(`
    ${accessKeyId} was last used by ${response.UserName} via
    the ${response.AccessKeyLastUsed.ServiceName} service on
    ${response.AccessKeyLastUsed.LastUsedDate.toISOString()}
    `);
  }

  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetAccessKeyLastUsed를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getAccessKeyLastUsed(
  { AccessKeyId: "ACCESS_KEY_ID" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
```

```
      console.log("Success", data.AccessKeyLastUsed);
    }
  }
);
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetAccessKeyLastUsed를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 제공된 액세스 키의 소유 사용자 이름과 마지막 사용 정보를 반환합니다.

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetAccessKeyLastUsed를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
```

```
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 GetAccessKeyLastUsed를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetAccountAuthorizationDetails** 사용

다음 코드 예제는 GetAccountAuthorizationDetails의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 계정 관리

CLI

AWS CLI

AWS 계정 IAM 사용자, 그룹, 역할 및 정책 나열

다음 get-account-authorization-details 명령은 AWS 계정의 모든 IAM 사용자, 그룹, 역할 및 정책에 대한 정보를 반환합니다.

```
aws iam get-account-authorization-details
```

출력:

```
{
    "RoleDetailList": [
        {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "RoleId": "AROA1234567890EXAMPLE",
            "CreateDate": "2014-07-30T17:09:20Z",
            "InstanceProfileList": [
                {
                    "InstanceProfileId": "AIPA1234567890EXAMPLE",
                    "Roles": [
                        {
                            "AssumeRolePolicyDocument": {
                                "Version": "2012-10-17",
                                "Statement": [
                                    {
                                        "Sid": "",
                                        "Effect": "Allow",
                                        "Principal": {
                                            "Service": "ec2.amazonaws.com"
                                        },
                                        "Action": "sts:AssumeRole"
                                    }
                                ]
                            },
                            "RoleId": "AROA1234567890EXAMPLE",
                            "CreateDate": "2014-07-30T17:09:20Z",
                            "RoleName": "EC2role",
```

```
                            "Path": "/",
                            "Arn": "arn:aws:iam::123456789012:role/EC2role"
                        }
                    ],
                    "CreateDate": "2014-07-30T17:09:20Z",
                    "InstanceProfileName": "EC2role",
                    "Path": "/",
                    "Arn": "arn:aws:iam::123456789012:instance-profile/EC2role"
                }
            ],
            "RoleName": "EC2role",
            "Path": "/",
            "AttachedManagedPolicies": [
                {
                    "PolicyName": "AmazonS3FullAccess",
                    "PolicyArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
                },
                {
                    "PolicyName": "AmazonDynamoDBFullAccess",
                    "PolicyArn": "arn:aws:iam::aws:policy/
AmazonDynamoDBFullAccess"
                }
            ],
            "RoleLastUsed": {
                "Region": "us-west-2",
                "LastUsedDate": "2019-11-13T17:30:00Z"
            },
            "RolePolicyList": [],
            "Arn": "arn:aws:iam::123456789012:role/EC2role"
        }
    ],
    "GroupDetailList": [
        {
            "GroupId": "AIDA1234567890EXAMPLE",
            "AttachedManagedPolicies": {
                "PolicyName": "AdministratorAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
            },
            "GroupName": "Admins",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Admins",
            "CreateDate": "2013-10-14T18:32:24Z",
            "GroupPolicyList": []
        },
```

```
        {
            "GroupId": "AIDA1234567890EXAMPLE",
            "AttachedManagedPolicies": {
                "PolicyName": "PowerUserAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/PowerUserAccess"
            },
            "GroupName": "Dev",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Dev",
            "CreateDate": "2013-10-14T18:33:55Z",
            "GroupPolicyList": []
        },
        {
            "GroupId": "AIDA1234567890EXAMPLE",
            "AttachedManagedPolicies": [],
            "GroupName": "Finance",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Finance",
            "CreateDate": "2013-10-14T18:57:48Z",
            "GroupPolicyList": [
                {
                    "PolicyName": "policygen-201310141157",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Action": "aws-portal:*",
                                "Sid": "Stmt1381777017000",
                                "Resource": "*",
                                "Effect": "Allow"
                            }
                        ]
                    }
                }
            ]
        }
    ],
    "UserDetailList": [
        {
            "UserName": "Alice",
            "GroupList": [
                "Admins"
            ],
            "CreateDate": "2013-10-14T18:32:24Z",
```

```
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Alice"
        },
        {
            "UserName": "Bob",
            "GroupList": [
                "Admins"
            ],
            "CreateDate": "2013-10-14T18:32:25Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [
                {
                    "PolicyName": "DenyBillingAndIAMPolicy",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": {
                            "Effect": "Deny",
                            "Action": [
                                "aws-portal:*",
                                "iam:*"
                            ],
                            "Resource": "*"
                        }
                    }
                }
            ],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Bob"
        },
        {
            "UserName": "Charlie",
            "GroupList": [
                "Dev"
            ],
            "CreateDate": "2013-10-14T18:33:56Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Charlie"
```

```
        }
    ],
    "Policies": [
        {
            "PolicyName": "create-update-delete-set-managed-policies",
            "CreateDate": "2015-02-06T19:58:34Z",
            "AttachmentCount": 1,
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2015-02-06T19:58:34Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version": "2012-10-17",
                        "Statement": {
                            "Effect": "Allow",
                            "Action": [
                                "iam:CreatePolicy",
                                "iam:CreatePolicyVersion",
                                "iam:DeletePolicy",
                                "iam:DeletePolicyVersion",
                                "iam:GetPolicy",
                                "iam:GetPolicyVersion",
                                "iam:ListPolicies",
                                "iam:ListPolicyVersions",
                                "iam:SetDefaultPolicyVersion"
                            ],
                            "Resource": "*"
                        }
                    },
                    "IsDefaultVersion": true
                }
            ],
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:policy/create-update-delete-set-
managed-policies",
            "UpdateDate": "2015-02-06T19:58:34Z"
        },
        {
            "PolicyName": "S3-read-only-specific-bucket",
            "CreateDate": "2015-01-21T21:39:41Z",
            "AttachmentCount": 1,
```

```
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2015-01-21T21:39:41Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "s3:Get*",
                                    "s3:List*"
                                ],
                                "Resource": [
                                    "arn:aws:s3:::example-bucket",
                                    "arn:aws:s3:::example-bucket/*"
                                ]
                            }
                        ]
                    },
                    "IsDefaultVersion": true
                }
            ],
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:policy/S3-read-only-specific-
bucket",
            "UpdateDate": "2015-01-21T23:39:41Z"
        },
        {
            "PolicyName": "AmazonEC2FullAccess",
            "CreateDate": "2015-02-06T18:40:15Z",
            "AttachmentCount": 1,
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2014-10-30T20:59:46Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version": "2012-10-17",
```

```
                        "Statement": [
                            {
                                "Action": "ec2:*",
                                "Effect": "Allow",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "elasticloadbalancing:*",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "cloudwatch:*",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "autoscaling:*",
                                "Resource": "*"
                            }
                        ]
                    },
                    "IsDefaultVersion": true
                }
            ],
            "Path": "/",
            "Arn": "arn:aws:iam::aws:policy/AmazonEC2FullAccess",
            "UpdateDate": "2015-02-06T18:40:15Z"
        }
    ],
    "Marker": "EXAMPLEkakv9BCuUNFDtxWSyfzetYwEx2ADc8dnzfvERF5S6YMvXKx41t6gCl/
eeaCX3Jo94/bKqezEAg8TEVS99EKFLxm3jtbpl25FDWEXAMPLE",
    "IsTruncated": true
}
```

자세한 내용은 AWS IAM 사용 설명서의 AWS 보안 감사 지침을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetAccountAuthorizationDetails를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 AWS 계정의 ID에 대한 권한 부여 세부 정보를 가져오고 사용자, 그룹 및 역할을 포함하여 반환된 객체의 요소 목록을 표시합니다. 예를 들어, **UserDetailList** 속성은 사용자에 대한 세부 정보를 표시합니다. **RoleDetailList** 및 **GroupDetailList** 속성에서 유사한 정보를 확인할 수 있습니다.

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

출력:

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

출력:

```
Arn             : arn:aws:iam::123456789012:user/Administrator
CreateDate      : 10/16/2014 9:03:09 AM
GroupList       : {Administrators}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE1
UserName        : Administrator
UserPolicyList : {}

Arn             : arn:aws:iam::123456789012:user/Bob
CreateDate      : 4/6/2015 12:54:42 PM
GroupList       : {Developers}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE2
UserName        : bab
UserPolicyList : {}

Arn             : arn:aws:iam::123456789012:user/BackupToS3
CreateDate      : 1/27/2015 10:15:08 AM
```

```
GroupList      : {Backup}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE3
UserName       : BackupToS3
UserPolicyList : {BackupServicePermissionsToS3Buckets}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetAccountAuthorizationDetails를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report,
 such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 GetAccountAuthorizationDetails를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetAccountPasswordPolicy** 사용

다음 코드 예제는 GetAccountPasswordPolicy의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetAccountPasswordPolicy를 참조하십시오.

CLI

AWS CLI

현재 계정 암호 정책 보기

다음 `get-account-password-policy` 명령은 현재 계정의 암호 정책에 대한 세부 정보를 표시합니다.

```
aws iam get-account-password-policy
```

출력:

```
{
    "PasswordPolicy": {
        "AllowUsersToChangePassword": false,
        "RequireLowercaseCharacters": false,
        "RequireUppercaseCharacters": false,
        "MinimumPasswordLength": 8,
        "RequireNumbers": true,
        "RequireSymbols": true
    }
}
```

계정에 대해 정의된 암호 정책이 없는 경우 명령은 `NoSuchEntity` 오류를 반환합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자의 계정 암호 정책 설정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetAccountPasswordPolicy를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy() (*types.PasswordPolicy,
 error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(context.TODO(),
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 GetAccountPasswordPolicy를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

계정 암호 정책을 가져옵니다.

```
import {
  GetAccountPasswordPolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const getAccountPasswordPolicy = async () => {
  const command = new GetAccountPasswordPolicyCommand({});

  const response = await client.send(command);
  console.log(response.PasswordPolicy);
  return response;
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetAccountPasswordPolicy를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function getAccountPasswordPolicy()
    {
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetAccountPasswordPolicy를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 현재 계정의 암호 정책에 대한 세부 정보를 반환합니다. 계정에 대해 정의된 암호 정책이 없는 경우 명령은 **NoSuchEntity** 오류를 반환합니다.

```
Get-IAMAccountPasswordPolicy
```

출력:

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetAccountPasswordPolicy를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def print_password_policy():
    """
    Prints the password policy for the account.
    """
```

```
    try:
        pw_policy = iam.AccountPasswordPolicy()
        print("Current account password policy:")
        print(
            f"\tallow_users_to_change_password:
{pw_policy.allow_users_to_change_password}"
        )
        print(f"\texpire_passwords: {pw_policy.expire_passwords}")
        print(f"\thard_expiry: {pw_policy.hard_expiry}")
        print(f"\tmax_password_age: {pw_policy.max_password_age}")
        print(f"\tminimum_password_length: {pw_policy.minimum_password_length}")
        print(f"\tpassword_reuse_prevention:
{pw_policy.password_reuse_prevention}")
        print(
            f"\trequire_lowercase_characters:
{pw_policy.require_lowercase_characters}"
        )
        print(f"\trequire_numbers: {pw_policy.require_numbers}")
        print(f"\trequire_symbols: {pw_policy.require_symbols}")
        print(
            f"\trequire_uppercase_characters:
{pw_policy.require_uppercase_characters}"
        )
        printed = True
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchEntity":
            print("The account does not have a password policy set.")
        else:
            logger.exception("Couldn't get account password policy.")
            raise
    else:
        return printed
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 GetAccountPasswordPolicy를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Class to manage IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger

  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "IAMPolicyManager"
  end

  # Retrieves and logs the account password policy
  def print_account_password_policy
    begin
      response = @iam_client.get_account_password_policy
      @logger.info("The account password policy is:
 #{response.password_policy.to_h}")
    rescue Aws::IAM::Errors::NoSuchEntity
      @logger.info("The account does not have a password policy.")
    rescue Aws::Errors::ServiceError => e
      @logger.error("Couldn't print the account password policy. Error: #{e.code}
 - #{e.message}")
      raise
    end
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetAccountPasswordPolicy를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn get_account_password_policy(
    client: &iamClient,
) -> Result<GetAccountPasswordPolicyOutput,
 SdkError<GetAccountPasswordPolicyError>> {
    let response = client.get_account_password_policy().send().await?;

    Ok(response)
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 GetAccountPasswordPolicy을 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **`GetAccountSummary`** 사용

다음 코드 예제는 `GetAccountSummary`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 계정 관리

CLI

AWS CLI

현재 계정의 IAM 엔터티 사용량 및 IAM 할당량에 대한 정보 가져오기

다음 `get-account-summary` 명령은 계정의 현재 IAM 엔터티 사용량과 현재 IAM 엔터티 할당량에 대한 정보를 반환합니다.

```
aws iam get-account-summary
```

출력:

```
{
    "SummaryMap": {
        "UsersQuota": 5000,
        "GroupsQuota": 100,
        "InstanceProfiles": 6,
        "SigningCertificatesPerUserQuota": 2,
        "AccountAccessKeysPresent": 0,
        "RolesQuota": 250,
        "RolePolicySizeQuota": 10240,
        "AccountSigningCertificatesPresent": 0,
        "Users": 27,
        "ServerCertificatesQuota": 20,
        "ServerCertificates": 0,
        "AssumeRolePolicySizeQuota": 2048,
        "Groups": 7,
        "MFADevicesInUse": 1,
        "Roles": 3,
        "AccountMFAEnabled": 1,
        "MFADevices": 3,
        "GroupsPerUserQuota": 10,
        "GroupPolicySizeQuota": 5120,
        "InstanceProfilesQuota": 100,
        "AccessKeysPerUserQuota": 2,
        "Providers": 0,
        "UserPolicySizeQuota": 2048
    }
}
```

엔터티 제한에 대한 자세한 내용은 AWS IAM 사용 설명서의 IAM 및 AWS STS 할당량을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetAccountSummary를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 AWS 계정의 현재 IAM 엔터티 사용량과 현재 IAM 엔터티 할당량에 대한 정보를 반환합니다.

```
Get-IAMAccountSummary
```

출력:

```
Key                                     Value
Users                                   7
GroupPolicySizeQuota                    5120
PolicyVersionsInUseQuota                10000
ServerCertificatesQuota                 20
AccountSigningCertificatesPresent       0
AccountAccessKeysPresent                0
Groups                                  3
UsersQuota                              5000
RolePolicySizeQuota                     10240
UserPolicySizeQuota                     2048
GroupsPerUserQuota                      10
AssumeRolePolicySizeQuota               2048
AttachedPoliciesPerGroupQuota           2
Roles                                   9
VersionsPerPolicyQuota                  5
GroupsQuota                             100
PolicySizeQuota                         5120
Policies                                5
RolesQuota                              250
ServerCertificates                      0
AttachedPoliciesPerRoleQuota            2
MFADevicesInUse                         2
PoliciesQuota                           1000
AccountMFAEnabled                       1
Providers                               2
InstanceProfilesQuota                   100
MFADevices                              4
AccessKeysPerUserQuota                  2
AttachedPoliciesPerUserQuota            2
SigningCertificatesPerUserQuota         2
```

```
PolicyVersionsInUse                     4
InstanceProfiles                        1
...
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetAccountSummary를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 GetAccountSummary를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetContextKeysForCustomPolicy** 사용

다음 코드 예제는 GetContextKeysForCustomPolicy의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: 명령줄에서 파라미터로 제공된 하나 이상의 사용자 지정 JSON 정책에서 참조하는 컨텍스트 키 나열

다음 get-context-keys-for-custom-policy 명령은 제공된 각 정책을 구문 분석하고 해당 정책에서 사용되는 컨텍스트 키를 나열합니다. 이 명령을 사용하여 정책 시뮬레이터 명령 simulate-custom-policy 및 simulate-custom-policy를 성공적으로 사용하기 위해 제공해야 하는 컨텍스트 키 값을 식별합니다. get-context-keys-for-custom-policy 명령을 사용하여 IAM 사용자 또는 역할과 관련된 모든 정책에서 사용되는 컨텍스트 키 목록을 검색할 수도 있습니다. file://로 시작하는 파라미터 값은 명령에 파일을 읽고 파일 이름 자체 대신 파라미터 값으로 내용을 사용하도록 지시합니다.

```
aws iam get-context-keys-for-custom-policy \
    --policy-input-list '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/${aws:username}","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
```

출력:

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
}
```

예제 2: 파일 입력으로 제공된 하나 이상의 사용자 지정 JSON 정책에서 참조하는 컨텍스트 키 나열

다음 get-context-keys-for-custom-policy 명령은 정책이 파라미터 대신 파일로 제공된다는 점을 제외하면 이전 예와 동일합니다. 명령에는 JSON 구조 목록이 아닌 JSON 문자열 목록이 필요하므로 파일은 하나로 축소할 수 있지만 다음과 같이 구성되어야 합니다.

```
[
    "Policy1",
    "Policy2"
]
```

예를 들어 이전 예제의 정책이 포함된 파일은 다음과 같아야 합니다. 정책 문자열 내에 포함된 각 큰따옴표는 앞에 백슬래시를 붙여 이스케이프 처리해야 합니다.

```
[ "{\"Version\": \"2012-10-17\", \"Statement\": {\"Effect\": \"Allow
\", \"Action\": \"dynamodb:*\", \"Resource\": \"arn:aws:dynamodb:us-
west-2:128716708097:table/${aws:username}\", \"Condition\": {\"DateGreaterThan\":
 {\"aws:CurrentTime\": \"2015-08-16T12:00:00Z\"}}}}" ]
```

그런 다음 이 파일을 다음 명령에 제출할 수 있습니다.

```
aws iam get-context-keys-for-custom-policy \
    --policy-input-list file://policyfile.json
```

출력:

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 정책 시뮬레이터의 사용(AWS CLI 및 AWS API)을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetContextKeysForCustomPolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 제공된 정책 json에 있는 모든 컨텍스트 키를 가져옵니다. 여러 정책을 제공하려면 쉼표로 구분된 값 목록으로 제공합니다.

```
$policy1 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
```

```
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetContextKeysForCustomPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetContextKeysForPrincipalPolicy** 사용

다음 코드 예제는 GetContextKeysForPrincipalPolicy의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 보안 주체와 연결된 모든 정책에서 참조하는 컨텍스트 키 나열

다음 get-context-keys-for-principal-policy 명령은 사용자 saanvi 및 해당 사용자가 속한 그룹에 연결된 모든 정책을 검색합니다. 그런 다음 각 정책을 분석하여 해당 정책에서 사용하는 컨텍스트 키를 나열합니다. 이 명령을 사용하여 simulate-custom-policy 및 simulate-principal-policy 명령을 성공적으로 사용하기 위해 제공해야 하는 컨텍스트 키 값을 식별합니다. get-context-keys-for-custom-policy 명령을 사용하여 임의의 JSON 정책에서 사용하는 컨텍스트 키 목록을 검색할 수도 있습니다.

```
aws iam get-context-keys-for-principal-policy \
   --policy-source-arn arn:aws:iam::123456789012:user/saanvi
```

출력:

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
```

```
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 정책 시뮬레이터의 사용(AWS CLI 및 AWS API)을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetContextKeysForPrincipalPolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 제공된 정책 json에 있는 모든 컨텍스트 키와 IAM 엔티티(사용자/역할 등)에 연결된 정책을 가져옵니다. -PolicyInputList의 경우 여러 값 목록을 쉼표로 구분된 값으로 제공할 수 있습니다.

```
$policy1 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetContextKeysForPrincipalPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetCredentialReport** 사용

다음 코드 예제는 GetCredentialReport의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 계정 관리

CLI

AWS CLI

보안 인증 보고서 가져오기

이 예제에서는 반환된 보고서를 열고 파이프라인에 텍스트 라인 배열로 출력합니다.

```
aws iam get-credential-report
```

출력:

```
{
    "GeneratedTime":  "2015-06-17T19:11:50Z",
    "ReportFormat": "text/csv"
}
```

자세한 내용은 AWS IAM 사용 설명서의 AWS 계정의 보안 인증 보고서 가져오기를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetCredentialReport를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 반환된 보고서를 열고 파이프라인에 텍스트 라인 배열로 출력합니다. 줄 1은 쉼표로 구분된 열 이름이 있는 헤더입니다. 그 후의 각 행은 한 사용자에 대한 세부 정보 행이며 각 필드는 쉼표로 구분됩니다. 보고서를 보려면 먼저 **Request-IAMCredentialReport** cmdlet을 사용하여 보고서를 생성해야 합니다. 보고서를 단일 문자열로 검색하려면 **-AsTextArray** 대신 **-Raw**를 사용합니다. **-AsTextArray** 스위치에는 **-SplitLines** 별칭도 허용됩니다. 출력의 전체 열 목록은 서비스 API 참조를 참조하세요. **-AsTextArray** 또는 **-SplitLines**를 사용하지 않는 경우 .NET **StreamReader** 클래스를 사용하여 **.Content** 속성에서 텍스트를 추출해야 합니다.

```
Request-IAMCredentialReport
```

출력:

```
Description                                                    State
```

```
-----------                                                              -----
No report exists. Starting a new report generation task                  STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

출력:

```
 user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,pa
 root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015
A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/
Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:0
A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/
A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/
A,N/A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetCredentialReport를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def get_credential_report():
    """
    Gets the most recently generated credentials report about the current
 account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
```

```
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 GetCredentialReport를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetGroup** 사용

다음 코드 예제는 GetGroup의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 그룹 가져오기

이 예제는 IAM 그룹 Admins에 대한 세부 정보를 반환합니다.

```
aws iam get-group \
    --group-name Admins
```

출력:

```
{
    "Group": {
        "Path": "/",
        "CreateDate": "2015-06-16T19:41:48Z",
        "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/Admins",
        "GroupName": "Admins"
    },
    "Users": []
```

```
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM ID(사용자, 그룹 및 역할)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetGroup을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 그룹에 속한 모든 IAM 사용자의 모음을 포함하여 IAM 그룹 **Testers**에 대한 세부 정보를 반환합니다.

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

출력:

```
Group                                    IsTruncated           Marker
     Users
-----                                    -----------           ------
     -----
Amazon.IdentityManagement.Model.Group    False
     {Theresa, David}
```

```
$results.Group
```

출력:

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
Path       : /
```

```
$results.Users
```

출력:

```
Arn              : arn:aws:iam::123456789012:user/Theresa
```

```
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : 4OSVDDJJTF4XEEXAMPLE2
UserName         : Theresa

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetGroup을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetGroupPolicy** 사용

다음 코드 예제는 GetGroupPolicy의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 그룹에 연결된 정책에 대한 정보 가져오기

다음 get-group-policy 명령은 Test-Group이라는 그룹에 연결된 지정된 정책에 대한 정보를 가져옵니다.

```
aws iam get-group-policy \
    --group-name Test-Group \
    --policy-name S3-ReadOnly-Policy
```

출력:

```
{
    "GroupName": "Test-Group",
    "PolicyDocument": {
        "Statement": [
```

```
            {
                "Action": [
                    "s3:Get*",
                    "s3:List*"
                ],
                "Resource": "*",
                "Effect": "Allow"
            }
        ]
    },
    "PolicyName": "S3-ReadOnly-Policy"
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 정책 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetGroupPolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **Testers** 그룹에 대해 **PowerUserAccess-Testers**라는 포함된 인라인 정책에 대한 세부 정보를 반환합니다. **PolicyDocument** 속성은 URL로 인코딩됩니다. 이 예제에서는 **UrlDecode** .NET 메서드를 사용하여 디코딩됩니다.

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-
Testers
$results
```

출력:

```
GroupName     PolicyDocument
 PolicyName
---------     --------------
 ----------
Testers       %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
 PowerUserAccess-Testers

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "NotAction": "iam:*",
      "Resource": "*"
    }
  ]
}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetGroupPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetInstanceProfile** 사용

다음 코드 예제는 GetInstanceProfile의 사용 방법을 보여 줍니다.

CLI

AWS CLI

인스턴스 프로파일 정보 가져오기

다음 get-instance-profile 명령은 이름이 ExampleInstanceProfile인 인스턴스 프로파일에 대한 정보를 가져옵니다.

```
aws iam get-instance-profile \
    --instance-profile-name ExampleInstanceProfile
```

출력:

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AID2MAB8DPLSRHEXAMPLE",
        "Roles": [
            {
                "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
                "RoleId": "AIDGPMS9RO4H3FEXAMPLE",
                "CreateDate": "2013-01-09T06:33:26Z",
                "RoleName": "Test-Role",
                "Path": "/",
```

```
                "Arn": "arn:aws:iam::336924118301:role/Test-Role"
            }
        ],
        "CreateDate": "2013-06-12T23:52:02Z",
        "InstanceProfileName": "ExampleInstanceProfile",
        "Path": "/",
        "Arn": "arn:aws:iam::336924118301:instance-profile/
ExampleInstanceProfile"
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 인스턴스 프로파일 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetInstanceProfile을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 현재 AWS 계정에 정의된 **ec2instancerole**이라는 인스턴스 프로파일의 세부 정보를 반환합니다.

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

출력:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetInstanceProfile을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetLoginProfile** 사용

다음 코드 예제는 GetLoginProfile의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자의 암호 정보 가져오기

다음 `get-login-profile` 명령은 이름이 Bob인 IAM 사용자의 암호에 대한 정보를 가져옵니다.

```
aws iam get-login-profile \
    --user-name Bob
```

출력:

```
{
    "LoginProfile": {
        "UserName": "Bob",
        "CreateDate": "2012-09-21T23:03:39Z"
    }
}
```

`get-login-profile` 명령을 사용하여 IAM 사용자에게 암호가 있는지 확인할 수 있습니다. 사용자에 대해 정의된 암호가 없는 경우 명령은 `NoSuchEntity` 오류를 반환합니다.

이 명령을 사용해 암호를 볼 수는 없습니다. 암호를 잊어버린 경우 사용자의 암호를 재설정(`update-login-profile`)할 수 있습니다. 또는 사용자의 로그인 프로파일을 삭제(`delete-login-profile`)한 다음 새 프로파일을 생성(`create-login-profile`)할 수 있습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 암호 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetLoginProfile을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 암호 생성 날짜와 IAM 사용자 **David**에 대해 암호 재설정이 필요한지 여부를 반환합니다.

```
Get-IAMLoginProfile -UserName David
```

출력:

```
CreateDate                  PasswordResetRequired                UserName
----------                  ---------------------                --------
12/10/2014 3:39:44 PM       False                                David
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetLoginProfile을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetOpenIdConnectProvider** 사용

다음 코드 예제는 GetOpenIdConnectProvider의 사용 방법을 보여 줍니다.

CLI

AWS CLI

지정된 OpenID Connect 제공업체에 대한 정보 반환

이 예제는 ARN이 arn:aws:iam::123456789012:oidc-provider/server.example.com인 OpenID Connect 제공업체에 대한 세부 정보를 반환합니다.

```
aws iam get-open-id-connect-provider \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
server.example.com
```

출력:

```
{
    "Url": "server.example.com"
        "CreateDate": "2015-06-16T19:41:48Z",
        "ThumbprintList": [
        "12345abcdefghijk67890lmnopqrst987example"
        ],
        "ClientIDList": [
        "example-application-ID"
        ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM에서 OIDC(OpenID Connect) ID 제공업체 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetOpenIdConnectProvider를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:oidc-provider/accounts.google.com**인 OpenID Connect 제공업체에 대한 세부 정보를 반환합니다. **ClientIDList** 속성은 이 제공업체에 대해 정의된 모든 클라이언트 ID를 포함하는 컬렉션입니다.

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

출력:

```
ClientIDList         CreateDate                ThumbprintList
           Url
------------         ----------                --------------
           ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetOpenIdConnectProvider를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetPolicy** 사용

다음 코드 예제는 GetPolicy의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- IAM 정책 빌더 API 작업

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetPolicy를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::getPolicy(const Aws::String &policyArn,
```

```
                            const Aws::Client::ClientConfiguration &clientConfig)
 {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetPolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.GetPolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error getting policy " << policyArn << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &policy = outcome.GetResult().GetPolicy();
        std::cout << "Name: " << policy.GetPolicyName() << std::endl <<
                  "ID: " << policy.GetPolicyId() << std::endl << "Arn: " <<
                  policy.GetArn() << std::endl << "Description: " <<
                  policy.GetDescription() << std::endl << "CreateDate: " <<

 policy.GetCreateDate().ToGmtString(Aws::Utils::DateFormat::ISO_8601)
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 GetPolicy를 참조하십시오.

CLI

AWS CLI

지정된 관리형 정책에 대한 정보 검색

이 예제는 ARN이 `arn:aws:iam::123456789012:policy/MySamplePolicy`인 관리형 정책에 대한 세부 정보를 반환합니다.

```
aws iam get-policy \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

출력:

```
{
    "Policy": {
        "PolicyName": "MySamplePolicy",
        "CreateDate": "2015-06-17T19:23;32Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "Z27SI6FQMGNQ2EXAMPLE1",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:policy/MySamplePolicy",
        "UpdateDate": "2015-06-17T19:23:32Z"
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetPolicy를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(policyArn string) (*types.Policy, error) {
```

```
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(context.TODO(), &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 GetPolicy를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

정책을 가져옵니다.

```
import { GetPolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const getPolicy = (policyArn) => {
  const command = new GetPolicyCommand({
    PolicyArn: policyArn,
  });

  return client.send(command);
```

```
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetPolicy를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  PolicyArn: "arn:aws:iam::aws:policy/AWSLambdaExecute",
};

iam.getPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Policy.Description);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetPolicy를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 GetPolicy를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function getPolicy($policyArn)
    {
```

```
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetPolicy를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/MySamplePolicy**인 관리형 정책에 대한 세부 정보를 반환합니다.

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

출력:

```
Arn              : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 2/6/2015 10:40:08 AM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetPolicy를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 GetPolicy를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
```

```
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetPolicy를 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public func getPolicy(arn: String) async throws -> IAMClientTypes.Policy {
        let input = GetPolicyInput(
            policyArn: arn
        )
        do {
            let output = try await client.getPolicy(input: input)
            guard let policy = output.policy else {
                throw ServiceHandlerError.noSuchPolicy
```

```
            }
            return policy
        } catch {
            throw error
        }
    }
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 GetPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetPolicyVersion** 사용

다음 코드 예제는 GetPolicyVersion의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 정책 관리
- IAM 정책 빌더 API 작업

CLI

AWS CLI

지정된 관리형 정책의 지정된 버전에 대한 정보 검색

이 예제는 ARN이 arn:aws:iam::123456789012:policy/MyManagedPolicy인 정책의 v2 버전에 대한 정책 문서를 반환합니다.

```
aws iam get-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

출력:

```
{
    "PolicyVersion": {
        "Document": {
```

```
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": "iam:*",
                    "Resource": "*"
                }
            ]
        },
        "VersionId": "v2",
        "IsDefaultVersion": true,
        "CreateDate": "2023-04-11T00:22:54+00:00"
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetPolicyVersion을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/MyManagedPolicy**인 정책의 **v2** 버전에 대한 정책 문서를 반환합니다. **Document** 속성의 정책 문서는 URL로 인코딩되며 이 예제에서는 **UrlDecode** .NET 메서드를 사용하여 디코딩됩니다.

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

출력:

```
CreateDate             Document
 IsDefaultVersion     VersionId
----------             --------
 ----------------     ---------
2/12/2015 9:39:53 AM   %7B%0A%20%20%22Version%22%3A%20%222012-10...    True
           v2

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
```

```
$policy
{
  "Version": "2012-10-17",
  "Statement":
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetPolicyVersion을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
```

```
        return policy_statement
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 GetPolicyVersion를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetRole** 사용

다음 코드 예제는 GetRole의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Get information about an IAM role.
/// </summary>
/// <param name="roleName">The name of the IAM role to retrieve information
/// for.</param>
/// <returns>The IAM role that was retrieved.</returns>
public async Task<Role> GetRoleAsync(string roleName)
{
    var response = await _IAMService.GetRoleAsync(new GetRoleRequest
    {
        RoleName = roleName,
    });

    return response.Role;
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetRole을 참조하십시오.

CLI

AWS CLI

IAM 역할 정보 가져오기

다음 `get-role` 명령은 이름이 `Test-Role`인 역할에 대한 정보를 가져옵니다.

```
aws iam get-role \
    --role-name Test-Role
```

출력:

```
{
    "Role": {
        "Description": "Test Role",
        "AssumeRolePolicyDocument":"<URL-encoded-JSON>",
        "MaxSessionDuration": 3600,
        "RoleId": "AROA1234567890EXAMPLE",
        "CreateDate": "2019-11-13T16:45:56Z",
        "RoleName": "Test-Role",
        "Path": "/",
        "RoleLastUsed": {
            "Region": "us-east-1",
            "LastUsedDate": "2019-11-13T17:14:00Z"
        },
        "Arn": "arn:aws:iam::123456789012:role/Test-Role"
    }
}
```

이 명령은 역할에 연결된 신뢰 정책을 표시합니다. 역할에 연결된 권한 정책을 나열하려면 `list-role-policies` 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 역할 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetRole을 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(roleName string) (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(context.TODO(),
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 GetRole을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

역할을 가져옵니다.

```
import { GetRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const getRole = (roleName) => {
  const command = new GetRoleCommand({
    RoleName: roleName,
  });

  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetRole을 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
            return $this->iamClient->getRole(['RoleName' => $roleName]);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 GetRole을 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **lamda_exec_role**의 세부 정보를 반환합니다. 여기에는 이 역할을 수임할 수 있는 사용자를 지정하는 신뢰 정책 문서가 포함됩니다. 정책 문서는 URL로 인코딩되며 .NET **UrlDecode** 메서드를 사용하여 디코딩할 수 있습니다. 이 예제에서는 원래 정책에 업로드되기 전에 모든 공백이 제거됩니다. 역할을 수임한 사람이 수행할 수 있는 작업을 결정하는 권한 정책 문서를 보려면 인라인 정책에는 **Get-IAMRolePolicy**를 사용하고 연결된 관리형 정책에는 **Get-IAMPolicyVersion**을 사용합니다.

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

출력:

```
Arn                       : arn:aws:iam::123456789012:role/lambda_exec_role
AssumeRolePolicyDocument : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement
%22%3A%5B%7B%22Sid%22
                           %3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal
%22%3A%7B%22Service
                           %22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A
%22sts%3AAssumeRole
                           %22%7D%5D%7D
CreateDate                : 4/2/2015 9:16:11 AM
Path                      : /
RoleId                    : 2YBIKAIBHNKB4EXAMPLE1
```

```
RoleName                : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

출력:

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

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetRole을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def get_role(role_name):
    """
    Gets a role by name.

    :param role_name: The name of the role to retrieve.
    :return: The specified role.
    """
    try:
        role = iam.Role(role_name)
        role.load()  # calls GetRole to load attributes
        logger.info("Got role with arn %s.", role.arn)
    except ClientError:
        logger.exception("Couldn't get role named %s.", role_name)
        raise
    else:
        return role
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 GetRole를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
  role = @iam_client.get_role({
                               role_name: name,
                             }).role
  puts("Got data for role '#{role.role_name}'. Its ARN is '#{role.arn}'.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get data for role '#{name}' Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  role
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetRole을 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn get_role(
    client: &iamClient,
    role_name: String,
) -> Result<GetRoleOutput, SdkError<GetRoleError>> {
    let response = client.get_role().role_name(role_name).send().await?;
    Ok(response)
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 GetRole을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public func getRole(name: String) async throws -> IAMClientTypes.Role {
        let input = GetRoleInput(
```

```
            roleName: name
        )
        do {
            let output = try await client.getRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            return role
        } catch {
            throw error
        }
    }
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 GetRole을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetRolePolicy** 사용

다음 코드 예제는 GetRolePolicy의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 역할에 연결된 정책에 대한 정보 가져오기

다음 get-role-policy 명령은 Test-Role이라는 역할에 연결된 지정된 정책에 대한 정보를 가져옵니다.

```
aws iam get-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy
```

출력:

```
{
  "RoleName": "Test-Role",
  "PolicyDocument": {
      "Statement": [
```

```
            {
                "Action": [
                    "s3:ListBucket",
                    "s3:Put*",
                    "s3:Get*",
                    "s3:*MultipartUpload*"
                ],
                "Resource": "*",
                "Effect": "Allow",
                "Sid": "1"
            }
        ]
    }
    "PolicyName": "ExamplePolicy"
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 역할 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetRolePolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM 역할 **lamda_exec_role**에 포함된 **oneClick_lambda_exec_role_policy**라는 정책에 대한 권한 정책 문서를 반환합니다. 결과 정책 문서는 URL로 인코딩됩니다. 이 예제에서는 **UrlDecode** .NET 메서드를 사용하여 디코딩됩니다.

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

출력:

```
PolicyDocument                                                  PolicyName
              UserName
--------------                                                  ----------
              --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...
 oneClick_lambda_exec_role_policy     lambda_exec_role
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

출력:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ]
    }
  ]
}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetRolePolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **`GetSamlProvider`** 사용

다음 코드 예제는 `GetSamlProvider`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

SAML 제공업체 메타문서 검색

이 예제는 ARM이 arn:aws:iam::123456789012:saml-provider/SAMLADFS인 SAML 2.0 제공업체에 대한 세부 정보를 검색합니다. 응답에는 AWS SAML 제공업체 엔터티를 생성하기 위해 ID 제공업체로부터 받은 메타데이터 문서와 생성 및 만료 날짜가 포함됩니다.

```
aws iam get-saml-provider \
    --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

출력:

```
{
    "SAMLMetadataDocument": "...SAMLMetadataDocument-XML...",
    "CreateDate": "2017-03-06T22:29:46+00:00",
    "ValidUntil": "2117-03-06T22:29:46.433000+00:00",
    "Tags": [
        {
            "Key": "DeptID",
            "Value": "123456"
        },
        {
            "Key": "Department",
            "Value": "Accounting"
        }
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM SAML 자격 증명 공급자 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetSamlProvider를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARM이 arn:aws:iam::123456789012:saml-provider/SAMLADFS인 SAML 2.0 제공업체에 대한 세부 정보를 검색합니다. 응답에는 AWS SAML 제공업체 엔터티를 생성하기 위해 ID 제공업체로부터 받은 메타데이터 문서와 생성 및 만료 날짜가 포함됩니다.

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

출력:

```
CreateDate                  SAMLMetadataDocument
        ValidUntil
----------                  --------------------
        ----------
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-
example1...    12/23/2114 12:16:54 PM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetSamlProvider를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetServerCertificate** 사용

다음 코드 예제는 GetServerCertificate의 사용 방법을 보여 줍니다.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::getServerCertificate(const Aws::String &certificateName,
                                       const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    auto outcome = iam.GetServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error getting server certificate " << certificateName
 <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
```

```
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        const auto &certificate = outcome.GetResult().GetServerCertificate();
        std::cout << "Name: " <<

 certificate.GetServerCertificateMetadata().GetServerCertificateName()
                  << std::endl << "Body: " << certificate.GetCertificateBody() <<
                  std::endl << "Chain: " << certificate.GetCertificateChain() <<
                  std::endl;
    }

    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 GetServerCertificate를 참조하십시오.

CLI

AWS CLI

AWS 계정의 서버 인증서에 대한 세부 정보 가져오기

다음 `get-server-certificate` 명령은 AWS 계정의 지정된 서버 인증서에 대한 모든 세부 정보를 검색합니다.

```
aws iam get-server-certificate \
    --server-certificate-name myUpdatedServerCertificate
```

출력:

```
{
    "ServerCertificate": {
        "ServerCertificateMetadata": {
            "Path": "/",
            "ServerCertificateName": "myUpdatedServerCertificate",
```

```
            "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/
myUpdatedServerCertificate",
            "UploadDate": "2019-04-22T21:13:44+00:00",
            "Expiration": "2019-10-15T22:23:16+00:00"
        },
        "CertificateBody": "-----BEGIN CERTIFICATE-----
            MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
            VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
            b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
            BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
            MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
            VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
            b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
            YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
            21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
            rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
            Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
            nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
            FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
            NYiytVbZPQUQ5Yaxu2jXnimvrszlaEXAMPLE=-----END CERTIFICATE-----",
        "CertificateChain": "-----BEGIN CERTIFICATE-----\nMIICiTCCAfICCQD6md
            7oRw0uXOjANBgkqhkiG9w0BAqQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgT
            AldBMRAwDgYDVQQHEwdTZWF0drGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAs
            TC0lBTSBDb25zb2xlMRIwEAYDVsQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQ
            jb20wHhcNMTEwNDI1MjA0NTIxWhtcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
            MCVVMxCzAJBgNVBAgTAldBMRAwDgsYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
            WF6b24xFDASBgNVBAsTC0lBTSBDb2d5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
            HzAdBgkqhkiG9w0BCQEWEG5vb25lQGfFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
            BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIgWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
            k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8mh9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
            ITxOUSQv7c7ugFFDzQGBzZswY6786m86gjpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
            AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCku4nUhVVxYUntneD9+h8Mg9q6q+auN
            KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FlkbFFBjvSfpJIlJ00zbhNYS5f6Guo
            EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjS;TbNYiytVbZPQUQ5Yaxu2jXnimvw
            3rrszlaEWEG5vb25lQGFtsYXpvbiEXAMPLE=\n-----END CERTIFICATE-----"
    }
}
```

AWS 계정에서 사용 가능한 서버 인증서를 나열하려면 `list-server-certificates` 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM에서 서버 인증서 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetServerCertificate를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

서버 인증서를 가져옵니다.

```
import { GetServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
 * @returns
 */
export const getServerCertificate = async (certName) => {
  const command = new GetServerCertificateCommand({
    ServerCertificateName: certName,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetServerCertificate를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetServerCertificate를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **MyServerCertificate**라는 서버 인증서에 대한 세부 정보를 검색합니다. **CertificateBody** 및 **ServerCertificateMetadata** 속성에서 인증서 세부 정보를 찾을 수 있습니다.

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
```

```
$result | format-list
```

출력:

```
CertificateBody           : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt

 YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                           21uUSfwfEvySWtC2XADZ4nB
+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                           rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                           nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                           NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                           -----END CERTIFICATE-----
CertificateChain          :
ServerCertificateMetadata :
 Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

출력:

```
Arn                  : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration           : 1/14/2018 9:52:36 AM
Path                 : /Org1/Org2/
ServerCertificateId  : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate           : 4/21/2015 11:14:16 AM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetServerCertificate를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetServiceLastAccessedDetails** 사용

다음 코드 예제는 GetServiceLastAccessedDetails의 사용 방법을 보여 줍니다.

CLI

AWS CLI

서비스 액세스 보고서 검색

다음 get-service-last-accessed-details 예제는 IAM 엔터티가 액세스한 서비스를 나열하는 이전에 생성된 보고서를 검색합니다. 보고서를 생성하려면 generate-service-last-accessed-details 명령을 사용합니다.

```
aws iam get-service-last-accessed-details \
    --job-id 2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc
```

출력:

```
{
    "JobStatus": "COMPLETED",
    "JobCreationDate": "2019-10-01T03:50:35.929Z",
    "ServicesLastAccessed": [
        ...
        {
            "ServiceName": "AWS Lambda",
```

```
            "LastAuthenticated": "2019-09-30T23:02:00Z",
            "ServiceNamespace": "lambda",
            "LastAuthenticatedEntity": "arn:aws:iam::123456789012:user/admin",
            "TotalAuthenticatedEntities": 6
        },
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetServiceLastAccessedDetails를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예에서는 요청 직접 호출과 연결된 IAM 엔터티(사용자, 그룹, 역할 또는 정책)가 마지막으로 액세스한 서비스에 대한 세부 정보를 제공합니다.

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

출력:

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetServiceLastAccessedDetails를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetServiceLastAccessedDetailsWithEntities** 사용

다음 코드 예제는 GetServiceLastAccessedDetailsWithEntities의 사용 방법을 보여 줍니다.

CLI

AWS CLI

서비스 세부 정보가 포함된 서비스 액세스 보고서 검색

다음 `get-service-last-accessed-details-with-entities` 예제는 지정된 서비스에 액세스한 IAM 사용자와 기타 엔터티에 대한 세부 정보가 포함된 보고서를 검색합니다. 보고서를 생성하려면 `generate-service-last-accessed-details` 명령을 사용합니다. 네임스페이스로 액세스하는 서비스 목록을 가져오려면 `get-service-last-accessed-details`를 사용합니다.

```
aws iam get-service-last-accessed-details-with-entities \
    --job-id 78b6c2ba-d09e-6xmp-7039-ecde30b26916 \
    --service-namespace lambda
```

출력:

```
{
    "JobStatus": "COMPLETED",
    "JobCreationDate": "2019-10-01T03:55:41.756Z",
    "JobCompletionDate": "2019-10-01T03:55:42.533Z",
    "EntityDetailsList": [
        {
            "EntityInfo": {
                "Arn": "arn:aws:iam::123456789012:user/admin",
                "Name": "admin",
                "Type": "USER",
                "Id": "AIDAIO2XMPLENQEXAMPLE",
                "Path": "/"
            },
            "LastAuthenticated": "2019-09-30T23:02:00Z"
        },
        {
            "EntityInfo": {
                "Arn": "arn:aws:iam::123456789012:user/developer",
                "Name": "developer",
                "Type": "USER",
                "Id": "AIDAIBEYXMPL2YEXAMPLE",
                "Path": "/"
            },
            "LastAuthenticated": "2019-09-16T19:34:00Z"
```

```
        }
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetServiceLastAccessedDetailsWithEntities를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 해당 IAM 엔터티의 요청에 있는 서비스에 대해 마지막으로 액세스한 타임스탬프를 제공합니다.

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

출력:

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}
Error             :
IsTruncated       : False
JobCompletionDate : 12/29/19 11:19:31 AM
JobCreationDate   : 12/29/19 11:19:31 AM
JobStatus         : COMPLETED
Marker            :
```

```
$results.EntityDetailsList
```

출력:

```
EntityInfo                                  LastAuthenticated
----------                                  -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

출력:

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetServiceLastAccessedDetailsWithEntities를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetServiceLinkedRoleDeletionStatus** 사용

다음 코드 예제는 GetServiceLinkedRoleDeletionStatus의 사용 방법을 보여 줍니다.

CLI

AWS CLI

서비스 연결 역할 삭제 요청 상태 확인

다음 get-service-linked-role-deletion-status 예제에서는 이전 서비스 연결 역할 삭제 요청의 상태를 표시합니다. 삭제 작업은 비동기식으로 이루어집니다. 요청을 하면 이 명령의 파라미터로 제공하는 DeletionTaskId 값을 가져옵니다.

```
aws iam get-service-linked-role-deletion-status \
    --deletion-task-id task/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE
```

출력:

```
{
"Status": "SUCCEEDED"
}
```

자세한 내용은 AWS IAM 사용 설명서의 서비스 연결 역할 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetServiceLinkedRoleDeletionStatus를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import {
  GetServiceLinkedRoleDeletionStatusCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} deletionTaskId
 */
export const getServiceLinkedRoleDeletionStatus = (deletionTaskId) => {
  const command = new GetServiceLinkedRoleDeletionStatusCommand({
    DeletionTaskId: deletionTaskId,
  });

  return client.send(command);
};
```

- API에 대한 세부 정보는 AWS SDK for JavaScript API 참조의 GetServiceLinkedRoleDeletionStatus를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **`GetUser`** 사용

다음 코드 예제는 `GetUser`의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
        return response.User;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 GetUser을 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
```

```
########################################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

########################################################################################
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
########################################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}
```

- API 세부 정보는 AWS CLI 명령 참조의 GetUser를 참조하십시오.

CLI

AWS CLI

IAM 사용자 정보 가져오기

다음 `get-user` 명령은 이름이 `Paulo`인 IAM 사용자에 대한 정보를 가져옵니다.

```
aws iam get-user \
    --user-name Paulo
```

출력:

```
{
    "User": {
        "UserName": "Paulo",
        "Path": "/",
        "CreateDate": "2019-09-21T23:03:13Z",
        "UserId": "AIDA123456789EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:user/Paulo"
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetUser를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(userName string) (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(context.TODO(), &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
  user = result.User
 }
 return user, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 GetUser을 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **David**라는 사용자에 대한 세부 정보를 검색합니다.

```
Get-IAMUser -UserName David
```

출력:

```
Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE1
UserName         : David
```

예제 2: 이 예제는 현재 로그인한 IAM 사용자에 대한 세부 정보를 검색합니다.

```
Get-IAMUser
```

출력:

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path             : /
UserId           : 7K3GJEANSKZF2EXAMPLE2
UserName         : Bob
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetUser를 참조하세요.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  # Retrieves a user's details
  #
  # @param user_name [String] The name of the user to retrieve
  # @return [Aws::IAM::Types::User, nil] The user object if found, or nil if an
error occurred
  def get_user(user_name)
```

```
    response = @iam_client.get_user(user_name: user_name)
    response.user
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("User '#{user_name}' not found.")
    nil
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error retrieving user '#{user_name}': #{e.message}")
    nil
  end
```

- API에 대한 세부 정보는 AWS SDK for Ruby API Reference(Go API 참조)의 GetUser를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetUserPolicy** 사용

다음 코드 예제는 GetUserPolicy의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자에 대한 정책 세부 정보 나열

다음 get-user-policy 명령은 이름이 Bob인 IAM 사용자에게 연결된 지정된 정책의 세부 정보를 나열합니다.

```
aws iam get-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy
```

출력:

```
{
    "UserName": "Bob",
    "PolicyName": "ExamplePolicy",
    "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
```

```
            {
                "Action": "*",
                "Resource": "*",
                "Effect": "Allow"
            }
        ]
    }
}
```

IAM 사용자의 정책 목록을 가져오려면 `list-user-policies` 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetUserPolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **David**라는 IAM 사용자에게 포함된 **Davids_IAM_Admin_Policy**라는 인라인 정책의 세부 정보를 검색합니다. 정책 문서는 URL로 인코딩됩니다.

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

출력:

```
PolicyDocument                                                  PolicyName
       UserName
--------------                                                  ----------
       --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...    Davids_IAM_Admin_Policy
       David

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:*"
```

```
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetUserPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListAccessKeys** 사용

다음 코드 예제는 ListAccessKeys의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 액세스 키 관리

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}
```

```
##############################################################################
# function iam_list_access_keys
#
# This function lists the access keys for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#
# Returns:
#       access_key_ids
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_list_access_keys() {

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_access_keys"
    echo "Lists the AWS Identity and Access Management (IAM) access key IDs for
 the specified user."
    echo "  -u user_name   The name of the IAM user."
    echo ""
  }

  local user_name response
  local option OPTARG # Required to use getopts command in a function.
  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1
```

```
  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam list-access-keys \
    --user-name "$user_name" \
    --output text \
    --query 'AccessKeyMetadata[].AccessKeyId')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-access-keys operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API 세부 정보는 AWS CLI 명령 참조의 ListAccessKeys를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::listAccessKeys(const Aws::String &userName,
                                 const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccessKeysRequest request;
```

```
    request.SetUserName(userName);

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccessKeys(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list access keys for user " << userName
                      << ": " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "UserName" <<
                      std::setw(30) << "KeyID" << std::setw(20) << "Status" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &keys = outcome.GetResult().GetAccessKeyMetadata();
        const Aws::String DATE_FORMAT = "%Y-%m-%d";

        for (const auto &key: keys) {
            Aws::String statusString =
                    Aws::IAM::Model::StatusTypeMapper::GetNameForStatusType(
                            key.GetStatus());
            std::cout << std::left << std::setw(32) << key.GetUserName() <<
                      std::setw(30) << key.GetAccessKeyId() << std::setw(20) <<
                      statusString << std::setw(20) <<
                      key.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
 std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ListAccessKeys를 참조하세요.

CLI

AWS CLI

IAM 사용자의 액세스 키 ID 나열

다음 `list-access-keys` 명령은 이름이 Bob인 IAM 사용자의 액세스 키 ID를 나열합니다.

```
aws iam list-access-keys \
    --user-name Bob
```

출력:

```
{
    "AccessKeyMetadata": [
        {
            "UserName": "Bob",
            "Status": "Active",
            "CreateDate": "2013-06-04T18:17:34Z",
            "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
        },
        {
            "UserName": "Bob",
            "Status": "Inactive",
            "CreateDate": "2013-06-06T20:42:26Z",
            "AccessKeyId": "AKIAI44QH8DHBEXAMPLE"
        }
    ]
}
```

IAM 사용자의 비밀 액세스 키는 나열할 수 없습니다. 비밀 액세스 키를 분실한 경우 `create-access-keys` 명령을 사용하여 새 액세스 키를 생성해야 합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListAccessKeys를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(context.TODO(),
 &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ListAccessKeys를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListAccessKeys {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
                    userName - The name of the user for which access keys are
 retrieved.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listKeys(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void listKeys(IamClient iam, String userName) {
        try {
            boolean done = false;
            String newMarker = null;

            while (!done) {
                ListAccessKeysResponse response;

                if (newMarker == null) {
                    ListAccessKeysRequest request =
ListAccessKeysRequest.builder()
                            .userName(userName)
                            .build();

                    response = iam.listAccessKeys(request);

                } else {
                    ListAccessKeysRequest request =
ListAccessKeysRequest.builder()
                            .userName(userName)
                            .marker(newMarker)
                            .build();

                    response = iam.listAccessKeys(request);
                }

                for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                    System.out.format("Retrieved access key %s",
metadata.accessKeyId());
                }

                if (!response.isTruncated()) {
```

```
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListAccessKeys를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

액세스 키를 나열합니다.

```
import { ListAccessKeysCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 * @param {string} userName
 */
export async function* listAccessKeys(userName) {
  const command = new ListAccessKeysCommand({
```

```
    MaxItems: 5,
    UserName: userName,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListAccessKeysCommandOutput |
 undefined}
   */
  let response = await client.send(command);

  while (response?.AccessKeyMetadata?.length) {
    for (const key of response.AccessKeyMetadata) {
      yield key;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAccessKeysCommand({
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListAccessKeys를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
```

```
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 5,
  UserName: "IAM_USER_NAME",
};

iam.listAccessKeys(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListAccessKeys를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
```

```
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListAccessKeys를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 명령은 **Bob**이라는 IAM 사용자의 액세스 키를 나열합니다. IAM 사용자의 비밀 액세스 키는 나열할 수 없습니다. 비밀 액세스 키를 분실한 경우 **New-IAMAccessKey** cmdlet을 사용하여 새 액세스 키를 생성해야 합니다.

```
Get-IAMAccessKey -UserName "Bob"
```

출력:

```
AccessKeyId               CreateDate                 Status
 UserName
-----------               ----------                 ------
 --------
AKIAIOSFODNN7EXAMPLE      12/3/2014 10:53:41 AM      Active             Bob
AKIAI44QH8DHBEXAMPLE      6/6/2013 8:42:26 PM        Inactive           Bob
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListAccessKeys를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def list_keys(user_name):
    """
```

```
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 ListAccessKeys를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 액세스 키를 나열, 생성, 비활성화 및 삭제합니다.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
```

```
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded => e
    @logger.error("Error creating access key: limit exceeded. Cannot create
more.")
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: "Inactive"
```

```
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListAccessKeys를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListAccountAliases** 사용

다음 코드 예제는 ListAccountAliases의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 계정 관리

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool
AwsDoc::IAM::listAccountAliases(const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccountAliasesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccountAliases(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list account aliases: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        const auto &aliases = outcome.GetResult().GetAccountAliases();
        if (!header) {
            if (aliases.size() == 0) {
                std::cout << "Account has no aliases" << std::endl;
                break;
            }
            std::cout << std::left << std::setw(32) << "Alias" << std::endl;
            header = true;
        }

        for (const auto &alias: aliases) {
            std::cout << std::left << std::setw(32) << alias << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
```

```
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ListAccountAliases를 참조하세요.

CLI

AWS CLI

계정 별칭 나열

다음 `list-account-aliases` 명령은 현재 계정의 별칭을 나열합니다.

```
aws iam list-account-aliases
```

출력:

```
{
    "AccountAliases": [
    "mycompany"
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 AWS 계정 ID 및 별칭을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListAccountAliases를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAliases(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAliases(IamClient iam) {
        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
            for (String alias : response.accountAliases()) {
                System.out.printf("Retrieved account alias %s", alias);
            }
```

```
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListAccountAliases를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

계정 별칭을 나열합니다.

```
import { ListAccountAliasesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 */
export async function* listAccountAliases() {
  const command = new ListAccountAliasesCommand({ MaxItems: 5 });

  let response = await client.send(command);

  while (response.AccountAliases?.length) {
    for (const alias of response.AccountAliases) {
      yield alias;
    }
```

```
    if (response.IsTruncated) {
      response = await client.send(
        new ListAccountAliasesCommand({
          Marker: response.Marker,
          MaxItems: 5,
        }),
      );
    } else {
      break;
    }
  }
}
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListAccountAliases를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listAccountAliases({ MaxItems: 10 }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListAccountAliases를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAliases() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListAccountAliases를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 명령은 AWS 계정에 대한 계정 별칭을 반환합니다.

```
Get-IAMAccountAlias
```

출력:

```
ExampleCo
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListAccountAliases를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListAccountAliases를 참조하세요.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

계정 별칭을 나열하고, 생성하고, 삭제합니다.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListAccountAliases를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListAttachedGroupPolicies** 사용

다음 코드 예제는 `ListAttachedGroupPolicies`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

지정된 그룹에 연결된 모든 관리형 정책 나열

이 예제는 AWS 계정의 이름이 `Admins`인 IAM 그룹에 연결된 관리형 정책의 이름과 ARN을 반환합니다.

```
aws iam list-attached-group-policies \
    --group-name Admins
```

출력:

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        },
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListAttachedGroupPolicies를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 명령은 AWS 계정의 이름이 **Admins**인 IAM 그룹에 연결된 관리형 정책의 이름과 ARN을 반환합니다. 그룹에 포함된 인라인 정책의 목록을 보려면 **Get-IAMGroupPolicyList** 명령을 사용합니다.

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

출력:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess               AdministratorAccess
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListAttachedGroupPolicies를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListAttachedRolePolicies** 사용

다음 코드 예제는 ListAttachedRolePolicies의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListAttachedRolePolicies를 참조하세요.

CLI

AWS CLI

지정된 IAM 역할에 연결된 모든 관리형 정책 나열

이 명령은 AWS 계정의 이름이 `SecurityAuditRole`인 IAM 역할에 연결된 관리형 정책의 이름과 ARN을 반환합니다.

```
aws iam list-attached-role-policies \
    --role-name SecurityAuditRole
```

출력:

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListAttachedRolePolicies를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
```

```
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(roleName string)
 ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(context.TODO(),
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ListAttachedRolePolicies를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

역할에 연결된 정책을 나열합니다.

```
import {
```

```
  ListAttachedRolePoliciesCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 * @param {string} roleName
 */
export async function* listAttachedRolePolicies(roleName) {
  const command = new ListAttachedRolePoliciesCommand({
    RoleName: roleName,
  });

  let response = await client.send(command);

  while (response.AttachedPolicies?.length) {
    for (const policy of response.AttachedPolicies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAttachedRolePoliciesCommand({
          RoleName: roleName,
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListAttachedRolePolicies를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker
 = "", $maxItems = 0)
    {
        $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
        if ($pathPrefix) {
            $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
        }
        if ($marker) {
            $listAttachRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->iamClient-
>listAttachedRolePolicies($listAttachRolePoliciesArguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListAttachedRolePolicies를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 명령은 AWS 계정의 이름이 **SecurityAuditRole**인 IAM 역할에 연결된 관리형 정책의 이름과 ARN을 반환합니다. 역할에 포함된 인라인 정책의 목록을 보려면 **Get-IAMRolePolicyList** 명령을 사용합니다.

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

출력:

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListAttachedRolePolicies를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def list_attached_policies(role_name):
    """
    Lists policies attached to a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.attached_policies.all():
            logger.info("Got policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't list attached policies for %s.", role_name)
        raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 ListAttachedRolePolicies를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 역할 정책을 나열, 생성, 연결 및 분리합니다.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
```

```
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
```

```
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListAttachedRolePolicies를 참조하세요.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn list_attached_role_policies(
    client: &iamClient,
    role_name: String,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListAttachedRolePoliciesOutput,
 SdkError<ListAttachedRolePoliciesError>> {
    let response = client
        .list_attached_role_policies()
        .role_name(role_name)
        .set_path_prefix(path_prefix)
        .set_marker(marker)
```

```
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListAttachedRolePolicies을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /// Returns a list of AWS Identity and Access Management (IAM) policies
   /// that are attached to the role.
   ///
   /// - Parameter role: The IAM role to return the policy list for.
   ///
   /// - Returns: An array of `IAMClientTypes.AttachedPolicy` objects
   ///   describing each managed policy that's attached to the role.
   public func listAttachedRolePolicies(role: String) async throws ->
[IAMClientTypes.AttachedPolicy] {
       var policyList: [IAMClientTypes.AttachedPolicy] = []
       var marker: String? = nil
       var isTruncated: Bool

       repeat {
```

```
            let input = ListAttachedRolePoliciesInput(
                marker: marker,
                roleName: role
            )
            let output = try await client.listAttachedRolePolicies(input: input)

            guard let attachedPolicies = output.attachedPolicies else {
                return policyList
            }

            for attachedPolicy in attachedPolicies {
                policyList.append(attachedPolicy)
            }
            marker = output.marker
            isTruncated = output.isTruncated
        } while isTruncated == true
        return policyList
    }
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 ListAttachedRolePolicies를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListAttachedUserPolicies** 사용

다음 코드 예제는 `ListAttachedUserPolicies`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

지정된 사용자에게 연결된 모든 관리형 정책 나열

이 명령은 AWS 계정의 이름이 Bob인 IAM 사용자에 대한 관리형 정책의 이름과 ARN을 반환합니다.

```
aws iam list-attached-user-policies \
    --user-name Bob
```

출력:

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        },
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListAttachedUserPolicies를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 명령은 AWS 계정의 이름이 **Bob**인 IAM 사용자에 대한 관리형 정책의 이름과 ARN을 반환합니다. IAM 사용자에게 포함된 인라인 정책의 목록을 보려면 **Get-IAMUserPolicyList** 명령을 사용합니다.

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

출력:

```
PolicyArn                                                PolicyName
---------                                                ----------
arn:aws:iam::aws:policy/TesterPolicy                     TesterPolicy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListAttachedUserPolicies를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListEntitiesForPolicy** 사용

다음 코드 예제는 ListEntitiesForPolicy의 사용 방법을 보여 줍니다.

CLI

AWS CLI

지정된 관리형 정책이 연결된 모든 사용자, 그룹 및 역할 나열

이 예제는 arn:aws:iam::123456789012:policy/TestPolicy 정책이 연결된 IAM 그룹, 역할 및 사용자 목록을 반환합니다.

```
aws iam list-entities-for-policy \
    --policy-arn arn:aws:iam::123456789012:policy/TestPolicy
```

출력:

```
{
    "PolicyGroups": [
        {
            "GroupName": "Admins",
            "GroupId": "AGPACKCEVSQ6C2EXAMPLE"
        }
    ],
    "PolicyUsers": [
        {
            "UserName": "Alice",
            "UserId": "AIDACKCEVSQ6C2EXAMPLE"
        }
    ],
    "PolicyRoles": [
        {
            "RoleName": "DevRole",
            "RoleId": "AROADBQP57FF2AEXAMPLE"
        }
    ],
    "IsTruncated": false
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListEntitiesForPolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **arn:aws:iam::123456789012:policy/TestPolicy** 정책이 연결된 IAM 그룹, 역할 및 사용자 목록을 반환합니다.

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

출력:

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListEntitiesForPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListGroupPolicies** 사용

다음 코드 예제는 ListGroupPolicies의 사용 방법을 보여 줍니다.

CLI

AWS CLI

지정된 그룹에 연결된 모든 인라인 정책 나열

다음 list-group-policies 명령은 현재 계정에서 Admins라는 IAM 그룹에 연결된 인라인 정책의 이름을 나열합니다.

```
aws iam list-group-policies \
    --group-name Admins
```

출력:

```
{
    "PolicyNames": [
        "AdminRoot",
        "ExamplePolicy"
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 정책 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListGroupPolicies를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 그룹 **Testers**에 포함된 인라인 정책의 목록을 반환합니다. 그룹에 연결된 관리형 정책을 가져오려면 **Get-IAMAttachedGroupPolicyList** 명령을 사용합니다.

```
Get-IAMGroupPolicyList -GroupName Testers
```

출력:

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListGroupPolicies를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListGroups** 사용

다음 코드 예제는 ListGroups의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListGroups를 참조하십시오.

CLI

AWS CLI

현재 계정의 IAM 그룹 나열

다음 `list-groups` 명령은 현재 계정의 IAM 그룹을 나열합니다.

```
aws iam list-groups
```

출력:

```
{
    "Groups": [
        {
            "Path": "/",
            "CreateDate": "2013-06-04T20:27:27.972Z",
            "GroupId": "AIDACKCEVSQ6C2EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/Admins",
            "GroupName": "Admins"
        },
        {
            "Path": "/",
            "CreateDate": "2013-04-16T20:30:42Z",
            "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/S3-Admins",
            "GroupName": "S3-Admins"
        }
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 그룹 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListGroups를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// GroupWrapper encapsulates AWS Identity and Access Management (IAM) group
 actions
// used in the examples.
// It contains an IAM service client that is used to perform group actions.
type GroupWrapper struct {
 IamClient *iam.Client
```

```
}



// ListGroups lists up to maxGroups number of groups.
func (wrapper GroupWrapper) ListGroups(maxGroups int32) ([]types.Group, error) {
 var groups []types.Group
 result, err := wrapper.IamClient.ListGroups(context.TODO(),
 &iam.ListGroupsInput{
  MaxItems: aws.Int32(maxGroups),
 })
 if err != nil {
  log.Printf("Couldn't list groups. Here's why: %v\n", err)
 } else {
  groups = result.Groups
 }
 return groups, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ListGroups를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

그룹을 나열합니다.

```
import { ListGroupsCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
```

```
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 */
export async function* listGroups() {
  const command = new ListGroupsCommand({
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.Groups?.length) {
    for (const group of response.Groups) {
      yield group;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListGroupsCommand({
          Marker: response.Marker,
          MaxItems: 10,
        }),
      );
    } else {
      break;
    }
  }
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListGroups를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listGroupsArguments = [];
        if ($pathPrefix) {
            $listGroupsArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listGroupsArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listGroupsArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listGroups($listGroupsArguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListGroups를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 현재 AWS 계정에 정의된 모든 IAM 그룹의 모음을 반환합니다.

```
Get-IAMGroupList
```

출력:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE2
```

```
GroupName  : Developers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path       : /
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListGroups를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def list_groups(count):
    """
    Lists the specified number of groups for the account.

    :param count: The number of groups to list.
    """
    try:
        for group in iam.groups.limit(count):
            logger.info("Group: %s", group.name)
    except ClientError:
        logger.exception("Couldn't list groups for the account.")
        raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 ListGroups를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# A class to manage IAM operations via the AWS SDK client
class IamGroupManager
  # Initializes the IamGroupManager class
  # @param iam_client [Aws::IAM::Client] An instance of the IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of groups for the account.
  # @param count [Integer] The maximum number of groups to list.
  # @return [Aws::IAM::Client::Response]
  def list_groups(count)
    response = @iam_client.list_groups(max_items: count)
    response.groups.each do |group|
      @logger.info("\t#{group.group_name}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list groups for the account. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListGroups를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn list_groups(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListGroupsOutput, SdkError<ListGroupsError>> {
    let response = client
        .list_groups()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListGroups을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public func listGroups() async throws -> [String] {
    var groupList: [String] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListGroupsInput(marker: marker)
        let output = try await client.listGroups(input: input)

        guard let groups = output.groups else {
            return groupList
        }

        for group in groups {
            if let name = group.groupName {
                groupList.append(name)
            }
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return groupList
}
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 ListGroups를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **`ListGroupsForUser`** 사용

다음 코드 예제는 `ListGroupsForUser`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자가 속한 그룹 나열

다음 `list-groups-for-user` 명령은 Bob이라는 IAM 사용자가 속한 그룹을 표시합니다.

```
aws iam list-groups-for-user \
    --user-name Bob
```

출력:

```
{
    "Groups": [
        {
            "Path": "/",
            "CreateDate": "2013-05-06T01:18:08Z",
            "GroupId": "AKIAIOSFODNN7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/Admin",
            "GroupName": "Admin"
        },
        {
            "Path": "/",
            "CreateDate": "2013-05-06T01:37:28Z",
            "GroupId": "AKIAI44QH8DHBEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/s3-Users",
            "GroupName": "s3-Users"
        }
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 그룹 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListGroupsForUser를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM 사용자 **David**가 속한 IAM 그룹 목록을 반환합니다.

```
Get-IAMGroupForUser -UserName David
```

출력:

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE2
GroupName  : Testers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE3
GroupName  : Developers
Path       : /
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListGroupsForUser를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListInstanceProfiles** 사용

다음 코드 예제는 ListInstanceProfiles의 사용 방법을 보여 줍니다.

CLI

AWS CLI

계정의 인스턴스 프로파일 나열

다음 list-instance-profiles 명령은 현재 계정과 연결된 인스턴스 프로파일을 나열합니다.

```
aws iam list-instance-profiles
```

출력:

```
{
    "InstanceProfiles": [
        {
            "Path": "/",
            "InstanceProfileName": "example-dev-role",
            "InstanceProfileId": "AIPAIXEU4NUHUPEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:instance-profile/example-dev-role",
            "CreateDate": "2023-09-21T18:17:41+00:00",
            "Roles": [
                {
                    "Path": "/",
                    "RoleName": "example-dev-role",
                    "RoleId": "AROAJ52OTH4H7LEXAMPLE",
                    "Arn": "arn:aws:iam::123456789012:role/example-dev-role",
                    "CreateDate": "2023-09-21T18:17:40+00:00",
                    "AssumeRolePolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                    "Service": "ec2.amazonaws.com"
                                },
                                "Action": "sts:AssumeRole"
                            }
                        ]
                    }
                }
            ]
        },
        {
            "Path": "/",
            "InstanceProfileName": "example-s3-role",
            "InstanceProfileId": "AIPAJVJVNRIQFREXAMPLE",
            "Arn": "arn:aws:iam::123456789012:instance-profile/example-s3-role",
            "CreateDate": "2023-09-21T18:18:50+00:00",
            "Roles": [
                {
                    "Path": "/",
```

```
                    "RoleName": "example-s3-role",
                    "RoleId": "AROAINUBC5O7XLEXAMPLE",
                    "Arn": "arn:aws:iam::123456789012:role/example-s3-role",
                    "CreateDate": "2023-09-21T18:18:49+00:00",
                    "AssumeRolePolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                    "Service": "ec2.amazonaws.com"
                                },
                                "Action": "sts:AssumeRole"
                            }
                        ]
                    }
                }
            ]
        }
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 인스턴스 프로파일 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListInstanceProfiles를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 현재 AWS 계정에 정의된 인스턴스 프로파일의 모음을 반환합니다.

```
Get-IAMInstanceProfileList
```

출력:

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListInstanceProfiles를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListInstanceProfilesForRole** 사용

다음 코드 예제는 ListInstanceProfilesForRole의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 역할에 대한 인스턴스 프로파일 나열

다음 list-instance-profiles-for-role 명령은 Test-Role 역할과 연결된 인스턴스 프로파일을 나열합니다.

```
aws iam list-instance-profiles-for-role \
    --role-name Test-Role
```

출력:

```
{
    "InstanceProfiles": [
        {
            "InstanceProfileId": "AIDGPMS9RO4H3FEXAMPLE",
            "Roles": [
                {
                    "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
                    "RoleId": "AIDACKCEVSQ6C2EXAMPLE",
                    "CreateDate": "2013-06-07T20:42:15Z",
                    "RoleName": "Test-Role",
                    "Path": "/",
                    "Arn": "arn:aws:iam::123456789012:role/Test-Role"
                }
            ],
            "CreateDate": "2013-06-07T21:05:24Z",
            "InstanceProfileName": "ExampleInstanceProfile",
            "Path": "/",
```

```
            "Arn": "arn:aws:iam::123456789012:instance-profile/
ExampleInstanceProfile"
        }
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 인스턴스 프로파일 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListInstanceProfilesForRole을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **ec2instancerole** 역할과 연결된 인스턴스 프로파일의 세부 정보를 반환합니다.

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

출력:

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListInstanceProfilesForRole을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListMfaDevices** 사용

다음 코드 예제는 ListMfaDevices의 사용 방법을 보여 줍니다.

CLI

AWS CLI

지정된 사용자의 모든 MFA 디바이스 나열

이 예제는 IAM 사용자 Bob에게 할당된 MFA 디바이스에 대한 세부 정보를 반환합니다.

```
aws iam list-mfa-devices \
    --user-name Bob
```

출력:

```
{
    "MFADevices": [
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:mfa/Bob",
            "EnableDate": "2019-10-28T20:37:09+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "GAKT12345678",
            "EnableDate": "2023-02-18T21:44:42+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey1-7XNL7NFNLZ123456789EXAMPLE",
            "EnableDate": "2023-09-19T02:25:35+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey2-VDRQTDBBN5123456789EXAMPLE",
            "EnableDate": "2023-09-19T01:49:18+00:00"
        }
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 AWS에서 멀티 팩터 인증(MFA) 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListMfaDevices를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM 사용자 **David**에게 할당된 MFA 디바이스에 대한 세부 정보를 반환합니다. 이 예제에서는 **SerialNumber**가 물리적 디바이스의 실제 일련 번호가 아닌 ARN이므로 가상 장치임을 알 수 있습니다.

```
Get-IAMMFADevice -UserName David
```

출력:

```
EnableDate                SerialNumber                            UserName
----------                ------------                            --------
4/8/2015 9:41:10 AM       arn:aws:iam::123456789012:mfa/David     David
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListMfaDevices를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListOpenIdConnectProviders** 사용

다음 코드 예제는 `ListOpenIdConnectProviders`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

AWS 계정의 OpenID Connect 제공업체에 대한 정보 나열

이 예제는 현재 AWS 계정에 정의된 모든 OpenID Connect 제공업체의 ARNS 목록을 반환합니다.

```
aws iam list-open-id-connect-providers
```

출력:

```
{
    "OpenIDConnectProviderList": [
        {
```

```
            "Arn": "arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com"
        }
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM에서 OIDC(OpenID Connect) ID 제공업체 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListOpenIdConnectProviders를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 현재 AWS 계정에 정의된 모든 OpenID Connect 제공업체의 ARNS 목록을 반환합니다.

```
Get-IAMOpenIDConnectProviderList
```

출력:

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListOpenIdConnectProviders를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **`ListPolicies`** 사용

다음 코드 예제는 `ListPolicies`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 정책 관리

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListPolicies를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::listPolicies(const Aws::Client::ClientConfiguration
 &clientConfig) {
    const Aws::String DATE_FORMAT("%Y-%m-%d");
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListPoliciesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListPolicies(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam policies: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                      std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                      std::setw(64) << "Description" << std::setw(12) <<
                      "CreateDate" << std::endl;
            header = true;
        }

        const auto &policies = outcome.GetResult().GetPolicies();
        for (const auto &policy: policies) {
            std::cout << std::left << std::setw(55) <<
                      policy.GetPolicyName() << std::setw(30) <<
                      policy.GetPolicyId() << std::setw(80) << policy.GetArn() <<
                      std::setw(64) << policy.GetDescription() << std::setw(12)
 <<
                      policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
                      std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }
```

```
    return true;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ListPolicies를 참조하세요.

CLI

AWS CLI

AWS 계정에서 사용할 수 있는 관리형 정책 나열

이 예제는 현재 AWS 계정에서 사용할 수 있는 처음 2개의 관리형 정책 컬렉션을 반환합니다.

```
aws iam list-policies \
    --max-items 3
```

출력:

```
{
    "Policies": [
        {
            "PolicyName": "AWSCloudTrailAccessPolicy",
            "PolicyId": "ANPAXQE2B5PJ7YEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:policy/AWSCloudTrailAccessPolicy",
            "Path": "/",
            "DefaultVersionId": "v1",
            "AttachmentCount": 0,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "CreateDate": "2019-09-04T17:43:42+00:00",
            "UpdateDate": "2019-09-04T17:43:42+00:00"
        },
        {
            "PolicyName": "AdministratorAccess",
            "PolicyId": "ANPAIWMBCKSKIEE64ZLYK",
            "Arn": "arn:aws:iam::aws:policy/AdministratorAccess",
            "Path": "/",
            "DefaultVersionId": "v1",
            "AttachmentCount": 6,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "CreateDate": "2015-02-06T18:39:46+00:00",
```

```
            "UpdateDate": "2015-02-06T18:39:46+00:00"
        },
        {
            "PolicyName": "PowerUserAccess",
            "PolicyId": "ANPAJYRXTHIB4FOVS3ZXS",
            "Arn": "arn:aws:iam::aws:policy/PowerUserAccess",
            "Path": "/",
            "DefaultVersionId": "v5",
            "AttachmentCount": 1,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "CreateDate": "2015-02-06T18:39:47+00:00",
            "UpdateDate": "2023-07-06T22:04:00+00:00"
        }
    ],
    "NextToken": "EXAMPLErZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiA4fQ=="
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListPolicies를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}
```

```
// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(maxPolicies int32) ([]types.Policy,
 error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(context.TODO(),
 &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ListPolicies를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

정책을 나열합니다.

```
import { ListPoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
```

```
 *
 */
export async function* listPolicies() {
  const command = new ListPoliciesCommand({
    MaxItems: 10,
    OnlyAttached: false,
    // List only the customer managed policies in your Amazon Web Services
 account.
    Scope: "Local",
  });

  let response = await client.send(command);

  while (response.Policies?.length) {
    for (const policy of response.Policies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListPoliciesCommand({
          Marker: response.Marker,
          MaxItems: 10,
          OnlyAttached: false,
          Scope: "Local",
        }),
      );
    } else {
      break;
    }
  }
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListPolicies를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listPoliciesArguments = [];
        if ($pathPrefix) {
            $listPoliciesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listPoliciesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listPoliciesArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listPolicies($listPoliciesArguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListPolicies를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 현재 AWS 계정에서 사용할 수 있는 처음 3개의 관리형 정책 컬렉션을 반환합니다. **-scope**가 지정되지 않았기 때문에 기본값은 **all**이며 AWS 관리형 정책과 고객 관리형 정책을 모두 포함합니다.

```
Get-IAMPolicyList -MaxItem 3
```

출력:

```
Arn              : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : AWSDirectConnectReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:08 AM

Arn              : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : NJKMU274MET4EEXAMPLE2
PolicyName       : AmazonGlacierReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:27 AM

Arn              : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount  : 0
CreateDate       : 2/11/2015 9:21:45 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : 5ULJSO2FYVPYGEXAMPLE3
PolicyName       : AWSMarketplaceFullAccess
UpdateDate       : 2/11/2015 9:21:45 AM
```

예제 2: 이 예제는 현재 AWS 계정에서 사용할 수 있는 처음 2개의 고객 관리형 정책 컬렉션을 반환합니다. **`-Scope local`**을 사용하여 고객 관리형 정책으로만 출력을 제한합니다.

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

출력:

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
Path             : /
PolicyId         : SQVCBLC4VAOUCEXAMPLE4
PolicyName       : MyLocalPolicy
UpdateDate       : 2/12/2015 9:39:53 AM

Arn              : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount  : 1
CreateDate       : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description      :
IsAttachable     : True
Path             : /
PolicyId         : X5JPBLJH2Z2SOEXAMPLE5
PolicyName       : policyforec2instancerole
UpdateDate       : 2/18/2015 8:52:31 AM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListPolicies를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
```

```
                'Local' specifies that only locally managed policies are
returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 ListPolicies를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 역할 정책을 나열, 생성, 연결 및 분리합니다.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
```

```
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
```

```
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListPolicies를 참조하세요.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();

    Ok(policy_names)
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListPolicies을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public func listPolicies() async throws -> [MyPolicyRecord] {
    var policyList: [MyPolicyRecord] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListPoliciesInput(marker: marker)
        let output = try await client.listPolicies(input: input)

        guard let policies = output.policies else {
            return policyList
        }

        for policy in policies {
            guard   let name = policy.policyName,
                    let id = policy.policyId,
                    let arn = policy.arn else {
                throw ServiceHandlerError.noSuchPolicy
            }
            policyList.append(MyPolicyRecord(name: name, id: id, arn: arn))
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return policyList
```

```
    }
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 ListPolicies를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListPolicyVersions** 사용

다음 코드 예제는 ListPolicyVersions의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 정책 관리
- 정책 버전 롤백

CLI

AWS CLI

지정된 관리형 정책의 버전에 대한 정보 나열

이 예제는 ARN이 arn:aws:iam::123456789012:policy/MySamplePolicy인 정책의 사용 가능한 버전 목록을 반환합니다.

```
aws iam list-policy-versions \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

출력:

```
{
    "IsTruncated": false,
    "Versions": [
        {
        "VersionId": "v2",
        "IsDefaultVersion": true,
        "CreateDate": "2015-06-02T23:19:44Z"
        },
        {
```

```
        "VersionId": "v1",
        "IsDefaultVersion": false,
        "CreateDate": "2015-06-02T22:30:47Z"
        }
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListPolicyVersions를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/MyManagedPolicy**인 정책의 사용 가능한 버전 목록을 반환합니다. 특정 버전에 대한 정책 문서를 가져오려면 **Get-IAMPolicyVersion** 명령을 사용하고 원하는 버전의 **VersionId**를 지정합니다.

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy
```

출력:

```
CreateDate                  Document                IsDefaultVersion
       VersionId
----------                  --------                ----------------
       ---------
2/12/2015 9:39:53 AM                                True
       v2
2/12/2015 9:39:09 AM                                False
       v1
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListPolicyVersions를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListRolePolicies** 사용

다음 코드 예제는 ListRolePolicies의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListRolePolicies를 참조하십시오.

CLI

AWS CLI

IAM 역할에 연결된 정책 나열

다음 `list-role-policies` 명령은 지정된 IAM 역할의 권한 정책 이름을 나열합니다.

```
aws iam list-role-policies \
    --role-name Test-Role
```

출력:

```
{
    "PolicyNames": [
        "ExamplePolicy"
    ]
}
```

역할에 연결된 신뢰 정책을 보려면 `get-role` 명령을 사용합니다. 권한 정책의 세부 정보를 보려면 `get-role-policy` 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 역할 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListRolePolicies를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(roleName string) ([]string, error) {
```

```
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(context.TODO(),
 &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ListRolePolicies를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

정책을 나열합니다.

```
import { ListRolePoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 * @param {string} roleName
 */
```

```
export async function* listRolePolicies(roleName) {
  const command = new ListRolePoliciesCommand({
    RoleName: roleName,
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.PolicyNames?.length) {
    for (const policyName of response.PolicyNames) {
      yield policyName;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolePoliciesCommand({
          RoleName: roleName,
          MaxItems: 10,
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListRolePolicies를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();
```

```
    public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
    {
        $listRolePoliciesArguments = ['RoleName' => $roleName];
        if ($marker) {
            $listRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->customWaiter(function () use ($listRolePoliciesArguments) {
            return $this->iamClient-
>listRolePolicies($listRolePoliciesArguments);
        });
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListRolePolicies를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM 역할 **lamda_exec_role**에 포함된 인라인 정책 이름의 목록을 반환합니다. 인라인 정책의 세부 정보를 보려면 **Get-IAMRolePolicy** 명령을 사용합니다.

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

출력:

```
oneClick_lambda_exec_role_policy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListRolePolicies를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def list_policies(role_name):
    """
    Lists inline policies for a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.policies.all():
            logger.info("Got inline policy %s.", policy.name)
    except ClientError:
        logger.exception("Couldn't list inline policies for %s.", role_name)
        raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 ListRolePolicies를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  # Lists policy ARNs attached to a role
```

```
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListRolePolicies를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn list_role_policies(
    client: &iamClient,
    role_name: &str,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolePoliciesOutput, SdkError<ListRolePoliciesError>> {
    let response = client
        .list_role_policies()
        .role_name(role_name)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListRolePolicies을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public func listRolePolicies(role: String) async throws -> [String] {
    var policyList: [String] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListRolePoliciesInput(
            marker: marker,
            roleName: role
        )
        let output = try await client.listRolePolicies(input: input)

        guard let policies = output.policyNames else {
            return policyList
        }

        for policy in policies {
            policyList.append(policy)
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return policyList
```

```
    }
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 ListRolePolicies를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListRoleTags** 사용

다음 코드 예제는 ListRoleTags의 사용 방법을 보여 줍니다.

CLI

AWS CLI

역할에 연결된 태그 나열

다음 list-role-tags 명령은 지정된 역할과 연결된 태그 목록을 검색합니다.

```
aws iam list-role-tags \
    --role-name production-role
```

출력:

```
{
    "Tags": [
        {
            "Key": "Department",
            "Value": "Accounting"
        },
        {
            "Key": "DeptID",
            "Value": "12345"
        }
    ],
    "IsTruncated": false
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 리소스 태그 지정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListRoleTags를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 역할과 연결된 태그를 가져옵니다.

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListRoleTags를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListRoles** 사용

다음 코드 예제는 ListRoles의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
```

```
        }

        return roles;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListRoles를 참조하십시오.

CLI

AWS CLI

현재 계정의 IAM 역할 나열

다음 `list-roles` 명령은 현재 계정의 IAM 역할을 나열합니다.

```
aws iam list-roles
```

출력:

```
{
    "Roles": [
        {
            "Path": "/",
            "RoleName": "ExampleRole",
            "RoleId": "AROAJ52OTH4H7LEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:role/ExampleRole",
            "CreateDate": "2017-09-12T19:23:36+00:00",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
```

```
            "MaxSessionDuration": 3600
        },
        {
            "Path": "/example_path/",
            "RoleName": "ExampleRoleWithPath",
            "RoleId": "AROAI4QRP7UFT7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:role/example_path/
ExampleRoleWithPath",
            "CreateDate": "2023-09-21T20:29:38+00:00",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "MaxSessionDuration": 3600
        }
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 역할 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListRoles를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(maxRoles int32) ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(context.TODO(),
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ListRoles를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

역할을 나열합니다.

```
import { ListRolesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});
```

```
/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listRoles() {
  const command = new ListRolesCommand({
    MaxItems: 10,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListRolesCommandOutput | undefined}
   */
  let response = await client.send(command);

  while (response?.Roles?.length) {
    for (const role of response.Roles) {
      yield role;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolesCommand({
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListRoles를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    /**
     * @param string $pathPrefix
     * @param string $marker
     * @param int $maxItems
     * @return Result
     * $roles = $service->listRoles();
     */
    public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listRolesArguments = [];
        if ($pathPrefix) {
            $listRolesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listRolesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listRolesArguments["MaxItems"] = $maxItems;
        }
        return $this->iamClient->listRoles($listRolesArguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListRoles를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 AWS 계정의 모든 IAM 역할 목록을 검색합니다.

```
Get-IAMRoleList
```

예제 2: 이 예제 코드 조각은 AWS 계정의 IAM 역할 목록을 검색하여 한 번에 3개씩 표시하고 각 그룹 사이에서 Enter 키를 누를 때까지 기다립니다. 다음 그룹이 시작되어야 하는 위치를 지정하기 위해 이전 직접 호출의 **`Marker`** 값을 전달합니다.

```
$nextMarker = $null
Do
{
    $results = Get-IAMRoleList -MaxItem 3 -Marker $nextMarker
    $nextMarker = $AWSHistory.LastServiceResponse.Marker
    $results
    Read-Host
} while ($nextMarker -ne $null)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListRoles를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def list_roles(count):
    """
    Lists the specified number of roles for the account.

    :param count: The number of roles to list.
    """
```

```
    try:
        roles = list(iam.roles.limit(count=count))
        for role in roles:
            logger.info("Role: %s", role.name)
    except ClientError:
        logger.exception("Couldn't list roles for the account.")
        raise
    else:
        return roles
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 ListRoles를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Lists IAM roles up to a specified count.
# @param count [Integer] the maximum number of roles to list.
# @return [Array<String>] the names of the roles.
def list_roles(count)
  role_names = []
  roles_counted = 0

  @iam_client.list_roles.each_page do |page|
    page.roles.each do |role|
      break if roles_counted >= count
      @logger.info("\t#{roles_counted + 1}: #{role.role_name}")
      role_names << role.role_name
      roles_counted += 1
    end
    break if roles_counted >= count
  end
```

```
    role_names
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't list roles for the account. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListRoles를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn list_roles(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolesOutput, SdkError<ListRolesError>> {
    let response = client
        .list_roles()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListRoles을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public func listRoles() async throws -> [String] {
    var roleList: [String] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListRolesInput(marker: marker)
        let output = try await client.listRoles(input: input)

        guard let roles = output.roles else {
            return roleList
        }

        for role in roles {
            if let name = role.roleName {
                roleList.append(name)
            }
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return roleList
}
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 ListRoles를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **`ListSAMLProviders`** 사용

다음 코드 예제는 `ListSAMLProviders`의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListSAMLProviders를 참조하십시오.

CLI

AWS CLI

AWS 계정의 SAML 공급자 나열

이 예제는 현재 AWS 계정에서 생성된 SAML 2.0 공급자 목록을 검색합니다.

```
aws iam list-saml-providers
```

출력:

```
{
    "SAMLProviderList": [
        {
            "Arn": "arn:aws:iam::123456789012:saml-provider/SAML-ADFS",
            "ValidUntil": "2015-06-05T22:45:14Z",
            "CreateDate": "2015-06-05T22:45:14Z"
        }
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM SAML 자격 증명 공급자 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListSAMLProviders를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// ListSAMLProviders gets the SAML providers for the account.
```

```
func (wrapper AccountWrapper) ListSAMLProviders() ([]types.SAMLProviderListEntry,
 error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(context.TODO(),
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ListSAMLProviders를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SAML 공급자를 나열합니다.

```
import { ListSAMLProvidersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listSamlProviders = async () => {
  const command = new ListSAMLProvidersCommand({});

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListSAMLProviders를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListSAMLProviders를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 현재 AWS 계정에서 생성된 SAML 2.0 제공업체 목록을 검색합니다. 각 SAML 제공업체의 ARN, 생성 날짜 및 만료 날짜를 반환합니다.

```
Get-IAMSAMLProviderList
```

출력:

```
Arn                                                 CreateDate
   ValidUntil
---                                                 ----------
   ----------
```

```
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
   12/23/2114 12:16:54 PM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListSAMLProviders를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def list_saml_providers(count):
    """
    Lists the SAML providers for the account.

    :param count: The maximum number of providers to list.
    """
    try:
        found = 0
        for provider in iam.saml_providers.limit(count):
            logger.info("Got SAML provider %s.", provider.arn)
            found += 1
        if found == 0:
            logger.info("Your account has no SAML providers.")
    except ClientError:
        logger.exception("Couldn't list SAML providers.")
        raise
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 ListSAMLProviders를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SamlProviderLister
  # Initializes the SamlProviderLister with IAM client and a logger.
  # @param iam_client [Aws::IAM::Client] The IAM client object.
  # @param logger [Logger] The logger object for logging output.
  def initialize(iam_client, logger = Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of SAML providers for the account.
  # @param count [Integer] The maximum number of providers to list.
  # @return [Aws::IAM::Client::Response]
  def list_saml_providers(count)
    response = @iam_client.list_saml_providers
    response.saml_provider_list.take(count).each do |provider|
      @logger.info("\t#{provider.arn}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list SAML providers. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListSAMLProviders를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn list_saml_providers(
    client: &Client,
) -> Result<ListSamlProvidersOutput, SdkError<ListSAMLProvidersError>> {
    let response = client.list_saml_providers().send().await?;

    Ok(response)
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListSAMLProviders을 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 AWS SDK와 함께 이 서비스 사용 섹션을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **`ListServerCertificates`** 사용

다음 코드 예제는 `ListServerCertificates`의 사용 방법을 보여 줍니다.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::listServerCertificates(
        const Aws::Client::ClientConfiguration &clientConfig) {
```

```
    const Aws::String DATE_FORMAT = "%Y-%m-%d";

    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListServerCertificatesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListServerCertificates(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list server certificates: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                      std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                      std::setw(14) << "UploadDate" << std::setw(14) <<
                      "ExpirationDate" << std::endl;
            header = true;
        }

        const auto &certificates =
                outcome.GetResult().GetServerCertificateMetadataList();

        for (const auto &certificate: certificates) {
            std::cout << std::left << std::setw(55) <<
                      certificate.GetServerCertificateName() << std::setw(30) <<
                      certificate.GetServerCertificateId() << std::setw(80) <<
                      certificate.GetArn() << std::setw(14) <<

certificate.GetUploadDate().ToGmtString(DATE_FORMAT.c_str()) <<
                      std::setw(14) <<

certificate.GetExpiration().ToGmtString(DATE_FORMAT.c_str()) <<
                      std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
```

```
        }
    }

    return true;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ListServerCertificates를 참조하세요.

CLI

AWS CLI

AWS 계정의 서버 인증서 나열

다음 `list-server-certificates` 명령은 AWS 계정에 저장되어 사용 가능한 모든 서버 인증서를 나열합니다.

```
aws iam list-server-certificates
```

출력:

```
{
    "ServerCertificateMetadataList": [
        {
            "Path": "/",
            "ServerCertificateName": "myUpdatedServerCertificate",
            "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/
myUpdatedServerCertificate",
            "UploadDate": "2019-04-22T21:13:44+00:00",
            "Expiration": "2019-10-15T22:23:16+00:00"
        },
        {
            "Path": "/cloudfront/",
            "ServerCertificateName": "MyTestCert",
            "ServerCertificateId": "ASCAEXAMPLE456EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyTestCert",
            "UploadDate": "2015-04-21T18:14:16+00:00",
            "Expiration": "2018-01-14T17:52:36+00:00"
        }
```

```
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM에서 서버 인증서 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListServerCertificates를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

인증서를 나열합니다.

```
import { ListServerCertificatesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listServerCertificates() {
  const command = new ListServerCertificatesCommand({});
  let response = await client.send(command);

  while (response.ServerCertificateMetadataList?.length) {
    for await (const cert of response.ServerCertificateMetadataList) {
      yield cert;
    }

    if (response.IsTruncated) {
      response = await client.send(new ListServerCertificatesCommand({}));
    } else {
```

```
      break;
    }
  }
}
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListServerCertificates를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listServerCertificates({}, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListServerCertificates를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 현재 AWS 계정에 업로드된 서버 인증서 목록을 검색합니다.

```
Get-IAMServerCertificateList
```

출력:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListServerCertificates를 참조하세요.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

서버 인증서를 나열하고, 업데이트하고, 삭제합니다.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "ServerCertificateManager"
  end

  # Creates a new server certificate.
```

```
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key,
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info("No server certificates found.")
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
'#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
```

```
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListServerCertificates를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListSigningCertificates** 사용

다음 코드 예제는 ListSigningCertificates의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자의 서명 인증서 나열

다음 list-signing-certificates 명령은 Bob이라는 IAM 사용자의 서명 인증서를 나열합니다.

```
aws iam list-signing-certificates \
    --user-name Bob
```

출력:

```
{
    "Certificates": [
        {
            "UserName": "Bob",
```

```
            "Status": "Inactive",
            "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-
body>-----END CERTIFICATE-----",
            "CertificateId": "TA7SMP42TDN5Z26OBPJE7EXAMPLE",
            "UploadDate": "2013-06-06T21:40:08Z"
        }
    ]
}
```

자세한 내용은 Amazon EC2 사용 설명서의 서명 인증서 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListSigningCertificates를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **Bob**이라는 사용자와 연결된 서명 인증서에 대한 세부 정보를 검색합니다.

```
Get-IAMSigningCertificate -UserName Bob
```

출력:

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                    YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                    21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
```

```
                    rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                    nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                    NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                    -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListSigningCertificates를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListUserPolicies** 사용

다음 코드 예제는 `ListUserPolicies`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자에 대한 정책 나열

다음 `list-user-policies` 명령은 이름이 Bob인 IAM 사용자에게 연결된 정책을 나열합니다.

```
aws iam list-user-policies \
    --user-name Bob
```

출력:

```
{
    "PolicyNames": [
```

```
        "ExamplePolicy",
        "TestPolicy"
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 AWS 계정에서 IAM 사용자 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListUserPolicies를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(userName string) ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(context.TODO(),
 &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
```

```
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ListUserPolicies을 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **David**라는 IAM 사용자에게 포함된 인라인 정책의 이름 목록을 검색합니다.

```
Get-IAMUserPolicyList -UserName David
```

출력:

```
Davids_IAM_Admin_Policy
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListUserPolicies를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListUserTags** 사용

다음 코드 예제는 ListUserTags의 사용 방법을 보여 줍니다.

CLI

AWS CLI

사용자에게 연결된 태그 나열

다음 list-user-tags 명령은 지정된 IAM 사용자와 연결된 태그 목록을 검색합니다.

```
aws iam list-user-tags \
    --user-name alice
```

출력:

```
{
    "Tags": [
        {
            "Key": "Department",
            "Value": "Accounting"
        },
        {
            "Key": "DeptID",
            "Value": "12345"
        }
    ],
    "IsTruncated": false
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 리소스 태그 지정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListUserTags를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 사용자와 연결된 태그를 가져옵니다.

```
Get-IAMUserTagList -UserName joe
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListUserTags를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListUsers** 사용

다음 코드 예제는 ListUsers의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 읽기 전용 및 읽기-쓰기 사용자 생성

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 ListUsers를 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
########################################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
########################################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

########################################################################################
# function iam_list_users
#
# List the IAM users in the account.
#
# Returns:
#       The list of users names
#    And:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
########################################################################################
function iam_list_users() {
  local option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_users"
    echo "Lists the AWS Identity and Access Management (IAM) user in the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
```

```
    esac
  done
  export OPTIND=1

  local response

  response=$(aws iam list-users \
    --output text \
    --query "Users[].UserName")
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-users operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API 세부 정보는 AWS CLI 명령 참조의 ListUsers를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::listUsers(const Aws::Client::ClientConfiguration &clientConfig)
 {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListUsersRequest request;

    bool done = false;
```

```
    bool header = false;
    while (!done) {
        auto outcome = iam.ListUsers(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam users:" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "Name" <<
                      std::setw(30) << "ID" << std::setw(64) << "Arn" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &users = outcome.GetResult().GetUsers();
        for (const auto &user: users) {
            std::cout << std::left << std::setw(32) << user.GetUserName() <<
                      std::setw(30) << user.GetUserId() << std::setw(64) <<
                      user.GetArn() << std::setw(20) <<
                      user.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
                      << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 ListUsers를 참조하십시오.

CLI

AWS CLI

IAM 사용자 나열

다음 `list-users` 명령은 현재 계정의 IAM 사용자를 나열합니다.

```
aws iam list-users
```

출력:

```
{
    "Users": [
        {
            "UserName": "Adele",
            "Path": "/",
            "CreateDate": "2013-03-07T05:14:48Z",
            "UserId": "AKIAI44QH8DHBEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/Adele"
        },
        {
            "UserName": "Bob",
            "Path": "/",
            "CreateDate": "2012-09-21T23:03:13Z",
            "UserId": "AKIAIOSFODNN7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/Bob"
        }
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 나열을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListUsers를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(maxUsers int32) ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(context.TODO(), &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 ListUsers를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
```

```
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
                if (newMarker == null) {
                    ListUsersRequest request =
 ListUsersRequest.builder().build();
                    response = iam.listUsers(request);
                } else {
                    ListUsersRequest request = ListUsersRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = iam.listUsers(request);
                }

                for (User user : response.users()) {
                    System.out.format("\n Retrieved user %s", user.userName());
                    AttachedPermissionsBoundary permissionsBoundary =
 user.permissionsBoundary();
                    if (permissionsBoundary != null)
                        System.out.format("\n Permissions boundary details %s",

 permissionsBoundary.permissionsBoundaryTypeAsString());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListUsers를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

사용자를 나열합니다.

```
import { ListUsersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listUsers = async () => {
  const command = new ListUsersCommand({ MaxItems: 10 });

  const response = await client.send(command);
  response.Users?.forEach(({ UserName, CreateDate }) => {
    console.log(`${UserName} created on: ${CreateDate}`);
  });
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListUsers를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
```

```
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 10,
};

iam.listUsers(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var users = data.Users || [];
    users.forEach(function (user) {
      console.log("User " + user.UserName + " created", user.CreateDate);
    });
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 ListUsers를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listAllUsers() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
            if (permissionsBoundary != null) {
```

```
                println("Permissions boundary details
 ${permissionsBoundary.permissionsBoundaryType}")
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 ListUsers를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$uuid = uniqid();
$service = new IAMService();

    public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listUsersArguments = [];
        if ($pathPrefix) {
            $listUsersArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listUsersArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listUsersArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listUsers($listUsersArguments);
    }
```

- API 세부 정보는 AWS SDK for PHP API 참조의 ListUsers를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 현재 AWS 계정의 사용자 모음을 검색합니다.

```
Get-IAMUserList
```

출력:

```
Arn              : arn:aws:iam::123456789012:user/Administrator
CreateDate       : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path             : /
UserId           : 7K3GJEANSKZF2EXAMPLE1
UserName         : Administrator

Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 4/6/2015 12:54:42 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : L3EWNONDOM3YUEXAMPLE2
UserName         : bab

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListUsers를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 ListUsers를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Lists all users in the AWS account
#
# @return [Array<Aws::IAM::Types::User>] An array of user objects
def list_users
  users = []
  @iam_client.list_users.each_page do |page|
    page.users.each do |user|
      users << user
    end
  end
  users
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing users: #{e.message}")
    []
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListUsers를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn list_users(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListUsersOutput, SdkError<ListUsersError>> {
    let response = client
        .list_users()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 ListUsers을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public func listUsers() async throws -> [MyUserRecord] {
    var userList: [MyUserRecord] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListUsersInput(marker: marker)
        let output = try await client.listUsers(input: input)

        guard let users = output.users else {
            return userList
        }

        for user in users {
            if let id = user.userId, let name = user.userName {
                userList.append(MyUserRecord(id: id, name: name))
            }
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return userList
}
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 ListUsers를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ListVirtualMfaDevices** 사용

다음 코드 예제는 `ListVirtualMfaDevices`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

가상 MFA 디바이스 나열

다음 `list-virtual-mfa-devices` 명령은 현재 계정에 대해 구성된 가상 MFA 디바이스를 나열합니다.

```
aws iam list-virtual-mfa-devices
```

출력:

```
{
    "VirtualMFADevices": [
        {
            "SerialNumber": "arn:aws:iam::123456789012:mfa/ExampleMFADevice"
        },
        {
            "SerialNumber": "arn:aws:iam::123456789012:mfa/Fred"
        }
    ]
}
```

자세한 내용은 AWS IAM 사용 설명서의 가상 멀티 팩터 인증(MFA) 디바이스 활성화를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ListVirtualMfaDevices를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 AWS 계정의 사용자에게 할당된 가상 MFA 디바이스 모음을 검색합니다. 각각의 **User** 속성은 해당 디바이스가 할당된 IAM 사용자의 세부 정보가 포함된 객체입니다.

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

출력:

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/David
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ListVirtualMfaDevices를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutGroupPolicy** 사용

다음 코드 예제는 PutGroupPolicy의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 그룹 생성 및 사용자 추가

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutGroupPolicyAsync(string groupName, string
policyName, string policyDocument)
    {
        var request = new PutGroupPolicyRequest
        {
            GroupName = groupName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 PutGroupPolicy를 참조하십시오.

CLI

AWS CLI

그룹에 정책 추가

다음 put-group-policy 명령은 이름이 Admins인 IAM 그룹에 정책을 추가합니다.

```
aws iam put-group-policy \
    --group-name Admins \
    --policy-document file://AdminPolicy.json \
    --policy-name AdminRoot
```

이 명령은 출력을 생성하지 않습니다.

정책은 AdminPolicy.json 파일에서 JSON 문서로 정의됩니다. (파일 이름과 확장자는 중요하지 않습니다.)

자세한 내용은 AWS IAM 사용 설명서의 IAM 정책 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 PutGroupPolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **AppTesterPolicy**라는 인라인 정책을 생성하고 이를 IAM 그룹 **AppTesters**에 포함합니다. 동일한 이름의 인라인 정책이 이미 존재하는 경우 해당 정책을 덮어씁니다. JSON 정책 내용은 **apptesterpolicy.json** 파일로 제공됩니다. JSON 파일의 내용을 성공적으로 처리하려면 **-Raw** 파라미터를 사용해야 합니다.

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 PutGroupPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutRolePermissionsBoundary** 사용

다음 코드 예제는 PutRolePermissionsBoundary의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: IAM 역할에 사용자 지정 정책을 기반으로 권한 경계 적용

다음 put-role-permissions-boundary 예제는 intern-boundary라는 사용자 지정 정책을 지정된 IAM 역할에 대한 권한 경계로 적용합니다.

```
aws iam put-role-permissions-boundary \
    --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \
    --role-name lambda-application-role
```

이 명령은 출력을 생성하지 않습니다.

예제 2: IAM 역할에 AWS 관리형 정책을 기반으로 권한 경계를 적용하는 방법

다음 put-role-permissions-boundary 예제는 AWS 관리형PowerUserAccess 정책을 지정된 IAM 역할에 대한 권한 경계로 적용합니다.

```
aws iam put-role-permissions-boundary \
    --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \
    --role-name x-account-admin
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 역할 변경을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 PutRolePermissionsBoundary를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM 역할의 권한 경계를 설정하는 방법을 보여줍니다. AWS 관리형 정책 또는 사용자 지정 정책을 권한 경계로 설정할 수 있습니다.

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 PutRolePermissionsBoundary를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutRolePolicy** 사용

다음 코드 예제는 PutRolePolicy의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 PutRolePolicy를 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::putRolePolicy(
        const Aws::String &roleName,
        const Aws::String &policyName,
        const Aws::String &policyDocument,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iamClient(clientConfig);
    Aws::IAM::Model::PutRolePolicyRequest request;

    request.SetRoleName(roleName);
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(policyDocument);

    Aws::IAM::Model::PutRolePolicyOutcome outcome =
 iamClient.PutRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error putting policy on role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully put the role policy." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 PutRolePolicy를 참조하십시오.

CLI

AWS CLI

IAM 역할에 권한 정책 연결

다음 put-role-policy 명령은 이름이 Test-Role인 역할에 권한 정책을 추가합니다.

```
aws iam put-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy \
    --policy-document file://AdminPolicy.json
```

이 명령은 출력을 생성하지 않습니다.

정책은 AdminPolicy.json 파일에서 JSON 문서로 정의됩니다. (파일 이름과 확장자는 중요하지 않습니다.)

신뢰 정책을 역할에 연결하려면 update-assume-role-policy 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 역할 변경을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 PutRolePolicy를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { PutRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const examplePolicyDocument = JSON.stringify({
  Version: "2012-10-17",
  Statement: [
    {
      Sid: "VisualEditor0",
      Effect: "Allow",
```

```
      Action: [
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListBucket",
        "s3:ListMultipartUploadParts",
      ],
      Resource: "arn:aws:s3:::some-test-bucket",
    },
    {
      Sid: "VisualEditor1",
      Effect: "Allow",
      Action: [
        "s3:ListStorageLensConfigurations",
        "s3:ListAccessPointsForObjectLambda",
        "s3:ListAllMyBuckets",
        "s3:ListAccessPoints",
        "s3:ListJobs",
        "s3:ListMultiRegionAccessPoints",
      ],
      Resource: "*",
    },
  ],
});

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 * @param {string} policyDocument
 */
export const putRolePolicy = async (roleName, policyName, policyDocument) => {
  const command = new PutRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
    PolicyDocument: policyDocument,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 PutRolePolicy를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **FedTesterRolePolicy**라는 인라인 정책을 생성하고 이를 IAM 역할 **FedTesterRole**에 포함합니다. 동일한 이름의 인라인 정책이 이미 존재하는 경우 해당 정책을 덮어씁니다. JSON 정책 내용은 **FedTesterPolicy.json** 파일에서 가져옵니다. JSON 파일의 내용을 성공적으로 처리하려면 **-Raw** 파라미터를 사용해야 합니다.

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 PutRolePolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutUserPermissionsBoundary** 사용

다음 코드 예제는 PutUserPermissionsBoundary의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: IAM 사용자에게 사용자 지정 정책을 기반으로 권한 경계 적용

다음 put-user-permissions-boundary 예제는 intern-boundary라는 사용자 지정 정책을 지정된 IAM 사용자에 대한 권한 경계로 적용합니다.

```
aws iam put-user-permissions-boundary \
    --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \
    --user-name intern
```

이 명령은 출력을 생성하지 않습니다.

예제 2: IAM 사용자에게 AWS 관리형 정책을 기반으로 권한 경계를 적용하는 방법

다음 `put-user-permissions-boundary` 예제는 `PowerUserAccess`라는 AWS 관리형 정책을 지정된 IAM 사용자에 대한 권한 경계로 적용합니다.

```
aws iam put-user-permissions-boundary \
    --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \
    --user-name developer
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 PutUserPermissionsBoundary를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 사용자의 권한 경계를 설정하는 방법을 보여줍니다. AWS 관리형 정책 또는 사용자 지정 정책을 권한 경계로 설정할 수 있습니다.

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 PutUserPermissionsBoundary를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **PutUserPolicy** 사용

다음 코드 예제는 `PutUserPolicy`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 사용자 생성 및 역할 수임

CLI

AWS CLI

IAM 사용자에게 정책 연결

다음 put-user-policy 명령은 정책을 이름이 Bob인 IAM 사용자에게 연결합니다.

```
aws iam put-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy \
    --policy-document file://AdminPolicy.json
```

이 명령은 출력을 생성하지 않습니다.

정책은 AdminPolicy.json 파일에서 JSON 문서로 정의됩니다. (파일 이름과 확장자는 중요하지 않습니다.)

자세한 내용은 AWS IAM 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 PutUserPolicy를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}
```

```
// CreateUserPolicy adds an inline policy to a user. This example creates a
 policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(userName string, policyName string,
 actions []string,
  roleArn string) error {
 policyDoc := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Action: actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(context.TODO(),
 &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 PutUserPolicy를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **EC2AccessPolicy**라는 인라인 정책을 생성하고 이를 IAM 사용자 **Bob**에게 포함합니다. 동일한 이름의 인라인 정책이 이미 존재하는 경우 해당 정책을 덮어씁니다. JSON 정책 내용은 **EC2AccessPolicy.json** 파일에서 가져옵니다. JSON 파일의 내용을 성공적으로 처리하려면 **-Raw** 파라미터를 사용해야 합니다.

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument
 (Get-Content -Raw EC2AccessPolicy.json)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 PutUserPolicy를 참조하세요.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
 # Creates an inline policy for a specified user.
 # @param username [String] The name of the IAM user.
 # @param policy_name [String] The name of the policy to create.
 # @param policy_document [String] The JSON policy document.
 # @return [Boolean]
 def create_user_policy(username, policy_name, policy_document)
   @iam_client.put_user_policy({
     user_name: username,
     policy_name: policy_name,
     policy_document: policy_document
   })
   @logger.info("Policy #{policy_name} created for user #{username}.")
   true
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Couldn't create policy #{policy_name} for user #{username}.
Here's why:")
```

```
    @logger.error("\t#{e.code}: #{e.message}")
    false
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 PutUserPolicy를 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    func putUserPolicy(policyDocument: String, policyName: String, user:
IAMClientTypes.User) async throws {
        let input = PutUserPolicyInput(
            policyDocument: policyDocument,
            policyName: policyName,
            userName: user.userName
        )
        do {
            _ = try await iamClient.putUserPolicy(input: input)
        } catch {
            throw error
        }
    }
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 PutUserPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **RemoveClientIdFromOpenIdConnectProvider** 사용

다음 코드 예제는 RemoveClientIdFromOpenIdConnectProvider의 사용 방법을 보여 줍니다.

CLI

AWS CLI

지정된 IAM OpenID Connect 제공업체에 등록된 클라이언트 ID 목록에서 지정된 클라이언트 ID 제거

이 예제는 ARN이 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com인 IAM OIDC 제공업체와 연결된 클라이언트 ID 목록에서 클라이언트 ID My-TestApp-3을 제거합니다.

```
aws iam remove-client-id-from-open-id-connect-provider
    --client-id My-TestApp-3 \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM에서 OIDC(OpenID Connect) ID 제공업체 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 RemoveClientIdFromOpenIdConnectProvider를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com**인 IAM OIDC 제공업체와 연결된 클라이언트 ID 목록에서 클라이언트 ID **My-TestApp-3**을 제거합니다.

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
 -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 RemoveClientIdFromOpenIdConnectProvider를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **RemoveRoleFromInstanceProfile** 사용

다음 코드 예제는 RemoveRoleFromInstanceProfile의 사용 방법을 보여 줍니다.

CLI

AWS CLI

인스턴스 프로파일에서 역할 제거

다음 remove-role-from-instance-profile 명령은 이름이 ExampleInstanceProfile인 인스턴스 프로파일에서 이름이 Test-Role인 역할을 제거합니다.

```
aws iam remove-role-from-instance-profile \
    --instance-profile-name ExampleInstanceProfile \
    --role-name Test-Role
```

자세한 내용은 AWS IAM 사용 설명서의 인스턴스 프로파일 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 RemoveRoleFromInstanceProfile을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **MyNewRole**이라는 EC2 인스턴스 프로파일에서 **MyNewRole**이라는 역할을 삭제합니다. IAM 콘솔에서 생성되는 인스턴스 프로파일은 이 예제와 같이 항상 역할과 동일한 이름을 갖습니다. API 또는 CLI에서 생성하는 경우 서로 다른 이름을 가질 수 있습니다.

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName
 MyNewRole -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 RemoveRoleFromInstanceProfile를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **RemoveUserFromGroup** 사용

다음 코드 예제는 RemoveUserFromGroup의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 그룹 생성 및 사용자 추가

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Remove a user from an IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to remove.</param>
    /// <param name="groupName">The name of the IAM group to remove the user
from.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> RemoveUserFromGroupAsync(string userName, string
groupName)
    {
        // Remove the user from the group.
        var removeUserRequest = new RemoveUserFromGroupRequest()
        {
            UserName = userName,
            GroupName = groupName,
        };

        var response = await
_IAMService.RemoveUserFromGroupAsync(removeUserRequest);
```

```
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 RemoveUserFromGroup을 참조하십시오.

CLI

AWS CLI

IAM 그룹에서 사용자 제거

다음 `remove-user-from-group` 명령은 이름이 `Admins`인 IAM 그룹에서 이름이 `Bob`인 사용자를 제거합니다.

```
aws iam remove-user-from-group \
    --user-name Bob \
    --group-name Admins
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 그룹에서 사용자 추가 및 제거를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 RemoveUserFromGroup을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 그룹 **Testers**에서 IAM 사용자 **Bob**을 제거합니다.

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

예제 2: 이 예제는 IAM 사용자 **Theresa**가 구성원으로 속한 그룹을 찾은 다음 해당 그룹에서 **Theresa**를 제거합니다.

```
$groups = Get-IAMGroupForUser -UserName Theresa
```

```
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName Theresa -Force }
```

예제 3: 이 예제는 **Testers** 그룹에서 IAM 사용자 **Bob**을 제거하는 다른 방법을 보여줍니다.

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -
GroupName Testers -Force
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 RemoveUserFromGroup을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **ResyncMfaDevice** 사용

다음 코드 예제는 ResyncMfaDevice의 사용 방법을 보여 줍니다.

CLI

AWS CLI

MFA 디바이스 동기화

다음 resync-mfa-device 예제는 IAM 사용자 Bob과 연결되어 있고 ARN이 arn:aws:iam::123456789012:mfa/BobsMFADevice인 MFA 디바이스를 두 개의 인증 코드를 제공한 인증 프로그램과 동기화합니다.

```
aws iam resync-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \
    --authentication-code1 123456 \
    --authentication-code2 987654
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 AWS에서 멀티 팩터 인증(MFA) 사용을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 ResyncMfaDevice를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM 사용자 **Bob**과 연결되어 있고 ARN이 **arn:aws:iam::123456789012:mfa/bob**인 MFA 디바이스를 두 개의 인증 코드를 제공한 인증 프로그램과 동기화합니다.

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

예제 2: 이 예제는 IAM 사용자 **Theresa**와 연결된 IAM MFA 디바이스를 일련 번호가 **ABCD12345678**이고 두 개의 인증 코드를 제공한 물리적 디바이스와 동기화합니다.

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 ResyncMfaDevice를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **SetDefaultPolicyVersion** 사용

다음 코드 예제는 SetDefaultPolicyVersion의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 정책 관리
- 정책 버전 롤백

CLI

AWS CLI

지정된 정책의 지정된 버전을 정책의 기본 버전으로 설정

이 예제는 ARN이 arn:aws:iam::123456789012:policy/MyPolicy인 정책의 v2 버전을 기본 활성 버전으로 설정합니다.

```
aws iam set-default-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

자세한 내용은 AWS IAM 사용 설명서의 IAM의 정책 및 권한을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 SetDefaultPolicyVersion을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:policy/MyPolicy**인 정책의 **v2** 버전을 기본 활성 버전으로 설정합니다.

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy
 -VersionId v2
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 SetDefaultPolicyVersion을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **TagRole** 사용

다음 코드 예제는 TagRole의 사용 방법을 보여 줍니다.

CLI

AWS CLI

역할에 태그 추가

다음 tag-role 명령은 지정된 역할에 Department 이름이 포함된 태그를 추가합니다.

```
aws iam tag-role --role-name my-role \
    --tags '{"Key": "Department", "Value": "Accounting"}'
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 리소스 태그 지정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 TagRole을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ID 관리 서비스의 역할에 태그를 추가합니다.

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 TagRole을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **TagUser** 사용

다음 코드 예제는 TagUser의 사용 방법을 보여 줍니다.

CLI

AWS CLI

사용자에게 태그 추가

다음 tag-user 명령은 지정된 사용자에게 연관된 Department가 포함된 태그를 추가합니다.

```
aws iam tag-user \
    --user-name alice \
    --tags '{"Key": "Department", "Value": "Accounting"}'
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 리소스 태그 지정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 TagUser를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ID 관리 서비스의 사용자에게 태그를 추가합니다.

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 TagUser를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UntagRole** 사용

다음 코드 예제는 UntagRole의 사용 방법을 보여 줍니다.

CLI

AWS CLI

역할에서 태그 제거

다음 untag-role 명령은 지정된 역할에서 키 이름이 'Department'인 모든 태그를 제거합니다.

```
aws iam untag-role \
    --role-name my-role \
    --tag-keys Department
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 리소스 태그 지정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UntagRole을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 태그 키가 'abac'인 'MyRoleName' 역할에서 태그를 제거합니다. 여러 태그를 제거하려면 쉼표로 구분된 태그 키 목록을 제공합니다.

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UntagRole을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UntagUser** 사용

다음 코드 예제는 UntagUser의 사용 방법을 보여 줍니다.

CLI

AWS CLI

사용자에게서 태그 제거

다음 untag-user 명령은 지정된 사용자에게서 키 이름이 'Department'인 모든 태그를 제거합니다.

```
aws iam untag-user \
    --user-name alice \
    --tag-keys Department
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 리소스 태그 지정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UntagUser를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 태그 키가 'abac' 및 'xyzw'인 'joe'라는 사용자에게서 태그를 제거합니다. 여러 태그를 제거하려면 쉼표로 구분된 태그 키 목록을 제공합니다.

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UntagUser를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateAccessKey** 사용

다음 코드 예제는 UpdateAccessKey의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 액세스 키 관리

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::updateAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  Aws::IAM::Model::StatusType status,
                                  const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);
    request.SetStatus(status);

    auto outcome = iam.UpdateAccessKey(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated status of access key "
                  << accessKeyID << " for user " << userName << std::endl;
    }
    else {
        std::cerr << "Error updated status of access key " << accessKeyID <<
                  " for user " << userName << ": " <<
```

```
                    outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 UpdateAccessKey를 참조하십시오.

CLI

AWS CLI

IAM 사용자의 액세스 키를 활성화 또는 비활성화

다음 update-access-key 명령은 이름이 Bob인 IAM 사용자의 지정된 액세스 키(액세스 키 ID 및 비밀 액세스 키)를 비활성화합니다.

```
aws iam update-access-key \
    --access-key-id AKIAIOSFODNN7EXAMPLE \
    --status Inactive \
    --user-name Bob
```

이 명령은 출력을 생성하지 않습니다.

키를 비활성화하면 프로그래밍 방식으로 AWS에 액세스하는 데 키를 사용할 수 없습니다. 하지만 키는 계속 사용할 수 있고 다시 활성화할 수 있습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateAccessKey를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UpdateAccessKey {

    private static StatusType statusType;

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username> <accessId> <status>\s

                Where:
                    username - The name of the user whose key you want to update.
\s
                    accessId - The access key ID of the secret access key you
 want to update.\s
                    status - The status you want to assign to the secret access
 key.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessId = args[1];
        String status = args[2];
        Region region = Region.AWS_GLOBAL;
```

```
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateKey(iam, username, accessId, status);
        System.out.println("Done");
        iam.close();
    }

    public static void updateKey(IamClient iam, String username, String accessId,
 String status) {
        try {
            if (status.toLowerCase().equalsIgnoreCase("active")) {
                statusType = StatusType.ACTIVE;
            } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
                statusType = StatusType.INACTIVE;
            } else {
                statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
            }

            UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
                    .accessKeyId(accessId)
                    .userName(username)
                    .status(statusType)
                    .build();

            iam.updateAccessKey(request);
            System.out.printf("Successfully updated the status of access key %s
 to" +
                    "status %s for user %s", accessId, status, username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateAccessKey를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

액세스 키를 업데이트합니다.

```
import {
  UpdateAccessKeyCommand,
  IAMClient,
  StatusType,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const updateAccessKey = (userName, accessKeyId) => {
  const command = new UpdateAccessKeyCommand({
    AccessKeyId: accessKeyId,
    Status: StatusType.Inactive,
    UserName: userName,
  });

  return client.send(command);
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 UpdateAccessKey를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  Status: "Active",
  UserName: "USER_NAME",
};

iam.updateAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 UpdateAccessKey를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **Bob**이라는 IAM 사용자에 대한 액세스 키 **AKIAIOSFODNN7EXAMPLE**의 상태를 **Inactive**로 변경합니다.

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status
 Inactive
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UpdateAccessKey를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
 deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
 key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate",
 key_id
        )
        raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 UpdateAccessKey를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateAccountPasswordPolicy** 사용

다음 코드 예제는 UpdateAccountPasswordPolicy의 사용 방법을 보여 줍니다.

CLI

AWS CLI

현재 계정 암호 정책 설정 또는 변경

다음 update-account-password-policy 명령은 최소 8자 길이를 요구하고 암호에 하나 이상의 숫자를 요구하도록 암호 정책을 설정합니다.

```
aws iam update-account-password-policy \
    --minimum-password-length 8 \
    --require-numbers
```

이 명령은 출력을 생성하지 않습니다.

계정의 암호 정책 변경은 해당 계정의 IAM 사용자에 대해 새로 생성되는 모든 암호에 영향을 줍니다. 암호 정책 변경은 기존 암호에 영향을 주지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자의 계정 암호 정책 설정을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateAccountPasswordPolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 계정의 암호 정책을 지정된 설정으로 업데이트합니다. 단, 명령에 포함되지 않은 파라미터는 수정되지 않은 채로 남아 있지 않습니다. 대신 기본값으로 재설정됩니다.

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UpdateAccountPasswordPolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateAssumeRolePolicy** 사용

다음 코드 예제는 UpdateAssumeRolePolicy의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 역할의 신뢰 정책 업데이트

다음 update-assume-role-policy 명령은 Test-Role이라는 역할에 대한 신뢰 정책을 업데이트합니다.

```
aws iam update-assume-role-policy \
    --role-name Test-Role \
    --policy-document file://Test-Role-Trust-Policy.json
```

이 명령은 출력을 생성하지 않습니다.

신뢰 정책은 Test-Role-Trust-Policy.json 파일에 JSON 문서로 정의됩니다. (파일 이름과 확장자는 중요하지 않습니다.) 신뢰 정책에서 보안 주체를 지정해야 합니다.

역할에 대한 권한 정책을 업데이트하려면 put-role-policy 명령을 사용합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 역할 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateAssumeRolePolicy를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 이름이 **ClientRole**인 IAM 역할을 새 신뢰 정책으로 업데이트하고, 그 내용은 **ClientRolePolicy.json** 파일에서 가져옵니다. JSON 파일의 내용을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다.

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UpdateAssumeRolePolicy를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateGroup** 사용

다음 코드 예제는 UpdateGroup의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 그룹 이름 바꾸기

다음 update-group 명령은 IAM 그룹의 이름을 Test에서 Test-1로 변경합니다.

```
aws iam update-group \
    --group-name Test \
    --new-group-name Test-1
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 그룹 이름 변경을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateGroup을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM 그룹 이름을 **Testers**에서 **AppTesters**로 변경합니다.

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

예제 2: 이 예제는 IAM 그룹 **AppTesters**의 경로를 **/Org1/Org2/**로 변경합니다. 그러면 그룹의 ARN이 **arn:aws:iam::123456789012:group/Org1/Org2/AppTesters**로 변경됩니다.

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UpdateGroup을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateLoginProfile** 사용

다음 코드 예제는 UpdateLoginProfile의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자의 암호 업데이트

다음 update-login-profile 명령은 Bob이라는 IAM 사용자에 대한 새 암호를 생성합니다.

```
aws iam update-login-profile \
    --user-name Bob \
    --password <password>
```

이 명령은 출력을 생성하지 않습니다.

계정의 암호 정책을 설정하려면 update-account-password-policy 명령을 사용합니다. 새 암호가 계정 암호 정책을 위반하는 경우 명령은 PasswordPolicyViolation 오류를 반환합니다.

계정 암호 정책에서 허용하는 경우 IAM 사용자는 `change-password` 명령을 사용하여 자신의 암호를 변경할 수 있습니다.

암호를 안전한 위치에 저장합니다. 암호를 분실한 경우 복구가 불가능하며, `create-login-profile` 명령을 사용하여 암호를 새로 생성해야 합니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 암호 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateLoginProfile을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM 사용자 **Bob**에 대한 새 임시 암호를 설정하고 사용자가 다음에 로그인할 때 암호를 변경하도록 요구합니다.

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -
PasswordResetRequired $true
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UpdateLoginProfile을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateOpenIdConnectProviderThumbprint** 사용

다음 코드 예제는 `UpdateOpenIdConnectProviderThumbprint`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

기존 서버 인증서 지문 목록을 새 목록으로 바꾸기

이 예제는 ARN이 `arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com`인 OIDC 제공업체에 대한 인증서 지문 목록을 업데이트하여 새 지문을 사용합니다.

```
aws iam update-open-id-connect-provider-thumbprint \
```

```
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com \
    --thumbprint-list 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM에서 OIDC(OpenID Connect) ID 제공업체 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateOpenIdConnectProviderThumbprint를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com**인 OIDC 제공업체에 대한 인증서 지문 목록을 업데이트하여 새 지문을 사용합니다. OIDC 제공업체는 제공업체와 연결된 인증서가 변경될 때 새 값을 공유합니다.

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UpdateOpenIdConnectProviderThumbprint를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateRole** 사용

다음 코드 예제는 UpdateRole의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 역할의 설명 또는 세션 기간 변경

다음 update-role 명령은 IAM 역할 production-role의 설명을 Main production role로 변경하고 최대 세션 기간을 12시간으로 설정합니다.

```
aws iam update-role \
    --role-name production-role \
    --description 'Main production role' \
    --max-session-duration 43200
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 역할 변경을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateRole을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 역할 설명과 역할 세션을 요청할 수 있는 최대 세션 기간 값 (초)을 업데이트합니다.

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UpdateRole을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateRoleDescription** 사용

다음 코드 예제는 UpdateRoleDescription의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 역할의 설명 변경

다음 update-role 명령은 IAM 역할에 대한 설명을 production-role에서 Main production role로 변경합니다.

```
aws iam update-role-description \
    --role-name production-role \
    --description 'Main production role'
```

출력:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "production-role",
        "RoleId": "AROA1234567890EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:role/production-role",
        "CreateDate": "2017-12-06T17:16:37+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::123456789012:root"
                    },
                    "Action": "sts:AssumeRole",
                    "Condition": {}
                }
            ]
        },
        "Description": "Main production role"
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 역할 변경을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateRoleDescription을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 계정의 IAM 역할 설명을 업데이트합니다.

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UpdateRoleDescription을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateSamlProvider** 사용

다음 코드 예제는 UpdateSamlProvider의 사용 방법을 보여 줍니다.

CLI

AWS CLI

기존 SAML 제공업체에 대한 메타데이터 문서 업데이트

이 예제는 ARN이 arn:aws:iam::123456789012:saml-provider/SAMLADFS인 IAM의 SAML 제공업체를 SAMLMetaData.xml 파일의 새 SAML 메타데이터 문서로 업데이트합니다.

```
aws iam update-saml-provider \
    --saml-metadata-document file://SAMLMetaData.xml \
    --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

출력:

```
{
    "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/SAMLADFS"
}
```

자세한 내용은 AWS IAM 사용 설명서의 IAM SAML 자격 증명 공급자 생성을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateSamlProvider를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 ARN이 **arn:aws:iam::123456789012:saml-provider/SAMLADFS**인 IAM의 SAML 제공업체를 **SAMLMetaData.xml** 파일의 새 SAML 메타데이터 문서로 업데이트합니다. JSON 파일의 내용을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다.

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UpdateSamlProvider를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateServerCertificate** 사용

다음 코드 예제는 UpdateServerCertificate의 사용 방법을 보여 줍니다.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::updateServerCertificate(const Aws::String
 &currentCertificateName,
                                           const Aws::String &newCertificateName,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateServerCertificateRequest request;
    request.SetServerCertificateName(currentCertificateName);
    request.SetNewServerCertificateName(newCertificateName);

    auto outcome = iam.UpdateServerCertificate(request);
    bool result = true;
    if (outcome.IsSuccess()) {
        std::cout << "Server certificate " << currentCertificateName
                  << " successfully renamed as " << newCertificateName
                  << std::endl;
    }
```

```
    else {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error changing name of server certificate " <<
                      currentCertificateName << " to " << newCertificateName <<
 ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << currentCertificateName
                      << "' not found." << std::endl;
        }
    }

    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 UpdateServerCertificate를 참조하세요.

CLI

AWS CLI

AWS 계정에 있는 서버 인증서의 경로 또는 이름 변경

다음 `update-server-certificate` 명령은 인증서의 이름을 `myServerCertificate`에서 `myUpdatedServerCertificate`로 변경합니다. 또한 Amazon CloudFront 서비스에서 액세스할 수 있도록 경로를 `/cloudfront/`로 변경합니다. 이 명령은 출력을 생성하지 않습니다. `list-server-certificates` 명령을 실행하여 업데이트 결과를 볼 수 있습니다.

```
aws-iam update-server-certificate \
    --server-certificate-name myServerCertificate \
    --new-server-certificate-name myUpdatedServerCertificate \
    --new-path /cloudfront/
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM에서 서버 인증서 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateServerCertificate를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

서버 인증서를 업데이트합니다.

```
import { UpdateServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} currentName
 * @param {string} newName
 */
export const updateServerCertificate = (currentName, newName) => {
  const command = new UpdateServerCertificateCommand({
    ServerCertificateName: currentName,
    NewServerCertificateName: newName,
  });

  return client.send(command);
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 UpdateServerCertificate를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  ServerCertificateName: "CERTIFICATE_NAME",
  NewServerCertificateName: "NEW_CERTIFICATE_NAME",
};

iam.updateServerCertificate(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 UpdateServerCertificate를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 인증서 이름을 **MyServerCertificate**에서 **MyRenamedServerCertificate**로 변경합니다.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

예제 2: 이 예제는 **MyServerCertificate**라는 인증서를 /Org1/Org2/ 경로로 이동합니다. 그러면 리소스의 ARN이 **arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate**로 변경됩니다.

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UpdateServerCertificate를 참조하세요.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

서버 인증서를 나열하고, 업데이트하고, 삭제합니다.

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "ServerCertificateManager"
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key,
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end
```

```
  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info("No server certificates found.")
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 UpdateServerCertificate를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateSigningCertificate** 사용

다음 코드 예제는 UpdateSigningCertificate의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자의 서명 인증서 활성화 또는 비활성

다음 update-signing-certificate 명령은 Bob이라는 IAM 사용자에 대해 지정된 서명 인증서를 비활성화합니다.

```
aws iam update-signing-certificate \
    --certificate-id TA7SMP42TDN5Z26OBPJE7EXAMPLE \
    --status Inactive \
    --user-name Bob
```

서명 인증서의 ID를 가져오려면 list-signing-certificates 명령을 사용합니다.

자세한 내용은 Amazon EC2 사용 설명서의 서명 인증서 관리를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateSigningCertificate를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 **Bob**이라는 IAM 사용자와 연결되어 있고 인증서 ID가 **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU**인 인증서를 업데이트하여 비활성으로 표시합니다.

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UpdateSigningCertificate를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UpdateUser** 사용

다음 코드 예제는 UpdateUser의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 읽기 전용 및 읽기-쓰기 사용자 생성

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::updateUser(const Aws::String &currentUserName,
                             const Aws::String &newUserName,
                             const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::UpdateUserRequest request;
    request.SetUserName(currentUserName);
    request.SetNewUserName(newUserName);

    auto outcome = iam.UpdateUser(request);
    if (outcome.IsSuccess()) {
        std::cout << "IAM user " << currentUserName <<
                  " successfully updated with new user name " << newUserName <<
                  std::endl;
    }
    else {
        std::cerr << "Error updating user name for IAM user " << currentUserName
 <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
```

```
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 UpdateUser를 참조하십시오.

CLI

AWS CLI

IAM 사용자의 이름 변경

다음 update-user 명령은 IAM 사용자의 이름을 Bob에서 Robert로 변경합니다.

```
aws iam update-user \
    --user-name Bob \
    --new-user-name Robert
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IAM 사용 설명서의 IAM 사용자 그룹 이름 변경을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UpdateUser를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UpdateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <curName> <newName>\s

                Where:
                    curName - The current user name.\s
                    newName - An updated user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String curName = args[0];
        String newName = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateIAMUser(iam, curName, newName);
        System.out.println("Done");
        iam.close();
    }

    public static void updateIAMUser(IamClient iam, String curName, String
 newName) {
        try {
            UpdateUserRequest request = UpdateUserRequest.builder()
                    .userName(curName)
                    .newUserName(newName)
```

```
                    .build();

            iam.updateUser(request);
            System.out.printf("Successfully updated user to username %s",
 newName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateUser를 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

사용자를 업데이트합니다.

```
import { UpdateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} currentUserName
 * @param {string} newUserName
 */
export const updateUser = (currentUserName, newUserName) => {
  const command = new UpdateUserCommand({
    UserName: currentUserName,
    NewUserName: newUserName,
  });
```

```
  return client.send(command);
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 UpdateUser를 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
  NewUserName: process.argv[3],
};

iam.updateUser(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 UpdateUser를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun updateIAMUser(
    curName: String?,
    newName: String?,
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 UpdateUser를 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 IAM 사용자 이름을 **Bob**에서 **Robert**로 변경합니다.

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

예제 2: 이 예제는 IAM 사용자 **Bob**의 경로를 **/Org1/Org2/**로 변경합니다. 그러면 해당 사용자의 ARN이 **arn:aws:iam::123456789012:user/Org1/Org2/bob**으로 효과적으로 변경됩니다.

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UpdateUser를 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 UpdateUser를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Updates an IAM user's name
#
# @param current_name [String] The current name of the user
# @param new_name [String] The new name of the user
def update_user_name(current_name, new_name)
  @iam_client.update_user(user_name: current_name, new_user_name: new_name)
  true
rescue StandardError => e
  @logger.error("Error updating user name from '#{current_name}' to
'#{new_name}': #{e.message}")
  false
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 UpdateUser를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UploadServerCertificate** 사용

다음 코드 예제는 UploadServerCertificate의 사용 방법을 보여 줍니다.

CLI

AWS CLI

AWS 계정에 서버 인증서 업로드

다음 upload-server-certificate 명령은 서버 인증서를 AWS 계정에 업로드합니다. 이 예제에서 인증서는 public_key_cert_file.pem 파일에 있고, 연결된 프라이빗 키

가 my_private_key.pem 파일에 있으며, CA(인증 기관)에서 제공하는 인증서 체인은 my_certificate_chain_file.pem 파일에 있습니다. 파일 업로드가 완료되면 myServerCertificate 이름 아래에서 사용할 수 있습니다. file://로 시작하는 파라미터는 명령에 파일 내용을 읽고 해당 내용을 파일 이름 대신 파라미터 값으로 사용하도록 지시합니다.

```
aws iam upload-server-certificate \
    --server-certificate-name myServerCertificate \
    --certificate-body file://public_key_cert_file.pem \
    --private-key file://my_private_key.pem \
    --certificate-chain file://my_certificate_chain_file.pem
```

출력:

```
{
    "ServerCertificateMetadata": {
        "Path": "/",
        "ServerCertificateName": "myServerCertificate",
        "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
        "Arn": "arn:aws:iam::1234567989012:server-certificate/
myServerCertificate",
        "UploadDate": "2019-04-22T21:13:44+00:00",
        "Expiration": "2019-10-15T22:23:16+00:00"
    }
}
```

자세한 내용은 IAM 사용 설명서의 서버 인증서 생성, 업로드 및 삭제를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UploadServerCertificate를 참조하세요.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import { UploadServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "fs";
```

```
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import * as path from "path";

const client = new IAMClient({});

const certMessage = `Generate a certificate and key with the following command,
 or the equivalent for your system.

openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -nodes \
-keyout example.key -out example.crt -subj "/CN=example.com" \
-addext "subjectAltName=DNS:example.com,DNS:www.example.net,IP:10.0.0.1"
`;

const getCertAndKey = () => {
  try {
    const cert = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.crt"),
    );
    const key = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.key"),
    );
    return { cert, key };
  } catch (err) {
    if (err.code === "ENOENT") {
      throw new Error(
        `Certificate and/or private key not found. ${certMessage}`,
      );
    }

    throw err;
  }
};

/**
 *
 * @param {string} certificateName
 */
export const uploadServerCertificate = (certificateName) => {
  const { cert, key } = getCertAndKey();
  const command = new UploadServerCertificateCommand({
    ServerCertificateName: certificateName,
    CertificateBody: cert.toString(),
    PrivateKey: key.toString(),
  });
```

```
  return client.send(command);
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 UploadServerCertificate을 참조하십시오.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 새 서버 인증서를 IAM 계정에 업로드합니다. 인증서 본문, 프라이빗 키 및 인증서 체인(선택 사항)이 포함된 파일은 모두 PEM 인코딩되어야 합니다. 파라미터에는 파일 이름 대신 파일의 실제 내용이 필요합니다. 파일 내용을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다.

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

출력:

```
Arn                   : arn:aws:iam::123456789012:server-certificate/MyTestCert
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /
ServerCertificateId   : ASCAJIEXAMPLE7J7HQZYW
ServerCertificateName : MyTestCert
UploadDate            : 4/21/2015 11:14:16 AM
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UploadServerCertificate를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **UploadSigningCertificate** 사용

다음 코드 예제는 UploadSigningCertificate의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자의 서명 인증서 업로드

다음 `upload-signing-certificate` 명령은 Bob이라는 IAM 사용자의 서명 인증서를 업로드합니다.

```
aws iam upload-signing-certificate \
    --user-name Bob \
    --certificate-body file://certificate.pem
```

출력:

```
{
    "Certificate": {
        "UserName": "Bob",
        "Status": "Active",
        "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-body>-----END
 CERTIFICATE-----",
        "CertificateId": "TA7SMP42TDN5Z26OBPJE7EXAMPLE",
        "UploadDate": "2013-06-06T21:40:08.121Z"
    }
}
```

인증서는 PEM 형식의 certificate.pem이라는 파일에 있습니다.

자세한 내용은 IAM 사용 설명서의 사용자 서명 인증서 생성 및 업로드를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 UploadSigningCertificate를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 이 예제는 새로운 X.509 서명 인증서를 업로드하고 이를 **Bob**이라는 IAM 사용자와 연결합니다. 인증서 본문이 포함된 파일은 PEM으로 인코딩됩니다. **CertificateBody** 파라미터에는 파일 이름이 아닌 인증서 파일의 실제 내용이 필요합니다. 파일을 성공적으로 처리하려면 **-Raw** 스위치 파라미터를 사용해야 합니다.

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

출력:

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                   YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                   21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                   rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                   nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                   NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                   -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 UploadSigningCertificate를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하는 IAM 시나리오

다음 코드 예제는 AWS SDK를 사용하여 IAM에서 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이러한 시나리오에서는 IAM 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접 호출하여 특정 태스크를 수행하는 방법을 보여줍니다. 각 시나리오에는 전체 소스 코드에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시나리오는 컨텍스트에 맞는 서비스 작업을 이해하는 데 도움이 되도록 중급 수준의 경험을 대상으로 합니다.

예시

- AWS SDK를 사용하여 복원력이 뛰어난 서비스 구축 및 관리
- AWS SDK를 사용하여 IAM 그룹을 생성하고 사용자를 그룹에 추가
- AWS SDK를 사용하여 AWS STS에서 IAM 사용자 생성 및 역할 수임
- AWS SDK를 사용하여 읽기 전용 및 읽기-쓰기 IAM 사용자 생성
- AWS SDK를 사용하여 IAM 액세스 키 관리
- AWS SDK를 사용하여 IAM 정책 관리
- AWS SDK를 사용하여 IAM 역할 관리
- AWS SDK를 사용하여 IAM 계정 관리
- AWS SDK를 사용하여 IAM 정책 버전 롤백
- AWS SDK를 사용하여 IAM 정책 빌더 API 작업

AWS SDK를 사용하여 복원력이 뛰어난 서비스 구축 및 관리

다음 코드 예제는 책, 영화, 노래 추천을 반환하는 로드 밸런싱 웹 서비스를 만드는 방법을 보여줍니다. 이 예제에서는 서비스가 장애에 대응하는 방법과 장애 발생 시 복원력을 높이기 위해 서비스를 재구성하는 방법을 보여줍니다.

- Amazon EC2 Auto Scaling 그룹을 사용하여 시작 템플릿을 기반으로 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 생성하고 인스턴스 수를 지정된 범위 내로 유지합니다.
- Elastic Load Balancing으로 HTTP 요청을 처리하고 배포합니다.
- Auto Scaling 그룹의 인스턴스 상태를 모니터링하고 요청을 정상 인스턴스로만 전달합니다.

- 각 EC2 인스턴스에서 Python 웹 서버를 실행하여 HTTP 요청을 처리합니다. 웹 서버는 추천 및 상태 확인으로 응답합니다.
- Amazon DynamoDB 테이블을 사용하여 추천 서비스를 시뮬레이션합니다.
- AWS Systems Manager 파라미터를 업데이트하여 요청 및 상태 확인에 대한 웹 서버 응답을 제어합니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
```

```
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
```

```
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
```

```
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
```

```
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
```

```
            + "single endpoint where clients connect and dispatches requests to
 instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
 _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
```

```
                    "\nFor this example to work, the default security group for
your default VPC must\n"
                    + "allows access from this computer. You can either add it
automatically from this\n"
                    + "example or add it yourself using the AWS Management
Console.\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
```

```
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
```

```
                        $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
                        "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
```

```
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
```

```
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");
```

```
        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
```

```
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
 might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Auto Scaling과 Amazon EC2 작업을 래핑하는 클래스를 생성합니다.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;
```

```
    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
```

```
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                    "\"Version\": \"2012-10-17\"," +
                                    "\"Statement\": [{" +
                                         "\"Effect\": \"Allow\"," +
                                         "\"Principal\": {" +
                                         "\"Service\": [" +
                                             "\"ec2.amazonaws.com\"" +
                                         "]" +
                                         "}," +
                                    "\"Action\": \"sts:AssumeRole\"" +
                                    "}]" +
                                "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
```

```
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
```

```
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
```

```
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
    /// The launch template specifies a Bash script in its user data field that
runs after
    /// the instance is started. This script installs the Python packages and
starts a Python
    /// web server on the instance.
    /// </summary>
```

```
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       await CreateKeyPair(_keyPairName);
       await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName, _instanceProfileName, instancePolicyPath);

       var startServerText = await File.ReadAllTextAsync(startupScriptPath);
       var plainTextBytes = System.Text.Encoding.UTF8.GetBytes(startServerText);

       var amiLatest = await _amazonSsm.GetParameterAsync(
           new GetParameterRequest() { Name = _amiParam });
       var amiId = amiLatest.Parameter.Value;
       var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
           new CreateLaunchTemplateRequest()
           {
               LaunchTemplateName = _launchTemplateName,
               LaunchTemplateData = new RequestLaunchTemplateData()
               {
                   InstanceType = _instanceType,
                   ImageId = amiId,
                   IamInstanceProfile =
                       new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                       {
                           Name = _instanceProfileName
                       },
                   KeyName = _keyPairName,
                   UserData = System.Convert.ToBase64String(plainTextBytes)
               }
           });
       return launchTemplateResponse.LaunchTemplate;

   }

   /// <summary>
   /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
```

```
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());
        return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
 Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
 size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
```

```
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
            new DescribeVpcsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("is-default", new List<string>() { "true" })
                }
            });
        return vpcResponse.Vpcs[0];
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
        var subnets = new List<Subnet>();
        var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
            new DescribeSubnetsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("vpc-id", new List<string>() { vpcId}),
                    new ("availability-zone", availabilityZones),
                    new ("default-for-az", new List<string>() { "true" })
                }
            });

        // Get the entire list using the paginator.
        await foreach (var subnet in subnetPaginator.Subnets)
        {
```

```
            subnets.Add(subnet);
        }

        return subnets;
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonClientException)
        {
            Console.WriteLine($"Unable to delete template {templateName}.");
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
```

```
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
```

```
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
            new DescribeIamInstanceProfileAssociationsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("instance-id", new List<string>() { instanceId })
                },
            });
        return response.IamInstanceProfileAssociations[0];
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
```

```
   public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
   {
       await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
               new ReplaceIamInstanceProfileAssociationRequest()
               {
                   AssociationId = associationId,
                   IamInstanceProfile = new IamInstanceProfileSpecification()
                   {
                       Name = credsProfileName
                   }
               });
       // Allow time before resetting.
       Thread.Sleep(25000);
       var instanceReady = false;
       var retries = 5;
       while (retries-- > 0 && !instanceReady)
       {
           await _amazonEc2.RebootInstancesAsync(
               new RebootInstancesRequest(new List<string>() { instanceId }));
           Thread.Sleep(10000);

           var instancesPaginator =
_amazonSsm.Paginators.DescribeInstanceInformation(
               new DescribeInstanceInformationRequest());
           // Get the entire list using the paginator.
           await foreach (var instance in
instancesPaginator.InstanceInformationList)
           {
               instanceReady = instance.InstanceId == instanceId;
               if (instanceReady)
               {
                   break;
               }
           }
       }
       Console.WriteLine($"Sending restart command to instance {instanceId}");
       await _amazonSsm.SendCommandAsync(
           new SendCommandRequest()
           {
               InstanceIds = new List<string>() { instanceId },
               DocumentName = "AWS-RunShellScript",
               Parameters = new Dictionary<string, List<string>>()
               {
```

```
                        {"commands", new List<string>() { "cd / && sudo python3
server.py 80" }}
                }
            });
        Console.WriteLine($"Restarted the web server on instance {instanceId}");
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
```

```
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
calling computer.
    /// This can be done by allowing ingress from this computer's IP address.
```

```
   /// In some situations, such as connecting from a corporate network, you must
instead specify
   /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
   /// If you do, be sure to remove public access when you're done.
   /// </summary>
   /// <param name="vpc">The group to check.</param>
   /// <param name="port">The port to verify.</param>
   /// <param name="ipAddress">This computer's IP address.</param>
   /// <returns>True if the ip address is allowed on the group.</returns>
   public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
   {
       var portIsOpen = false;
       foreach (var ipPermission in group.IpPermissions)
       {
           if (ipPermission.FromPort == port)
           {
               foreach (var ipRange in ipPermission.Ipv4Ranges)
               {
                   var cidr = ipRange.CidrIp;
                   if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                   {
                       portIsOpen = true;
                   }
               }

               if (ipPermission.PrefixListIds.Any())
               {
                   portIsOpen = true;
               }

               if (!portIsOpen)
               {
                   Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                     "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
               }
               else
               {
                   break;
               }
           }
```

```
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
 the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }
}
```

Elastic Load Balancing 작업을 래핑하는 클래스를 생성합니다.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
```

```
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
```

```
    public async Task<List<TargetHealthDescription>>
 CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
```

```
   /// <param name="vpcId">The Id of the Vpc in which the load balancer
exists.</param>
   /// <returns>The new TargetGroup object.</returns>
   public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
ProtocolEnum protocol, int port, string vpcId)
   {
       var createResponse = await
_amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
           new CreateTargetGroupRequest()
           {
               Name = groupName,
               Protocol = protocol,
               Port = port,
               HealthCheckPath = "/healthcheck",
               HealthCheckIntervalSeconds = 10,
               HealthCheckTimeoutSeconds = 5,
               HealthyThresholdCount = 2,
               UnhealthyThresholdCount = 2,
               VpcId = vpcId
           });
       var targetGroup = createResponse.TargetGroups[0];
       return targetGroup;
   }

   /// <summary>
   /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
   /// and forwards requests to the specified target group.
   /// </summary>
   /// <param name="name">The name for the new load balancer.</param>
   /// <param name="subnetIds">Subnets for the load balancer.</param>
   /// <param name="targetGroup">Target group for forwarded requests.</param>
   /// <returns>The new LoadBalancer object.</returns>
   public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
   {
       var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
           new CreateLoadBalancerRequest()
           {
               Name = name,
               Subnets = subnetIds
           });
       var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;
```

```
        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
```

```
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
```

```
                    });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn =
 targetArn });
                Console.WriteLine($"Deleted load balancing target group
 {groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

DynamoDB를 사용하여 추천 서비스를 시뮬레이션하는 클래스를 생성합니다.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
```

```
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
```

```
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
_amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
```

```
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await
 File.ReadAllTextAsync(recommendationsPath);
        var records =

 JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Systems Manager 작업을 래핑하는 클래스를 생성합니다.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
```

```
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하십시오.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup

- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
```

```
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException,
 InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
 Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
```

```
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
service.
                To keep things tidy and to avoid unwanted charges on your
account, we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
```

```
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
```

```
                Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged
credentials.

                The template also defines an IAM policy that each instance uses
to assume a role that grants
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
```

```
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
```

```
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and
shows how using a resilient
                        architecture can keep the web service running in spite
of these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                       """);
```

```
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
                        """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        Instead of failing when the recommendation service fails,
the web service can return a static response.
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns
a static response.
                The service still reports as healthy because health checks are
still shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance
id value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
```

```
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");
```

```
        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load
balancing rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance
is running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
```

```
   }

   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
           System.out.println("See the current state of the service by selecting
one of the following choices:");
           for (int i = 0; i < actions.length; i++) {
               System.out.println(i + ": " + actions[i]);
           }

           try {
               System.out.print("\nWhich action would you like to take? ");
               int choice = scanner.nextInt();
               System.out.println("-".repeat(88));

               switch (choice) {
                   case 0 -> {
                       System.out.println("Request:\n");
                       System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                       CloseableHttpClient httpClient =
HttpClients.createDefault();

                       // Create an HTTP GET request to the ELB.
                       HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                       // Execute the request and get the response.
                       HttpResponse response = httpClient.execute(httpGet);
                       int statusCode =
response.getStatusLine().getStatusCode();
                       System.out.println("HTTP Status Code: " + statusCode);

                       // Display the JSON response
                       BufferedReader reader = new BufferedReader(
```

```
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
balancer targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                    target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
```

```
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

Auto Scaling과 Amazon EC2 작업을 래핑하는 클래스를 생성합니다.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
```

```
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile
is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
```

```
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
```

```
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }
```

```
   /**
    * Detaches a role from an instance profile, detaches policies from the role,
    * and deletes all the resources.
    */
   public void deleteInstanceProfile(String roleName, String profileName) {
       try {
           software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                   .builder()
                   .instanceProfileName(profileName)
                   .build();

           GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
           String name = response.instanceProfile().instanceProfileName();
           System.out.println(name);

           RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                   .instanceProfileName(profileName)
                   .roleName(roleName)
                   .build();

           getIAMClient().removeRoleFromInstanceProfile(profileRequest);
           DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                   .instanceProfileName(profileName)
                   .build();

           getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
           System.out.println("Deleted instance profile " + profileName);

           DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                   .roleName(roleName)
                   .build();

           // List attached role policies.
           ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                   .listAttachedRolePolicies(role -> role.roleName(roleName));
           List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
           for (AttachedPolicy attachedPolicy : attachedPolicies) {
```

```
                DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
```

```
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " +
secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " +
ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
```

```
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
```

```
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
```

```
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
```

```
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }
```

```
   // Gets data about the profile associated with an instance.
   public String getInstanceProfile(String instanceId) {
       Filter filter = Filter.builder()
               .name("instance-id")
               .values(instanceId)
               .build();

       DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
               .builder()
               .filters(filter)
               .build();

       DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
               .describeIamInstanceProfileAssociations(associationsRequest);
       return response.iamInstanceProfileAssociations().get(0).associationId();
   }

   public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
       ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
       ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
       for (Policy policy : listPoliciesResponse.policies()) {
           if (policy.policyName().equals(policyName)) {
               // List the entities (users, groups, roles) that are attached to
the policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                       .builder()
                       .policyArn(policy.arn())
                       .build();
               ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                       .listEntitiesForPolicy(listEntitiesRequest);
               if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                       || !listEntitiesResponse.policyRoles().isEmpty()) {
                   // Detach the policy from any entities it is attached to.
                   DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
```

```
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
```

```
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Elastic Load Balancing 작업을 래핑하는 클래스를 생성합니다.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
 targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()

 .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
```

```
                .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
```

```
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load
balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }
```

```
        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how
instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
```

```
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
```

```
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .defaultActions(action)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

DynamoDB를 사용하여 추천 서비스를 시뮬레이션하는 클래스를 생성합니다.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
```

```
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
```

```
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws
IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
```

```
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
 enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Systems Manager 작업을 래핑하는 클래스를 생성합니다.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }
```

```
    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances

- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0


import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
```

```
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios);
}
```

모든 리소스를 배포하기 위한 단계를 생성합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
```

```
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
```

```
import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
```

```
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
```

```
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
```

```
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
```

```
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
```

```
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    // snippet-start:[javascript.v3.wkflw.resilient.CreateLaunchTemplate]
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
      // snippet-end:[javascript.v3.wkflw.resilient.CreateLaunchTemplate]
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
```

```
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.DescribeVpcs]
    const client = new EC2Client({});
```

```
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.DescribeVpcs]
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.DescribeSubnets]
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.DescribeSubnets]
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.CreateTargetGroup]
```

```
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.CreateTargetGroup]
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.CreateLoadBalancer]
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
```

```
      { Names: [NAMES.loadBalancerName] },
    );
    // snippet-end:[javascript.v3.wkflw.resilient.CreateLoadBalancer]
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.CreateListener]
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.CreateListener]
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
```

```
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.AttachTargetGroup]
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.AttachTargetGroup]
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
```

```
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      } else {
        return MESSAGES.noIpRules;
      }
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      } else {
        return MESSAGES.noIpRules;
      }
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
```

```
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    } else {
      return false;
    }
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
```

```
        state.endpointResponse,
      );
    }
  }),
];
```

데모를 실행하기 위한 단계를 생성합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
```

```
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.DescribeTargetGroups]
  const client = new ElasticLoadBalancingV2Client({});
```

```
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.DescribeTargetGroups]

  // snippet-start:[javascript.v3.wkflw.resilient.DescribeTargetHealth]
  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.DescribeTargetHealth]
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
```

```
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
```

```
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
```

```
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      // snippet-start:
[javascript.v3.wkflw.resilient.DescribeIamInstanceProfileAssociations]
      const ec2Client = new EC2Client({});
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
      // snippet-end:
[javascript.v3.wkflw.resilient.DescribeIamInstanceProfileAssociations]
```

```
      state.instanceProfileAssociationId =
        IamInstanceProfileAssociations[0].AssociationId;
      // snippet-start:
[javascript.v3.wkflw.resilient.ReplaceIamInstanceProfileAssociation]
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );
      // snippet-end:
[javascript.v3.wkflw.resilient.ReplaceIamInstanceProfileAssociation]

      await ec2Client.send(
        new RebootInstancesCommand({
          InstanceIds: [state.targetInstance.InstanceId],
        }),
      );

      const ssmClient = new SSMClient({});
      await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
        const { InstanceInformationList } = await ssmClient.send(
          new DescribeInstanceInformationCommand({}),
        );

        const instance = InstanceInformationList.find(
          (info) => info.InstanceId === state.targetInstance.InstanceId,
        );

        if (!instance) {
          throw new Error("Instance not found.");
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
```

```
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
```

```
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
```

```
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
```

```
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  // snippet-start:[javascript.v3.wkflw.resilient.CreateInstanceProfile]
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  // snippet-end:[javascript.v3.wkflw.resilient.CreateInstanceProfile]
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
```

```
  );

  return InstanceProfile;
}
```

모든 리소스를 폐기하는 단계를 생성합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
```

```
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    } else {
      return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
    }
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
```

```
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    } else {
      return MESSAGES.deletedKeyPair.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
```

```
    } else {
      return MESSAGES.detachedPolicyFromRole
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    } else {
      return MESSAGES.deletedPolicy.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
```

```
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    } else {
      return MESSAGES.removedRoleFromInstanceProfile
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    } else {
      return MESSAGES.deletedInstanceRole.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
```

```
    try {
      // snippet-start:[javascript.v3.wkflw.resilient.DeleteInstanceProfile]
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
      // snippet-end:[javascript.v3.wkflw.resilient.DeleteInstanceProfile]
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    } else {
      return MESSAGES.deletedInstanceProfile.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
```

```
    } else {
      return MESSAGES.deletedAutoScalingGroup.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      // snippet-start:[javascript.v3.wkflw.resilient.DeleteLaunchTemplate]
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
      // snippet-end:[javascript.v3.wkflw.resilient.DeleteLaunchTemplate]
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    } else {
      return MESSAGES.deletedLaunchTemplate.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      // snippet-start:[javascript.v3.wkflw.resilient.DeleteLoadBalancer]
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
```

```
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
      // snippet-end:[javascript.v3.wkflw.resilient.DeleteLoadBalancer]
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    } else {
      return MESSAGES.deletedLoadBalancer.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.DeleteTargetGroup]
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
```

```
      state.deleteLoadBalancerTargetGroupError = e;
    }
    // snippet-end:[javascript.v3.wkflw.resilient.DeleteTargetGroup]
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    } else {
      return MESSAGES.deletedLoadBalancerTargetGroup.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    } else {
      return MESSAGES.detachedSsmOnlyRoleFromProfile
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
```

```
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    } else {
      return MESSAGES.detachedSsmOnlyCustomRolePolicy
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
```

```
    } else {
      return MESSAGES.detachedSsmOnlyAWSRolePolicy
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    } else {
      return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
```

```
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    } else {
      return MESSAGES.deletedSsmOnlyPolicy.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    } else {
      return MESSAGES.deletedSsmOnlyRole.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
  }),
];

/**
```

```
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    } else {
      console.log(err.name);
      throw err;
    }
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
```

```
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 다음 주제를 참조하십시오.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile

- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
class Runner:
    def __init__(
        self, resource_path, recommendation, autoscaler, loadbalancer,
 param_helper
    ):
        self.resource_path = resource_path
```

```
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

    def deploy(self):
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        print(
            "\nFor this demo, we'll use the AWS SDK for Python (Boto3) to create
several AWS resources\n"
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n"
            "against various kinds of failures.\n\n"
            "Some of the resources create by this demo are:\n"
        )
        print(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations."
        )
        print(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server."
        )
        print(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones."
        )
        print(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests."
        )
        print("-" * 88)
        q.ask("Press Enter when you're ready to start deploying resources.")

        print(
            f"Creating and populating a DynamoDB table named
'{self.recommendation.table_name}'."
        )
```

```
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)
        print("-" * 88)

        print(
            f"Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.\n"
            f"This script starts a Python web server defined in the `server.py`
script. The web server\n"
            f"listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.\n"
            f"For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            f"run a web server, such as Apache, with least-privileged
credentials.\n"
        )
        print(
            f"The template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            f"permissions to access the DynamoDB recommendation table and Systems
Manager parameters\n"
            f"that control the flow of the demo.\n"
        )
        self.autoscaler.create_template(startup_script, instance_policy)
        print("-" * 88)

        print(
            f"Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            f"Availability Zone."
        )
        zones = self.autoscaler.create_group(3)
        print("-" * 88)
        print(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            "HTTP requests. You can see these instances in the console or
continue with the demo."
        )
        print("-" * 88)
        q.ask("Press Enter when you're ready to continue.")

        print(f"Creating variables that control the flow of the demo.\n")
        self.param_helper.reset()
```

```
        print(
            "\nCreating an Elastic Load Balancing target group and load balancer.
 The target group\n"
            "defines how the load balancer connects to instances. The load
 balancer provides a\n"
            "single endpoint where clients connect and dispatches requests to
 instances in the group.\n"
        )
        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            [subnet["SubnetId"] for subnet in subnets], target_group
        )
        self.autoscaler.attach_load_balancer_target_group(target_group)
        print(f"Verifying access to the load balancer endpoint...")
        lb_success = self.loadbalancer.verify_load_balancer_endpoint()
        if not lb_success:
            print(
                "Couldn't connect to the load balancer, verifying that the port
 is open..."
            )
            current_ip_address = requests.get(
                "http://checkip.amazonaws.com"
            ).text.strip()
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                print(
                    "For this example to work, the default security group for
 your default VPC must\n"
                    "allows access from this computer. You can either add it
 automatically from this\n"
                    "example or add it yourself using the AWS Management Console.
\n"
                )
                if q.ask(
```

```
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
 from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success = self.loadbalancer.verify_load_balancer_endpoint()
        if lb_success:
            print("Your load balancer is ready. You can access it by browsing to:
\n")
            print(f"\thttp://{self.loadbalancer.endpoint()}\n")
        else:
            print(
                "Couldn't get a successful response from the load balancer
 endpoint. Troubleshoot by\n"
                "manually verifying that your VPC and security group are
 configured correctly and that\n"
                "you can successfully make a GET request to the load balancer
 endpoint:\n"
            )
            print(f"\thttp://{self.loadbalancer.endpoint()}\n")
        print("-" * 88)
        q.ask("Press Enter when you're ready to continue with the demo.")

    def demo_choices(self):
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
```

```
        choice = 0
        while choice != 2:
            print("-" * 88)
            print(
                "\nSee the current state of the service by selecting one of the
following choices:\n"
            )
            choice = q.choose("\nWhich action would you like to take? ", actions)
            print("-" * 88)
            if choice == 0:
                print("Request:\n")
                print(f"GET http://{self.loadbalancer.endpoint()}")
                response = requests.get(f"http://{self.loadbalancer.endpoint()}")
                print("\nResponse:\n")
                print(f"{response.status_code}")
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                print("\nChecking the health of load balancer targets:\n")
                health = self.loadbalancer.check_target_health()
                for target in health:
                    state = target["TargetHealth"]["State"]
                    print(
                        f"\tTarget {target['Target']['Id']} on port
{target['Target']['Port']} is {state}"
                    )
                    if state != "healthy":
                        print(
                            f"\t\t{target['TargetHealth']['Reason']}:
{target['TargetHealth']['Description']}\n"
                        )
                print(
                    f"\nNote that it can take a minute or two for the health
check to update\n"
                    f"after changes are made.\n"
                )
            elif choice == 2:
                print("\nOkay, let's move on.")
                print("-" * 88)

    def demo(self):
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        print("\nResetting parameters to starting values for demo.\n")
```

```
        self.param_helper.reset()

        print(
            "\nThis part of the demonstration shows how to toggle different parts
of the system\n"
            "to create situations where the web service fails, and shows how
using a resilient\n"
            "architecture can keep the web service running in spite of these
failures."
        )
        print("-" * 88)

        print(
            "At the start, the load balancer endpoint returns recommendations and
reports that all targets are healthy."
        )
        self.demo_choices()

        print(
            f"The web service running on the EC2 instances gets recommendations
by querying a DynamoDB table.\n"
            f"The table name is contained in a Systems Manager parameter named
'{self.param_helper.table}'.\n"
            f"To simulate a failure of the recommendation service, let's set this
parameter to name a non-existent table.\n"
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        print(
            "\nNow, sending a GET request to the load balancer endpoint returns a
failure code. But, the service reports as\n"
            "healthy to the load balancer because shallow health checks don't
check for failure of the recommendation service."
        )
        self.demo_choices()

        print(
            f"Instead of failing when the recommendation service fails, the web
service can return a static response.\n"
            f"While this is not a perfect solution, it presents the customer with
a somewhat better experience than failure.\n"
        )
        self.param_helper.put(self.param_helper.failure_response, "static")
        print(
```

```
            f"\nNow, sending a GET request to the load balancer endpoint returns
a static response.\n"
            f"The service still reports as healthy because health checks are
still shallow.\n"
        )
        self.demo_choices()

        print("Let's reinstate the recommendation service.\n")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)
        print(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n"
            "access the DynamoDB recommendation table.\n"
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        print(
            f"\nReplacing the profile for instance {bad_instance_id} with a
profile that contains\n"
            f"bad credentials...\n"
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        print(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n"
            "depending on which instance is selected by the load balancer.\n"
        )
        self.demo_choices()

        print(
```

```
            "\nLet's implement a deep health check. For this demo, a deep health
check tests whether\n"
            "the web service can access the DynamoDB table that it depends on for
recommendations. Note that\n"
            "the deep health check is only for ELB routing and not for Auto
Scaling instance health.\n"
            "This kind of deep health check is not recommended for Auto Scaling
instance health, because it\n"
            "risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.\n"
        )
        print(
            "By implementing deep health checks, the load balancer can detect
when one of the instances is failing\n"
            "and take that instance out of rotation.\n"
        )
        self.param_helper.put(self.param_helper.health_check, "deep")
        print(
            f"\nNow, checking target health indicates that the instance with bad
credentials ({bad_instance_id})\n"
            f"is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy \n"
            f"instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because\n"
            "the load balancer takes unhealthy instances out of its rotation.\n"
        )
        self.demo_choices()

        print(
            "\nBecause the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy\n"
            "instance is to terminate it and let the auto scaler start a new
instance to replace it.\n"
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        print(
            "\nEven while the instance is terminating and the new instance is
starting, sending a GET\n"
            "request to the web service continues to get a successful
recommendation response because\n"
            "the load balancer routes requests to the healthy instances. After
the replacement instance\n"
            "starts and reports as healthy, it is included in the load balancing
rotation.\n"
```

```
            "\nNote that terminating and replacing an instance typically takes
several minutes, during which time you\n"
            "can see the changing health check status until the new instance is
running and healthy.\n"
        )
        self.demo_choices()

        print(
            "\nIf the recommendation service fails now, deep health checks mean
all instances report as unhealthy.\n"
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        print(
            "\nWhen all instances are unhealthy, the load balancer continues to
route requests even to\n"
            "unhealthy instances, allowing them to fail open and return a static
response rather than fail\n"
            "closed and report failure to the customer."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self):
        print(
            "This concludes the demo of how to build and manage a resilient
service.\n"
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n"
            "that were created for this demo."
        )
        if q.ask("Do you want to clean up all demo resources? (y/n) ",
q.is_yesno):
            self.loadbalancer.delete_load_balancer()
            self.loadbalancer.delete_target_group()
            self.autoscaler.delete_group()
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            self.recommendation.destroy()
        else:
            print(
```

```
                "Okay, we'll leave the resources intact.\n"
                "Don't forget to delete them when you're done with them or you
 might incur unexpected charges."
            )


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../workflows/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
 policies and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    print("-" * 88)
    print(
        "Welcome to the demonstration of How to Build and Manage a Resilient
 Service!"
    )
    print("-" * 88)

    prefix = "doc-example-resilience"
    recommendation = RecommendationService.from_client(
        "doc-example-recommendation-service"
    )
    autoscaler = AutoScaler.from_client(prefix)
    loadbalancer = LoadBalancer.from_client(prefix)
    param_helper = ParameterHelper.from_client(recommendation.table_name)
    runner = Runner(
        args.resource_path, recommendation, autoscaler, loadbalancer,
 param_helper
    )
```

```
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    print("-" * 88)
    print("Thanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Auto Scaling과 Amazon EC2 작업을 래핑하는 클래스를 생성합니다.

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is
```

```
                        created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    @classmethod
    def from_client(cls, resource_prefix):
        """
        Creates this class from Boto3 clients.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        """
        as_client = boto3.client("autoscaling")
        ec2_client = boto3.client("ec2")
        ssm_client = boto3.client("ssm")
        iam_client = boto3.client("iam")
        return cls(
            resource_prefix,
            "t3.micro",
            "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
            as_client,
            ec2_client,
            ssm_client,
            iam_client,
        )
```

```
   def create_instance_profile(
       self, policy_file, policy_name, role_name, profile_name,
aws_managed_policies=()
   ):
       """
       Creates a policy, role, and profile that is associated with instances
created by
       this class. An instance's associated profile defines a role that is
assumed by the
       instance. The role has attached policies that specify the AWS permissions
granted to
       clients that run on the instance.

       :param policy_file: The name of a JSON file that contains the policy
definition to
                           create and attach to the role.
       :param policy_name: The name to give the created policy.
       :param role_name: The name to give the created role.
       :param profile_name: The name to the created profile.
       :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                    the role, such as
AmazonSSMManagedInstanceCore to grant
                                    use of Systems Manager to send commands to
the instance.
       :return: The ARN of the profile that is created.
       """
       assume_role_doc = {
           "Version": "2012-10-17",
           "Statement": [
               {
                   "Effect": "Allow",
                   "Principal": {"Service": "ec2.amazonaws.com"},
                   "Action": "sts:AssumeRole",
               }
           ],
       }
       with open(policy_file) as file:
           instance_policy_doc = file.read()

       policy_arn = None
       try:
           pol_response = self.iam_client.create_policy(
```

```
                PolicyName=policy_name, PolicyDocument=instance_policy_doc
            )
            policy_arn = pol_response["Policy"]["Arn"]
            log.info("Created policy with ARN %s.", policy_arn)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                log.info("Policy %s already exists, nothing to do.", policy_name)
                list_pol_response = self.iam_client.list_policies(Scope="Local")
                for pol in list_pol_response["Policies"]:
                    if pol["PolicyName"] == policy_name:
                        policy_arn = pol["Arn"]
                        break
            if policy_arn is None:
                raise AutoScalerError(f"Couldn't create policy {policy_name}:
{err}")

        try:
            self.iam_client.create_role(
                RoleName=role_name,
AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            self.iam_client.attach_role_policy(RoleName=role_name,
PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info("Created role %s and attached policy %s.", role_name,
policy_arn)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                log.info("Role %s already exists, nothing to do.", role_name)
            else:
                raise AutoScalerError(f"Couldn't create role {role_name}: {err}")

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
```

```
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            else:
                raise AutoScalerError(
                    f"Couldn't create profile {profile_name} and attach it to
role\n"
                    f"{role_name}: {err}"
                )
        return profile_arn


    def get_instance_profile(self, instance_id):
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't get instance profile association for instance
{instance_id}: {err}"
            )
        else:
            return response["IamInstanceProfileAssociations"][0]
```

```
    def replace_instance_profile(
        self, instance_id, new_instance_profile_name, profile_association_id
    ):
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to update.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)
            inst_ready = False
            tries = 0
            while not inst_ready:
                if tries % 6 == 0:
                    self.ec2_client.reboot_instances(InstanceIds=[instance_id])
                    log.info(
                        "Rebooting instance %s and waiting for it to to be
ready.",
                        instance_id,
                    )
                tries += 1
                time.sleep(10)
                response = self.ssm_client.describe_instance_information()
```

```
                for info in response["InstanceInformationList"]:
                    if info["InstanceId"] == instance_id:
                        inst_ready = True
            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info("Restarted the Python web server on instance %s.",
instance_id)
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't replace instance profile for association
{profile_association_id}: {err}"
            )


    def delete_instance_profile(self, profile_name, role_name):
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
```

```
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )
            else:
                raise AutoScalerError(
                    f"Couldn't delete instance profile {profile_name} or detach "
                    f"policies and delete role {role_name}: {err}"
                )


    def create_key_pair(self, key_pair_name):
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        :return: The newly created key pair.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            raise AutoScalerError(f"Couldn't create key pair {key_pair_name}:
{err}")


    def delete_key_pair(self):
        """
        Deletes a key pair.

        :param key_pair_name: The name of the key pair to delete.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            raise AutoScalerError(
```

```
                f"Couldn't delete key pair {self.key_pair_name}: {err}"
            )
        except FileNotFoundError:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
        except PermissionError:
            log.info(
                "Inadequate permissions to delete key pair %s.",
self.key_pair_name
            )
        except Exception as err:
            raise AutoScalerError(
                f"Couldn't delete key pair {self.key_pair_name}: {err}"
            )


    def create_template(self, server_startup_script_file, instance_policy_file):
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )
```

```
            with open(server_startup_script_file) as file:
                start_server_script = file.read()
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                "Created launch template %s for AMI %s on %s.",
                self.launch_template_name,
                ami_id,
                self.inst_type,
            )
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.AlreadyExistsException"
            ):
                log.info(
                    "Launch template %s already exists, nothing to do.",
                    self.launch_template_name,
                )
            else:
                raise AutoScalerError(
                    f"Couldn't create launch template
{self.launch_template_name}: {err}."
                )
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
```

```
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            else:
                raise AutoScalerError(
                    f"Couldn't delete launch template
{self.launch_template_name}: {err}."
                )


    def get_availability_zones(self):
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
        except ClientError as err:
            raise AutoScalerError(f"Couldn't get availability zones: {err}.")
        else:
            return zones


    def create_group(self, group_size):
        """
        Creates an EC2 Auto Scaling group with the specified size.
```

```
        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        zones = []
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                "Created EC2 Auto Scaling group %s with availability zones %s.",
                self.launch_template_name,
                zones,
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExists":
                log.info(
                    "EC2 Auto Scaling group %s already exists, nothing to do.",
                    self.group_name,
                )
            else:
                raise AutoScalerError(
                    f"Couldn't create EC2 Auto Scaling group {self.group_name}:
{err}"
                )
        return zones


    def get_instances(self):
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: Data about the instances.
        """
        try:
```

```
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't get instances for Auto Scaling group
{self.group_name}: {err}"
            )
        else:
            return instance_ids


    def terminate_instance(self, instance_id):
        """
        Terminates and instances in an EC2 Auto Scaling group. After an instance
is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id, ShouldDecrementDesiredCapacity=False
            )
            log.info("Terminated instance %s.", instance_id)
        except ClientError as err:
            raise AutoScalerError(f"Couldn't terminate instance {instance_id}:
{err}")

    def attach_load_balancer_target_group(self, lb_target_group):
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forward requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
```

```
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't attach load balancer target group
{lb_target_group['TargetGroupName']}\n"
                f"to auto scaling group {self.group_name}"
            )


    def _try_terminate_instance(self, inst_id):
        stopping = False
        log.info(f"Stopping {inst_id}.")
        while not stopping:
            try:
                self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                    InstanceId=inst_id, ShouldDecrementDesiredCapacity=True
                )
                stopping = True
            except ClientError as err:
                if err.response["Error"]["Code"] == "ScalingActivityInProgress":
                    log.info("Scaling activity in progress for %s. Waiting...",
inst_id)
                    time.sleep(10)
                else:
                    raise AutoScalerError(f"Couldn't stop instance {inst_id}:
{err}.")

    def _try_delete_group(self):
        """
        Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
        the function waits and retries until the group is successfully deleted.
        """
        stopped = False
        while not stopped:
```

```
            try:
                self.autoscaling_client.delete_auto_scaling_group(
                    AutoScalingGroupName=self.group_name
                )
                stopped = True
                log.info("Deleted EC2 Auto Scaling group %s.", self.group_name)
            except ClientError as err:
                if (
                    err.response["Error"]["Code"] == "ResourceInUse"
                    or err.response["Error"]["Code"] ==
 "ScalingActivityInProgress"
                ):
                    log.info(
                        "Some instances are still running. Waiting for them to
 stop..."
                    )
                    time.sleep(10)
                else:
                    raise AutoScalerError(
                        f"Couldn't delete group {self.group_name}: {err}."
                    )

    def delete_group(self):
        """
        Terminates all instances in the group, deletes the EC2 Auto Scaling
 group.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=self.group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self._try_terminate_instance(inst_id)
                self._try_delete_group()
            else:
                log.info("No groups found named %s, nothing to do.",
 self.group_name)
```

```
        except ClientError as err:
            raise AutoScalerError(f"Couldn't delete group {self.group_name}:
{err}.")


    def get_default_vpc(self):
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            raise AutoScalerError(f"Couldn't get default VPC: {err}")
        else:
            return response["Vpcs"][0]


    def verify_inbound_port(self, vpc, port, ip_address):
        """
        Verify the default security group of the specified VPC allows ingress
from this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
you
        must instead specify a prefix list ID. You can also temporarily open the
port to
        any IP address while running this example. If you do, be sure to remove
public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specific VPC, and a value that
indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
```

```
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info("Found default security group %s.", sec_group["GroupId"])
            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info("Found inbound rule: %s", ip_perm)
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            "The inbound rule does not appear to be open to
either this computer's IP\n"
                            "address of %s, to all IP addresses (0.0.0.0/0), or
to a prefix list ID.",
                            ip_address,
                        )
                    else:
                        break
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't verify inbound rule for port {port} for VPC
{vpc['VpcId']}: {err}"
            )
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id, port, ip_address):
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
```

```
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't authorize ingress to {sec_group_id} on port {port}
from {ip_address}: {err}"
            )


    def get_subnets(self, vpc_id, zones):
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        try:
            response = self.ec2_client.describe_subnets(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )
            subnets = response["Subnets"]
            log.info("Found %s subnets for the specified zones.", len(subnets))
        except ClientError as err:
            raise AutoScalerError(f"Couldn't get subnets: {err}")
        else:
```

```
            return subnets
```

Elastic Load Balancing 작업을 래핑하는 클래스를 생성합니다.

```
class LoadBalancer:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, target_group_name, load_balancer_name, elb_client):
        """
        :param target_group_name: The name of the target group associated with
 the load balancer.
        :param load_balancer_name: The name of the load balancer.
        :param elb_client: A Boto3 Elastic Load Balancing client.
        """
        self.target_group_name = target_group_name
        self.load_balancer_name = load_balancer_name
        self.elb_client = elb_client
        self._endpoint = None


    @classmethod
    def from_client(cls, resource_prefix):
        """
        Creates this class from a Boto3 client.

        :param resource_prefix: The prefix to give to AWS resources created by
 this class.
        """
        elb_client = boto3.client("elbv2")
        return cls(f"{resource_prefix}-tg", f"{resource_prefix}-lb", elb_client)

    def endpoint(self):
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        if self._endpoint is None:
            try:
```

```
                response = self.elb_client.describe_load_balancers(
                    Names=[self.load_balancer_name]
                )
                self._endpoint = response["LoadBalancers"][0]["DNSName"]
            except ClientError as err:
                raise LoadBalancerError(
                    f"Couldn't get the endpoint for load balancer
{self.load_balancer_name}: {err}"
                )
        return self._endpoint


    def create_target_group(self, protocol, port, vpc_id):
        """
        Creates an Elastic Load Balancing target group. The target group
specifies how
        the load balancer forward requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
times and
        lower thresholds. In production, you might want to decrease the
sensitivity of
        your health checks to avoid unwanted failures.

        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=self.target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
```

```
            log.info("Created load balancing target group %s.",
self.target_group_name)
        except ClientError as err:
            raise LoadBalancerError(
                f"Couldn't create load balancing target group
{self.target_group_name}: {err}"
            )
        else:
            return target_group


    def delete_target_group(self):
        """
        Deletes the target group.
        """
        done = False
        while not done:
            try:
                response = self.elb_client.describe_target_groups(
                    Names=[self.target_group_name]
                )
                tg_arn = response["TargetGroups"][0]["TargetGroupArn"]
                self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
                log.info(
                    "Deleted load balancing target group %s.",
self.target_group_name
                )
                done = True
            except ClientError as err:
                if err.response["Error"]["Code"] == "TargetGroupNotFound":
                    log.info(
                        "Load balancer target group %s not found, nothing to
do.",
                        self.target_group_name,
                    )
                    done = True
                elif err.response["Error"]["Code"] == "ResourceInUse":
                    log.info(
                        "Target group not yet released from load balancer,
waiting..."
                    )
                    time.sleep(10)
                else:
                    raise LoadBalancerError(
```

```
                        f"Couldn't delete load balancing target group
{self.target_group_name}: {err}"
                    )


    def create_load_balancer(self, subnet_ids, target_group):
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param subnet_ids: A list of subnets to associate with the load balancer.
        :param target_group: An existing target group that is added as a listener
to the
                             load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=self.load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info("Created load balancer %s.", self.load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info("Waiting for load balancer to be available...")
            waiter.wait(Names=[self.load_balancer_name])
            log.info("Load balancer is available!")
            self.elb_client.create_listener(
                LoadBalancerArn=load_balancer["LoadBalancerArn"],
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                "Created listener to forward traffic from load balancer %s to
target group %s.",
                self.load_balancer_name,
                target_group["TargetGroupName"],
            )
```

```
        except ClientError as err:
            raise LoadBalancerError(
                f"Failed to create load balancer {self.load_balancer_name}"
                f"and add a listener for target group
{target_group['TargetGroupName']}: {err}"
            )
        else:
            self._endpoint = load_balancer["DNSName"]
            return load_balancer


    def delete_load_balancer(self):
        """
        Deletes a load balancer.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[self.load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", self.load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[self.load_balancer_name])
        except ClientError as err:
            if err.response["Error"]["Code"] == "LoadBalancerNotFound":
                log.info(
                    "Load balancer %s does not exist, nothing to do.",
                    self.load_balancer_name,
                )
            else:
                raise LoadBalancerError(
                    f"Couldn't delete load balancer {self.load_balancer_name}:
{err}"
                )


    def verify_load_balancer_endpoint(self):
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.
        """
        success = False
```

```
        retries = 3
        while not success and retries > 0:
            try:
                lb_response = requests.get(f"http://{self.endpoint()}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    success = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return success

    def check_target_health(self):
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[self.target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            raise LoadBalancerError(
                f"Couldn't check health of {self.target_group_name} targets:
{err}"
            )
        else:
            return health_response["TargetHealthDescriptions"]
```

DynamoDB를 사용하여 추천 서비스를 시뮬레이션하는 클래스를 생성합니다.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """

    def __init__(self, table_name, dynamodb_client):
        """
        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    @classmethod
    def from_client(cls, table_name):
        """
        Creates this class from a Boto3 client.

        :param table_name: The name of the DynamoDB recommendations table.
        """
        ddb_client = boto3.client("dynamodb")
        return cls(table_name, ddb_client)

    def create(self):
        """
        Creates a DynamoDB table to use a recommendation service. The table has a
        hash key named 'MediaType' that defines the type of media recommended,
 such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
 MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
```

```
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be do.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file):
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
```

```
                self.table_name, f"Couldn't populate table from {data_file}:
 {err}"
            )

    def destroy(self):
        """
        Deletes the recommendations table.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.",
 self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Systems Manager 작업을 래핑하는 클래스를 생성합니다.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table = "doc-example-resilient-architecture-table"
    failure_response = "doc-example-resilient-architecture-failure-response"
    health_check = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name, ssm_client):
        """
```

```
        :param table_name: The name of the DynamoDB table that is used as a
recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    @classmethod
    def from_client(cls, table_name):
        ssm_client = boto3.client("ssm")
        return cls(table_name, ssm_client)

    def reset(self):
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name, value):
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting demo parameter %s to '%s'.", name, value)
        except ClientError as err:
            raise ParameterHelperError(
                f"Couldn't set parameter {name} to {value}: {err}"
            )
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 IAM 그룹을 생성하고 사용자를 그룹에 추가

다음 코드 예시는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 그룹을 생성하고 모든 Amazon S3 액세스 권한을 그룹에 부여합니다.
- Amazon S3에 액세스할 권한이 없는 새 사용자를 생성합니다.
- 사용자를 그룹에 추가하고 이제 사용자에게 Amazon S3에 대한 권한이 있음을 표시한 다음 리소스를 정리합니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


namespace IAMActions;

public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;

    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
```

```
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
    /// Add an existing IAM user to an existing IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to add.</param>
    /// <param name="groupName">The name of the group to add the user to.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AddUserToGroupAsync(string userName, string
 groupName)
    {
        var response = await _IAMService.AddUserToGroupAsync(new
 AddUserToGroupRequest
        {
            GroupName = groupName,
            UserName = userName,
        });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }



    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
 CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }


    /// <summary>
    /// Create an IAM group.
    /// </summary>
    /// <param name="groupName">The name to give the IAM group.</param>
    /// <returns>The IAM group that was created.</returns>
    public async Task<Group> CreateGroupAsync(string groupName)
    {
        var response = await _IAMService.CreateGroupAsync(new CreateGroupRequest
 { GroupName = groupName });
        return response.Group;
    }


    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
```

```
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }


    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
 rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }


    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
```

```
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }


    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
```

```
    /// Delete an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupAsync(string groupName)
    {
        var response = await _IAMService.DeleteGroupAsync(new DeleteGroupRequest
{ GroupName = groupName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM policy associated with an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group associated with the
    /// policy.</param>
    /// <param name="policyName">The name of the policy to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupPolicyAsync(string groupName, string
policyName)
    {
        var request = new DeleteGroupPolicyRequest()
        {
            GroupName = groupName,
            PolicyName = policyName,
        };

        var response = await _IAMService.DeleteGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
```

```
    }


    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
 { RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
```

```
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
 GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }


    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }


    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }
```

```
    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }


    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();
```

```
        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }


    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }


    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
```

```
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }


    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }


    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }


    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
```

```
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Remove a user from an IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to remove.</param>
    /// <param name="groupName">The name of the IAM group to remove the user
from.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> RemoveUserFromGroupAsync(string userName, string
groupName)
    {
        // Remove the user from the group.
        var removeUserRequest = new RemoveUserFromGroupRequest()
        {
            UserName = userName,
            GroupName = groupName,
        };

        var response = await
_IAMService.RemoveUserFromGroupAsync(removeUserRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
```

```
    public async Task<bool> PutGroupPolicyAsync(string groupName, string
 policyName, string policyDocument)
    {
        var request = new PutGroupPolicyRequest
        {
            GroupName = groupName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM user.
    /// </summary>
    /// <param name="userName">The name of the IAM user.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
```

```
   /// <param name="policyDocument">The policy document defining the IAM
policy.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<bool> PutUserPolicyAsync(string userName, string
policyName, string policyDocument)
   {
       var request = new PutUserPolicyRequest
       {
           UserName = userName,
           PolicyName = policyName,
           PolicyDocument = policyDocument
       };

       var response = await _IAMService.PutUserPolicyAsync(request);
       return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
   }

   /// <summary>
   /// Wait for a new access key to be ready to use.
   /// </summary>
   /// <param name="accessKeyId">The Id of the access key.</param>
   /// <returns>A boolean value indicating the success of the action.</returns>
   public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
   {
       var keyReady = false;

       do
       {
           try
           {
               var response = await _IAMService.GetAccessKeyLastUsedAsync(
                   new GetAccessKeyLastUsedRequest { AccessKeyId =
accessKeyId });
               if (response.UserName is not null)
               {
                   keyReady = true;
               }
           }
           catch (NoSuchEntityException)
           {
               keyReady = false;
           }
       } while (!keyReady);
```

```
        return keyReady;
    }
}


using Microsoft.Extensions.Configuration;

namespace IAMGroups;

public class IAMGroups
{
    private static ILogger logger = null!;

    // Represents JSON code for AWS full access policy for Amazon Simple
    // Storage Service (Amazon S3).
    private const string S3FullAccessPolicyDocument = "{" +
        " \"Statement\" : [{" +
            " \"Action\" : [\"s3:*\"]," +
            " \"Effect\" : \"Allow\"," +
            " \"Resource\" : \"*\"" +
        "}]" +
    "}";


    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonIdentityManagementService>()
            .AddTransient<IAMWrapper>()
            .AddTransient<UIWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMGroups>();
```

```
        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var groupUserName = configuration["GroupUserName"];
        var groupName = configuration["GroupName"];
        var groupPolicyName = configuration["GroupPolicyName"];
        var groupBucketName = configuration["GroupBucketName"];

        var wrapper = host.Services.GetRequiredService<IAMWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        uiWrapper.DisplayGroupsOverview();
        uiWrapper.PressEnter();

        // Create an IAM group.
        uiWrapper.DisplayTitle("Create IAM group");
        Console.WriteLine("Let's begin by creating a new IAM group.");
        var group = await wrapper.CreateGroupAsync(groupName);

        // Add an inline IAM policy to the group.
        uiWrapper.DisplayTitle("Add policy to group");
        Console.WriteLine("Add an inline policy to the group that allows members
to have full access to");
        Console.WriteLine("Amazon Simple Storage Service (Amazon S3) buckets.");

        await wrapper.PutGroupPolicyAsync(group.GroupName, groupPolicyName,
S3FullAccessPolicyDocument);

        uiWrapper.PressEnter();

        // Now create a new user.
        uiWrapper.DisplayTitle("Create an IAM user");
        Console.WriteLine("Now let's create a new IAM user.");
        var groupUser = await wrapper.CreateUserAsync(groupUserName);

        // Add the new user to the group.
        uiWrapper.DisplayTitle("Add the user to the group");
        Console.WriteLine("Adding the user to the group, which will give the user
the same permissions as the group.");
        await wrapper.AddUserToGroupAsync(groupUser.UserName, group.GroupName);
```

```
        Console.WriteLine($"User, {groupUser.UserName}, has been added to the
group, {group.GroupName}.");
        uiWrapper.PressEnter();

        Console.WriteLine("Now that we have created a user, and added the user to
the group, let's create an IAM access key.");

        // Create access and secret keys for the user.
        var accessKey = await wrapper.CreateAccessKeyAsync(groupUserName);
        Console.WriteLine("Key created.");
        uiWrapper.WaitABit(15, "Waiting for the access key to be ready for
use.");

        uiWrapper.DisplayTitle("List buckets");
        Console.WriteLine("To prove that the user has access to Amazon S3, list
the S3 buckets for the account.");

        var s3Client = new AmazonS3Client(accessKey.AccessKeyId,
accessKey.SecretAccessKey);
        var stsClient = new
AmazonSecurityTokenServiceClient(accessKey.AccessKeyId,
accessKey.SecretAccessKey);

        var s3Wrapper = new S3Wrapper(s3Client, stsClient);

        var buckets = await s3Wrapper.ListMyBucketsAsync();

        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName}\tcreated on:
{bucket.CreationDate}");
            });
        }

        // Show that the user also has write access to Amazon S3 by creating
        // a new bucket.
        uiWrapper.DisplayTitle("Create a bucket");
        Console.WriteLine("Since group members have full access to Amazon S3,
let's create a bucket.");
        var success = await s3Wrapper.PutBucketAsync(groupBucketName);
```

```
        if (success)
        {
            Console.WriteLine($"Successfully created the bucket:
{groupBucketName}.");
        }

        uiWrapper.PressEnter();

        Console.WriteLine("Let's list the user's S3 buckets again to show the new
bucket.");

        buckets = await s3Wrapper.ListMyBucketsAsync();

        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName}\tcreated on:
{bucket.CreationDate}");
            });
        }

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("First delete the bucket we created.");
        await s3Wrapper.DeleteBucketAsync(groupBucketName);

        Console.WriteLine($"Now remove the user, {groupUserName}, from the group,
{groupName}.");
        await wrapper.RemoveUserFromGroupAsync(groupUserName, groupName);

        Console.WriteLine("Delete the user's access key.");
        await wrapper.DeleteAccessKeyAsync(accessKey.AccessKeyId, groupUserName);

        // Now we can safely delete the user.
        Console.WriteLine("Now we can delete the user.");
        await wrapper.DeleteUserAsync(groupUserName);

        uiWrapper.PressEnter();

        Console.WriteLine("Now we will delete the IAM policy attached to the
group.");
        await wrapper.DeleteGroupPolicyAsync(groupName, groupPolicyName);
```

```
        Console.WriteLine("Now we delete the IAM group.");
        await wrapper.DeleteGroupAsync(groupName);

        uiWrapper.PressEnter();

        Console.WriteLine("The IAM groups demo has completed.");

        uiWrapper.PressEnter();
    }
}


namespace IamScenariosCommon;

using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
    /// Constructor for the S3Wrapper class.
    /// </summary>
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</
param>
```

```
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };

        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
    }


    /// <summary>
    /// Delete an S3 bucket.
    /// </summary>
    /// <param name="bucketName">Name of the S3 bucket to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
        return result.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the buckets that are owned by the user's account.
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
```

```
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Create a new S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name for the new bucket.</param>
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
 { BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
 stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }
}


namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
```

```
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
(IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it
full access to Amazon S3.");
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries
to list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
```

```
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
```

```
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하십시오.
  - AddUserToGroup
  - AttachRolePolicy
  - CreateAccessKey
  - CreateGroup
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeleteGroup
  - DeleteGroupPolicy
  - DeleteUser
  - PutGroupPolicy
  - RemoveUserFromGroup

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 AWS STS에서 IAM 사용자 생성 및 역할 수임

다음 코드 예제에서는 사용자를 생성하고 역할을 수임하는 방법을 보여줍니다.

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 자격 증명 공급자를 통한 페더레이션을 사용하세요.

- 권한이 없는 사용자를 생성합니다.
- 계정에 대한 Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할을 생성합니다.
- 사용자가 역할을 수임할 수 있도록 정책을 추가합니다.
- 역할을 수임하고 임시 보안 인증 정보를 사용하여 S3 버킷을 나열한 후 리소스를 정리합니다.

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


namespace IAMActions;

public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;
```

```
    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
    /// Add an existing IAM user to an existing IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to add.</param>
    /// <param name="groupName">The name of the group to add the user to.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AddUserToGroupAsync(string userName, string
 groupName)
    {
        var response = await _IAMService.AddUserToGroupAsync(new
 AddUserToGroupRequest
        {
            GroupName = groupName,
            UserName = userName,
        });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
```

```
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }


    /// <summary>
    /// Create an IAM group.
    /// </summary>
    /// <param name="groupName">The name to give the IAM group.</param>
    /// <returns>The IAM group that was created.</returns>
    public async Task<Group> CreateGroupAsync(string groupName)
    {
        var response = await _IAMService.CreateGroupAsync(new CreateGroupRequest
{ GroupName = groupName });
        return response.Group;
    }


    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
```

```
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }


    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
 rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }


    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
```

```
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
string description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }


    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupAsync(string groupName)
    {
        var response = await _IAMService.DeleteGroupAsync(new DeleteGroupRequest
{ GroupName = groupName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM policy associated with an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group associated with the
    /// policy.</param>
    /// <param name="policyName">The name of the policy to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupPolicyAsync(string groupName, string
policyName)
    {
        var request = new DeleteGroupPolicyRequest()
        {
            GroupName = groupName,
            PolicyName = policyName,
        };

        var response = await _IAMService.DeleteGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
```

```
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
 DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
 { RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
```

```
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
```

```
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
 GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }


    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }


    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
```

```
        });

        return response.Role;
    }


    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }


    /// <summary>
    /// List IAM groups.
    /// </summary>
```

```
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
 ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }


    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }


    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
```

```
        var listRolePoliciesPaginator =
_IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }


    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }


    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }
```

```
    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Remove a user from an IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to remove.</param>
    /// <param name="groupName">The name of the IAM group to remove the user
from.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> RemoveUserFromGroupAsync(string userName, string
groupName)
    {
        // Remove the user from the group.
        var removeUserRequest = new RemoveUserFromGroupRequest()
        {
            UserName = userName,
            GroupName = groupName,
        };

        var response = await
_IAMService.RemoveUserFromGroupAsync(removeUserRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
```

```
    /// Add or update an inline policy document that is embedded in an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
 policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutGroupPolicyAsync(string groupName, string
 policyName, string policyDocument)
    {
        var request = new PutGroupPolicyRequest
        {
            GroupName = groupName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM user.
    /// </summary>
    /// <param name="userName">The name of the IAM user.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutUserPolicyAsync(string userName, string
policyName, string policyDocument)
    {
        var request = new PutUserPolicyRequest
        {
            UserName = userName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutUserPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Wait for a new access key to be ready to use.
    /// </summary>
    /// <param name="accessKeyId">The Id of the access key.</param>
    /// <returns>A boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
    {
        var keyReady = false;

        do
        {
            try
            {
                var response = await _IAMService.GetAccessKeyLastUsedAsync(
                    new GetAccessKeyLastUsedRequest { AccessKeyId =
accessKeyId });
                if (response.UserName is not null)
                {
                    keyReady = true;
                }
            }
```

```
            catch (NoSuchEntityException)
            {
                keyReady = false;
            }
        } while (!keyReady);

        return keyReady;
    }
}
```

```
using Microsoft.Extensions.Configuration;

namespace IAMBasics;

public class IAMBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonIdentityManagementService>()
            .AddTransient<IAMWrapper>()
            .AddTransient<UIWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMBasics>();


        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
```

```
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // Values needed for user, role, and policies.
        string userName = configuration["UserName"]!;
        string s3PolicyName = configuration["S3PolicyName"]!;
        string roleName = configuration["RoleName"]!;


        var iamWrapper = host.Services.GetRequiredService<IAMWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        uiWrapper.DisplayBasicsOverview();
        uiWrapper.PressEnter();

        // First create a user. By default, the new user has
        // no permissions.
        uiWrapper.DisplayTitle("Create User");
        Console.WriteLine($"Creating a new user with user name: {userName}.");
        var user = await iamWrapper.CreateUserAsync(userName);
        var userArn = user.Arn;

        Console.WriteLine($"Successfully created user: {userName} with ARN:
{userArn}.");
        uiWrapper.WaitABit(15, "Now let's wait for the user to be ready for
use.");

        // Define a role policy document that allows the new user
        // to assume the role.
        string assumeRolePolicyDocument = "{" +
          "\"Version\": \"2012-10-17\"," +
          "\"Statement\": [{" +
              "\"Effect\": \"Allow\"," +
              "\"Principal\": {" +
              $" \"AWS\": \"{userArn}\"" +
              "}," +
              "\"Action\": \"sts:AssumeRole\"" +
          "}]" +
        "}";

        // Permissions to list all buckets.
        string policyDocument = "{" +
            "\"Version\": \"2012-10-17\"," +
```

```
            " \"Statement\" : [{" +
                " \"Action\" : [\"s3:ListAllMyBuckets\"]," +
                " \"Effect\" : \"Allow\"," +
                " \"Resource\" : \"*\"" +
            "}]" +
        "}";

        // Create an AccessKey for the user.
        uiWrapper.DisplayTitle("Create access key");
        Console.WriteLine("Now let's create an access key for the new user.");
        var accessKey = await iamWrapper.CreateAccessKeyAsync(userName);

        var accessKeyId = accessKey.AccessKeyId;
        var secretAccessKey = accessKey.SecretAccessKey;

        Console.WriteLine($"We have created the access key with Access key id:
{accessKeyId}.");

        Console.WriteLine("Now let's wait until the IAM access key is ready to
use.");
        var keyReady = await iamWrapper.WaitUntilAccessKeyIsReady(accessKeyId);

        // Now try listing the Amazon Simple Storage Service (Amazon S3)
        // buckets. This should fail at this point because the user doesn't
        // have permissions to perform this task.
        uiWrapper.DisplayTitle("Try to display Amazon S3 buckets");
        Console.WriteLine("Now let's try to display a list of the user's Amazon
S3 buckets.");
        var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
        var stsClient1 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        var s3Wrapper = new S3Wrapper(s3Client1, stsClient1);
        var buckets = await s3Wrapper.ListMyBucketsAsync();

        Console.WriteLine(buckets is null
            ? "As expected, the call to list the buckets has returned a null
list."
            : "Something went wrong. This shouldn't have worked.");

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create IAM role");
        Console.WriteLine($"Creating the role: {roleName}");
```

```
        // Creating an IAM role to allow listing the S3 buckets. A role name
        // is not case sensitive and must be unique to the account for which it
        // is created.
        var roleArn = await iamWrapper.CreateRoleAsync(roleName,
assumeRolePolicyDocument);

        uiWrapper.PressEnter();

        // Create a policy with permissions to list S3 buckets.
        uiWrapper.DisplayTitle("Create IAM policy");
        Console.WriteLine($"Creating the policy: {s3PolicyName}");
        Console.WriteLine("with permissions to list the Amazon S3 buckets for the
account.");
        var policy = await iamWrapper.CreatePolicyAsync(s3PolicyName,
policyDocument);

        // Wait 15 seconds for the IAM policy to be available.
        uiWrapper.WaitABit(15, "Waiting for the policy to be available.");

        // Attach the policy to the role you created earlier.
        uiWrapper.DisplayTitle("Attach new IAM policy");
        Console.WriteLine("Now let's attach the policy to the role.");
        await iamWrapper.AttachRolePolicyAsync(policy.Arn, roleName);

        // Wait 15 seconds for the role to be updated.
        Console.WriteLine();
        uiWrapper.WaitABit(15, "Waiting for the policy to be attached.");

        // Use the AWS Security Token Service (AWS STS) to have the user
        // assume the role we created.
        var stsClient2 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        // Wait for the new credentials to become valid.
        uiWrapper.WaitABit(10, "Waiting for the credentials to be valid.");

        var assumedRoleCredentials = await
s3Wrapper.AssumeS3RoleAsync("temporary-session", roleArn);

        // Try again to list the buckets using the client created with
        // the new user's credentials. This time, it should work.
        var s3Client2 = new AmazonS3Client(assumedRoleCredentials);
```

```
        s3Wrapper.UpdateClients(s3Client2, stsClient2);

        buckets = await s3Wrapper.ListMyBucketsAsync();

        uiWrapper.DisplayTitle("List Amazon S3 buckets");
        Console.WriteLine("This time we should have buckets to list.");
        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName} created:
 {bucket.CreationDate}");
            });
        }

        uiWrapper.PressEnter();

        // Now clean up all the resources used in the example.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Thank you for watching. The IAM Basics demo is
 complete.");
        Console.WriteLine("Please wait while we clean up the resources we
 created.");

        await iamWrapper.DetachRolePolicyAsync(policy.Arn, roleName);

        await iamWrapper.DeletePolicyAsync(policy.Arn);

        await iamWrapper.DeleteRoleAsync(roleName);

        await iamWrapper.DeleteAccessKeyAsync(accessKeyId, userName);

        await iamWrapper.DeleteUserAsync(userName);

        uiWrapper.PressEnter();

        Console.WriteLine("All done cleaning up our resources. Thank you for your
 patience.");
    }
}


namespace IamScenariosCommon;
```

```
using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
    /// Constructor for the S3Wrapper class.
    /// </summary>
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</
param>
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
 roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };

        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
```

```
    }


    /// <summary>
    /// Delete an S3 bucket.
    /// </summary>
    /// <param name="bucketName">Name of the S3 bucket to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
        return result.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the buckets that are owned by the user's account.
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Create a new S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name for the new bucket.</param>
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
```

```
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
 { BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
 stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }
}


namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
 (IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it
 full access to Amazon S3.");
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
```

```
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries
to list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
```

```
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey

- CreatePolicy
- CreateRole
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#############################################################################
# function iam_create_user_assume_role
#
# Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
#
#     "IAM access" permissions are needed to run this code.
#     "STS assume role" permissions are needed to run this code. (Note: It might
 be necessary to
#           create a custom policy).
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
#############################################################################
function iam_create_user_assume_role() {
```

```
 {
   if [ "$IAM_OPERATIONS_SOURCED" != "True" ]; then

     source ./iam_operations.sh
   fi
 }

 echo_repeat "*" 88
 echo "Welcome to the IAM create user and assume role demo."
 echo
 echo "This demo will create an IAM user, create an IAM role, and apply the role
to the user."
 echo_repeat "*" 88
 echo

 echo -n "Enter a name for a new IAM user: "
 get_input
 user_name=$get_input_result

 local user_arn
 user_arn=$(iam_create_user -u "$user_name")

 # shellcheck disable=SC2181
 if [[ ${?} == 0 ]]; then
   echo "Created demo IAM user named $user_name"
 else
   errecho "$user_arn"
   errecho "The user failed to create. This demo will exit."
   return 1
 fi

 local access_key_response
 access_key_response=$(iam_create_user_access_key -u "$user_name")
 # shellcheck disable=SC2181
 if [[ ${?} != 0 ]]; then
   errecho "The access key failed to create. This demo will exit."
   clean_up "$user_name"
   return 1
 fi

 IFS=$'\t ' read -r -a access_key_values <<<"$access_key_response"
 local key_name=${access_key_values[0]}
 local key_secret=${access_key_values[1]}
```

```
  echo "Created access key named $key_name"

  echo "Wait 10 seconds for the user to be ready."
  sleep 10
  echo_repeat "*" 88
  echo

  local iam_role_name
  iam_role_name=$(generate_random_name "test-role")
  echo "Creating a role named $iam_role_name with user $user_name as the
principal."

  local assume_role_policy_document="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Principal\": {\"AWS\": \"$user_arn\"},
        \"Action\": \"sts:AssumeRole\"
        }]
    }"

  local role_arn
  role_arn=$(iam_create_role -n "$iam_role_name" -p
"$assume_role_policy_document")

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Created IAM role named $iam_role_name"
  else
    errecho "The role failed to create. This demo will exit."
    clean_up "$user_name" "$key_name"
    return 1
  fi

  local policy_name
  policy_name=$(generate_random_name "test-policy")
  local policy_document="{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]}"

  local policy_arn
```

```
 policy_arn=$(iam_create_policy -n "$policy_name" -p "$policy_document")
 # shellcheck disable=SC2181
 if [[ ${?} == 0 ]]; then
   echo "Created  IAM policy named $policy_name"
 else
   errecho "The policy failed to create."
   clean_up "$user_name" "$key_name" "$iam_role_name"
   return 1
 fi

 if (iam_attach_role_policy -n "$iam_role_name" -p "$policy_arn"); then
   echo "Attached policy $policy_arn to role $iam_role_name"
 else
   errecho "The policy failed to attach."
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
   return 1
 fi

 local assume_role_policy_document="{
              \"Version\": \"2012-10-17\",
              \"Statement\": [{
                  \"Effect\": \"Allow\",
                  \"Action\": \"sts:AssumeRole\",
                  \"Resource\": \"$role_arn\"}]}"

 local assume_role_policy_name
 assume_role_policy_name=$(generate_random_name "test-assume-role-")

 # shellcheck disable=SC2181
 local assume_role_policy_arn
 assume_role_policy_arn=$(iam_create_policy -n "$assume_role_policy_name" -p
"$assume_role_policy_document")
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   echo "Created  IAM policy named $assume_role_policy_name for sts assume role"
 else
   errecho "The policy failed to create."
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn"
   return 1
 fi

 echo "Wait 10 seconds to give AWS time to propagate these new resources and
connections."
```

```
  sleep 10
  echo_repeat "*" 88
  echo

  echo "Try to list buckets without the new user assuming the role."
  echo_repeat "*" 88
  echo

  # Set the environment variables for the created user.
  # bashsupport disable=BP2001
  export AWS_ACCESS_KEY_ID=$key_name
  # bashsupport disable=BP2001
  export AWS_SECRET_ACCESS_KEY=$key_secret

  local buckets
  buckets=$(s3_list_buckets)

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. This should not have
happened."
  else
    errecho "Because the role with permissions has not been assumed, listing
buckets failed."
  fi

  echo
  echo_repeat "*" 88
  echo "Now assume the role $iam_role_name and list the buckets."
  echo_repeat "*" 88
  echo

  local credentials

  credentials=$(sts_assume_role -r "$role_arn" -n "AssumeRoleDemoSession")
  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Assumed role $iam_role_name"
  else
    errecho "Failed to assume role."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
```

```
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
    return 1
  fi

  IFS=$'\t ' read -r -a credentials <<<"$credentials"

  export AWS_ACCESS_KEY_ID=${credentials[0]}
  export AWS_SECRET_ACCESS_KEY=${credentials[1]}
  # bashsupport disable=BP2001
  export AWS_SESSION_TOKEN=${credentials[2]}

  buckets=$(s3_list_buckets)

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. Listing buckets
succeeded because of "
    echo "the assumed role."
  else
    errecho "Failed to list buckets. This should not happen."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    export AWS_SESSION_TOKEN=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
    return 1
  fi

  local result=0
  export AWS_ACCESS_KEY_ID=""
  export AWS_SECRET_ACCESS_KEY=""

  echo
  echo_repeat "*" 88
  echo "The created resources will now be deleted."
  echo_repeat "*" 88
  echo

  clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
"$assume_role_policy_arn"
```

```
  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    result=1
  fi

  return $result
}
```

이 시나리오에 사용된 IAM 함수입니다.

```
########################################################################################
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
########################################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
```

```
    fi

    return 1 # 1 in Bash script means false.
  fi
}

###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an WS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user already exists, we don't want to try to create it.
  if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
  fi

  response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_user_access_key
#
```

```
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
    echo "  -u user_name   The name of the IAM user."
    echo "  [-f file_name]   Optional file name for the access key output."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:f:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      f) file_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
```

```
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam create-access-key \
    --user-name "$user_name" \
    --output text)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  fi

  if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
  fi

  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}

###############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
```

```
#       0 - If successful.
#       1 - If it fails.
################################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_json -- The assume role policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
```

```
  fi

  response=$(aws iam create-role \
    --role-name "$role_name" \
    --assume-role-policy-document "$policy_document" \
    --output text \
    --query Role.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
```

```
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) policy_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$policy_name" ]]; then
  errecho "ERROR: You must provide a policy name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-policy \
  --policy-name "$policy_name" \
  --policy-document "$policy_document" \
  --output text \
  --query Policy.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-policy operation failed.\n$response"
```

```
    return 1
  fi

  echo "$response"
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
```

```
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
```

```
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
```

```
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an WS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
```

```
}

# Retrieve the calling parameters.
while getopts "n:h" option; do
  case "${option}" in
    n) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy arn with the -n parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Policy arn:  $policy_arn"
iecho ""

response=$(aws iam delete-policy \
  --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
  return 1
fi

iecho "delete-policy response:$response"
iecho

return 0
```

```
}

########################################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
########################################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an WS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
```

```
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}

################################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.
```

```
 # bashsupport disable=BP5008
 function usage() {
   echo "function iam_delete_access_key"
   echo "Deletes an WS Identity and Access Management (IAM) access key for the
specified IAM user"
   echo "  -u user_name    The name of the user."
   echo "  -k access_key   The access key to delete."
   echo ""
 }

 # Retrieve the calling parameters.
 while getopts "u:k:h" option; do
   case "${option}" in
     u) user_name="${OPTARG}" ;;
     k) access_key="${OPTARG}" ;;
     h)
       usage
       return 0
       ;;
     \?)
       echo "Invalid parameter"
       usage
       return 1
       ;;
   esac
 done
 export OPTIND=1

 if [[ -z "$user_name" ]]; then
   errecho "ERROR: You must provide a username with the -u parameter."
   usage
   return 1
 fi

 if [[ -z "$access_key" ]]; then
   errecho "ERROR: You must provide an access key with the -k parameter."
   usage
   return 1
 fi

 iecho "Parameters:\n"
 iecho "    Username:   $user_name"
 iecho "    Access key:   $access_key"
 iecho ""
```

```
  response=$(aws iam delete-access-key \
    --user-name "$user_name" \
    --access-key-id "$access_key")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
  fi

  iecho "delete-access-key response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an WS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }
```

```
  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
  fi

  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi
```

```
  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- API 세부 정보는 AWS CLI 명령 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace AwsDoc {
    namespace IAM {

        //! Cleanup by deleting created entities.
        /*!
```

```
          \sa DeleteCreatedEntities
          \param client: IAM client.
          \param role: IAM role.
          \param user: IAM user.
          \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                          const Aws::IAM::Model::Role &role,
                                          const Aws::IAM::Model::User &user,
                                          const Aws::IAM::Model::Policy &policy);
    }

    static const int LIST_BUCKETS_WAIT_SEC = 20;

    static const char ALLOCATION_TAG[] = "example_code";
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
 necessary to
//    create a custom policy).
/*!
  \sa iamCreateUserAssumeRoleScenario
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);
```

```
        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            std::cout << "Successfully created IAM user " << userName <<
std::endl;
        }

        user = outcome.GetResult().GetUser();
    }

    // 2. Create a role.
    {
        // Get the IAM user for the current client in order to access its ARN.
        Aws::String iamUserArn;
        {
            Aws::IAM::Model::GetUserRequest request;
            Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error getting Iam user. " <<
                          outcome.GetError().GetMessage() << std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            else {
                std::cout << "Successfully retrieved Iam user "
                          << outcome.GetResult().GetUser().GetUserName()
                          << std::endl;
            }

            iamUserArn = outcome.GetResult().GetUser().GetArn();
        }

        Aws::IAM::Model::CreateRoleRequest request;

        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleName = "iam-demo-role-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleName(roleName);
```

```
        // Build policy document for role.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");

        Aws::Utils::Document jsonPrincipal;
        jsonPrincipal.WithString("AWS", iamUserArn);
        jsonStatement.WithObject("Principal", jsonPrincipal);
        jsonStatement.WithString("Action", "sts:AssumeRole");
        jsonStatement.WithObject("Condition", Aws::Utils::Document());

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Setting policy for role\n   "
                  << policyDocument.View().WriteCompact() << std::endl;

        // Set role policy document as JSON string.

request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating role. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a role with name " << roleName
                      << std::endl;
        }

        role = outcome.GetResult().GetRole();
    }

    // 3. Create an IAM policy.
    {
        Aws::IAM::Model::CreatePolicyRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
```

```
        Aws::String policyName = "iam-demo-policy-" +
                                 Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetPolicyName(policyName);

        // Build IAM policy document.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");
        jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
        jsonStatement.WithString("Resource", "arn:aws:s3:::*");

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Creating a policy.\n   " <<
policyDocument.View().WriteCompact()
                  << std::endl;

        // Set IAM policy document as JSON string.
        request.SetPolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreatePolicyOutcome outcome =
client.CreatePolicy(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a policy with name, " <<
policyName <<
                      "." << std::endl;
        }

        policy = outcome.GetResult().GetPolicy();
    }

    // 4. Assume the new role using the AWS Security Token Service (STS).
    Aws::STS::Model::Credentials credentials;
```

```
    {
        Aws::STS::STSClient stsClient(clientConfig);

        Aws::STS::Model::AssumeRoleRequest request;
        request.SetRoleArn(role.GetArn());
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleSessionName = "iam-demo-role-session-" +

Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleSessionName(roleSessionName);

        Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

        // Repeatedly call AssumeRole, because there is often a delay
        // before the role is available to be assumed.
        // Repeat at most 20 times when access is denied.
        int count = 0;
        while (true) {
            assumeRoleOutcome = stsClient.AssumeRole(request);
            if (!assumeRoleOutcome.IsSuccess()) {
                if (count > 20 ||
                    assumeRoleOutcome.GetError().GetErrorType() !=
                    Aws::STS::STSErrors::ACCESS_DENIED) {
                    std::cerr << "Error assuming role after 20 tries. " <<
                              assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

                    DeleteCreatedEntities(client, role, user, policy);
                    return false;
                }
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                std::cout << "Successfully assumed the role after " << count
                          << " seconds." << std::endl;
                break;
            }
            count++;
        }

        credentials = assumeRoleOutcome.GetResult().GetCredentials();
    }
```

```
   // 5. List objects in the bucket (This should fail).
   {
       Aws::S3::S3Client s3Client(
               Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                         credentials.GetSecretAccessKey(),
                                         credentials.GetSessionToken()),
               Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
               clientConfig);
       Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
       if (!listBucketsOutcome.IsSuccess()) {
           if (listBucketsOutcome.GetError().GetErrorType() !=
               Aws::S3::S3Errors::ACCESS_DENIED) {
               std::cerr << "Could not lists buckets. " <<
                         listBucketsOutcome.GetError().GetMessage() <<
std::endl;
           }
           else {
               std::cout
                       << "Access to list buckets denied because privileges have
not been applied."
                       << std::endl;
           }
       }
       else {
           std::cerr
                   << "Successfully retrieved bucket lists when this should not
happen."
                   << std::endl;
       }
   }

   // 6. Attach the policy to the role.
   {
       Aws::IAM::Model::AttachRolePolicyRequest request;
       request.SetRoleName(role.GetRoleName());
       request.WithPolicyArn(policy.GetArn());

       Aws::IAM::Model::AttachRolePolicyOutcome outcome =
client.AttachRolePolicy(
               request);
       if (!outcome.IsSuccess()) {
           std::cerr << "Error creating policy. " <<
                     outcome.GetError().GetMessage() << std::endl;
```

```
            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully attached the policy with name, "
                      << policy.GetPolicyName() <<
                      ", to the role, " << role.GetRoleName() << "." <<
std::endl;
        }
    }

    int count = 0;
    // 7. List objects in the bucket (this should succeed).
    // Repeatedly call ListBuckets, because there is often a delay
    // before the policy with ListBucket permissions has been applied to the
role.
    // Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
    while (true) {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if ((count > LIST_BUCKETS_WAIT_SEC) ||
                listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets after " <<
LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                          listBucketsOutcome.GetError().GetMessage() <<
std::endl;
                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }

            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {

            std::cout << "Successfully retrieved bucket lists after " << count
```

```
                        << " seconds." << std::endl;
            break;
        }
        count++;
    }

    // 8. Delete all the created resources.
    return DeleteCreatedEntities(client, role, user, policy);
}

bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                        const Aws::IAM::Model::Role &role,
                                        const Aws::IAM::Model::User &user,
                                        const Aws::IAM::Model::Policy &policy) {
    bool result = true;
    if (policy.ArnHasBeenSet()) {
        // Detach the policy from the role.
        {
            Aws::IAM::Model::DetachRolePolicyRequest request;
            request.SetPolicyArn(policy.GetArn());
            request.SetRoleName(role.GetRoleName());

            Aws::IAM::Model::DetachRolePolicyOutcome outcome =
 client.DetachRolePolicy(
                    request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error Detaching policy from roles. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully detached the policy with arn "
                          << policy.GetArn()
                          << " from role " << role.GetRoleName() << "." <<
 std::endl;
            }
        }

        // Delete the policy.
        {
            Aws::IAM::Model::DeletePolicyRequest request;
            request.WithPolicyArn(policy.GetArn());
```

```
            Aws::IAM::Model::DeletePolicyOutcome outcome =
client.DeletePolicy(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error deleting policy. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully deleted the policy with arn "
                          << policy.GetArn() << std::endl;
            }
        }

    }

    if (role.RoleIdHasBeenSet()) {
        // Delete the role.
        Aws::IAM::Model::DeleteRoleRequest request;
        request.SetRoleName(role.GetRoleName());

        Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting role. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the role with name "
                      << role.GetRoleName() << std::endl;
        }
    }

    if (user.ArnHasBeenSet()) {
        // Delete the user.
        Aws::IAM::Model::DeleteUserRequest request;
        request.WithUserName(user.GetUserName());

        Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting user. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
```

```
            std::cout << "Successfully deleted the user with name "
                      << user.GetUserName() << std::endl;
        }
    }

    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Go

SDK for Go V2

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
// AssumeRoleScenario shows you how to use the AWS Identity and Access Management
 (IAM)
// service to perform the following actions:
//
// 1. Create a user who has no permissions.
// 2. Create a role that grants permission to list Amazon Simple Storage Service
//    (Amazon S3) buckets for the account.
// 3. Add a policy to let the user assume the role.
// 4. Try and fail to list buckets without permissions.
// 5. Assume the role and list S3 buckets using temporary credentials.
// 6. Delete the policy, role, and user.
type AssumeRoleScenario struct {
 sdkConfig aws.Config
 accountWrapper actions.AccountWrapper
 policyWrapper actions.PolicyWrapper
 roleWrapper actions.RoleWrapper
 userWrapper actions.UserWrapper
 questioner demotools.IQuestioner
 helper IScenarioHelper
 isTestRun bool
}

// NewAssumeRoleScenario constructs an AssumeRoleScenario instance from a
 configuration.
// It uses the specified config to get an IAM client and create wrappers for the
 actions
// used in the scenario.
func NewAssumeRoleScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
  helper IScenarioHelper) AssumeRoleScenario {
 iamClient := iam.NewFromConfig(sdkConfig)
 return AssumeRoleScenario{
  sdkConfig:   sdkConfig,
  accountWrapper: actions.AccountWrapper{IamClient: iamClient},
  policyWrapper:  actions.PolicyWrapper{IamClient: iamClient},
  roleWrapper:    actions.RoleWrapper{IamClient: iamClient},
  userWrapper:    actions.UserWrapper{IamClient: iamClient},
  questioner:     questioner,
  helper:         helper,
 }
}

// addTestOptions appends the API options specified in the original configuration
 to
```

```
// another configuration. This is used to attach the middleware stubber to
 clients
// that are constructed during the scenario, which is needed for unit testing.
func (scenario AssumeRoleScenario) addTestOptions(scenarioConfig *aws.Config) {
 if scenario.isTestRun {
  scenarioConfig.APIOptions = append(scenarioConfig.APIOptions,
 scenario.sdkConfig.APIOptions...)
 }
}

// Run runs the interactive scenario.
func (scenario AssumeRoleScenario) Run() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong with the demo.\n")
   log.Println(r)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the AWS Identity and Access Management (IAM) assume role
 demo.")
 log.Println(strings.Repeat("-", 88))

 user := scenario.CreateUser()
 accessKey := scenario.CreateAccessKey(user)
 role := scenario.CreateRoleAndPolicies(user)
 noPermsConfig := scenario.ListBucketsWithoutPermissions(accessKey)
 scenario.ListBucketsWithAssumedRole(noPermsConfig, role)
 scenario.Cleanup(user, role)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateUser creates a new IAM user. This user has no permissions.
func (scenario AssumeRoleScenario) CreateUser() *types.User {
 log.Println("Let's create an example user with no permissions.")
 userName := scenario.questioner.Ask("Enter a name for the example user:",
 demotools.NotEmpty{})
 user, err := scenario.userWrapper.GetUser(userName)
 if err != nil {
  panic(err)
```

```
 }
 if user == nil {
  user, err = scenario.userWrapper.CreateUser(userName)
  if err != nil {
   panic(err)
  }
  log.Printf("Created user %v.\n", *user.UserName)
 } else {
  log.Printf("User %v already exists.\n", *user.UserName)
 }
 log.Println(strings.Repeat("-", 88))
 return user
}

// CreateAccessKey creates an access key for the user.
func (scenario AssumeRoleScenario) CreateAccessKey(user *types.User)
 *types.AccessKey {
 accessKey, err := scenario.userWrapper.CreateAccessKeyPair(*user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("Created access key %v for your user.", *accessKey.AccessKeyId)
 log.Println("Waiting a few seconds for your user to be ready...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return accessKey
}

// CreateRoleAndPolicies creates a policy that grants permission to list S3
 buckets for
// the current account and attaches the policy to a newly created role. It also
 adds an
// inline policy to the specified user that grants the user permission to assume
 the role.
func (scenario AssumeRoleScenario) CreateRoleAndPolicies(user *types.User)
 *types.Role {
 log.Println("Let's create a role and policy that grant permission to list S3
 buckets.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 listBucketsRole, err :=
 scenario.roleWrapper.CreateRole(scenario.helper.GetName(), *user.Arn)
 if err != nil {panic(err)}
 log.Printf("Created role %v.\n", *listBucketsRole.RoleName)
 listBucketsPolicy, err := scenario.policyWrapper.CreatePolicy(
```

```
  scenario.helper.GetName(), []string{"s3:ListAllMyBuckets"}, "arn:aws:s3:::*")
 if err != nil {panic(err)}
 log.Printf("Created policy %v.\n", *listBucketsPolicy.PolicyName)
 err = scenario.roleWrapper.AttachRolePolicy(*listBucketsPolicy.Arn,
 *listBucketsRole.RoleName)
 if err != nil {panic(err)}
 log.Printf("Attached policy %v to role %v.\n", *listBucketsPolicy.PolicyName,
  *listBucketsRole.RoleName)
 err = scenario.userWrapper.CreateUserPolicy(*user.UserName,
 scenario.helper.GetName(),
  []string{"sts:AssumeRole"}, *listBucketsRole.Arn)
 if err != nil {panic(err)}
 log.Printf("Created an inline policy for user %v that lets the user assume the
 role.\n",
  *user.UserName)
 log.Println("Let's give AWS a few seconds to propagate these new resources and
 connections...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return listBucketsRole
}

// ListBucketsWithoutPermissions creates an Amazon S3 client from the user's
 access key
// credentials and tries to list buckets for the account. Because the user does
 not have
// permission to perform this action, the action fails.
func (scenario AssumeRoleScenario) ListBucketsWithoutPermissions(accessKey
 *types.AccessKey) *aws.Config {
  log.Println("Let's try to list buckets without permissions. This should return
 an AccessDenied error.")
  scenario.questioner.Ask("Press Enter when you're ready.")
  noPermsConfig, err := config.LoadDefaultConfig(context.TODO(),
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *accessKey.AccessKeyId, *accessKey.SecretAccessKey, ""),
 ))
  if err != nil {panic(err)}

  // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&noPermsConfig)

  s3Client := s3.NewFromConfig(noPermsConfig)
  _, err = s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
```

```
  if err != nil {
   // The SDK for Go does not model the AccessDenied error, so check ErrorCode
 directly.
  var ae smithy.APIError
  if errors.As(err, &ae) {
   switch ae.ErrorCode() {
   case "AccessDenied":
    log.Println("Got AccessDenied error, which is the expected result because\n"
 +
     "the ListBuckets call was made without permissions.")
   default:
    log.Println("Expected AccessDenied, got something else.")
    panic(err)
   }
  }
  } else {
   log.Println("Expected AccessDenied error when calling ListBuckets without
 permissions,\n" +
    "but the call succeeded. Continuing the example anyway...")
 }
 log.Println(strings.Repeat("-", 88))
  return &noPermsConfig
}

// ListBucketsWithAssumedRole performs the following actions:
//
// 1. Creates an AWS Security Token Service (AWS STS) client from the config
 created from
//   the user's access key credentials.
// 2. Gets temporary credentials by assuming the role that grants permission to
 list the
//    buckets.
// 3. Creates an Amazon S3 client from the temporary credentials.
// 4. Lists buckets for the account. Because the temporary credentials are
 generated by
//    assuming the role that grants permission, the action succeeds.
func (scenario AssumeRoleScenario) ListBucketsWithAssumedRole(noPermsConfig
 *aws.Config, role *types.Role) {
 log.Println("Let's assume the role that grants permission to list buckets and
 try again.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 stsClient := sts.NewFromConfig(*noPermsConfig)
 tempCredentials, err := stsClient.AssumeRole(context.TODO(),
 &sts.AssumeRoleInput{
```

```
  RoleArn:           role.Arn,
  RoleSessionName:   aws.String("AssumeRoleExampleSession"),
  DurationSeconds:   aws.Int32(900),
 })
 if err != nil {
  log.Printf("Couldn't assume role %v.\n", *role.RoleName)
  panic(err)
 }
 log.Printf("Assumed role %v, got temporary credentials.\n", *role.RoleName)
 assumeRoleConfig, err := config.LoadDefaultConfig(context.TODO(),
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *tempCredentials.Credentials.AccessKeyId,
   *tempCredentials.Credentials.SecretAccessKey,
   *tempCredentials.Credentials.SessionToken),
  ),
 )
 if err != nil {panic(err)}

 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&assumeRoleConfig)

 s3Client := s3.NewFromConfig(assumeRoleConfig)
 result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
 if err != nil {
  log.Println("Couldn't list buckets with assumed role credentials.")
  panic(err)
 }
 log.Println("Successfully called ListBuckets with assumed role credentials, \n"
 +
  "here are some of them:")
 for i := 0; i < len(result.Buckets) && i < 5; i++ {
  log.Printf("\t%v\n", *result.Buckets[i].Name)
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup deletes all resources created for the scenario.
func (scenario AssumeRoleScenario) Cleanup(user *types.User, role *types.Role) {
 if scenario.questioner.AskBool(
  "Do you want to delete the resources created for this example? (y/n)", "y",
 ) {
   policies, err := scenario.roleWrapper.ListAttachedRolePolicies(*role.RoleName)
   if err != nil {panic(err)}
```

```
    for _, policy := range policies {
     err = scenario.roleWrapper.DetachRolePolicy(*role.RoleName,
 *policy.PolicyArn)
     if err != nil {panic(err)}
     err = scenario.policyWrapper.DeletePolicy(*policy.PolicyArn)
     if err != nil {panic(err)}
     log.Printf("Detached policy %v from role %v and deleted the policy.\n",
      *policy.PolicyName, *role.RoleName)
    }
    err = scenario.roleWrapper.DeleteRole(*role.RoleName)
    if err != nil {panic(err)}
    log.Printf("Deleted role %v.\n", *role.RoleName)

    userPols, err := scenario.userWrapper.ListUserPolicies(*user.UserName)
    if err != nil {panic(err)}
    for _, userPol := range userPols {
     err = scenario.userWrapper.DeleteUserPolicy(*user.UserName, userPol)
     if err != nil {panic(err)}
     log.Printf("Deleted policy %v from user %v.\n", userPol, *user.UserName)
    }
    keys, err := scenario.userWrapper.ListAccessKeys(*user.UserName)
    if err != nil {panic(err)}
    for _, key := range keys {
     err = scenario.userWrapper.DeleteAccessKey(*user.UserName, *key.AccessKeyId)
     if err != nil {panic(err)}
     log.Printf("Deleted access key %v from user %v.\n", *key.AccessKeyId,
 *user.UserName)
    }
    err = scenario.userWrapper.DeleteUser(*user.UserName)
    if err != nil {panic(err)}
    log.Printf("Deleted user %v.\n", *user.UserName)
    log.Println(strings.Repeat("-", 88))
 }

}
```

계정 작업을 래핑하는 구조체를 정의합니다.

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
```

```
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy() (*types.PasswordPolicy,
 error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(context.TODO(),
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}



// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders() ([]types.SAMLProviderListEntry,
 error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(context.TODO(),
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

정책 작업을 래핑하는 구조체를 정의합니다.

```
// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect string
 Action []string
 Principal map[string]string `json:",omitempty"`
 Resource *string `json:",omitempty"`
}



// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(maxPolicies int32) ([]types.Policy,
 error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(context.TODO(),
 &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}
```

```
// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(policyName string, actions []string,
  resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Action: actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreatePolicy(context.TODO(),
 &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(policyArn string) (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(context.TODO(), &iam.GetPolicyInput{
```

```
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(policyArn string) error {
 _, err := wrapper.IamClient.DeletePolicy(context.TODO(), &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

역할 작업을 래핑하는 구조체를 정의합니다.

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(maxRoles int32) ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(context.TODO(),
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
```

```
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(roleName string, trustedUserArn string)
 (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action: []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreateRole(context.TODO(),
 &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
```

```
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(roleName string) (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(context.TODO(),
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(serviceName string,
 description string) (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(context.TODO(),
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(roleName string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(context.TODO(),
 &iam.DeleteServiceLinkedRoleInput{
```

```
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
 roleName, err)
 }
 return err
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(policyArn string, roleName string)
 error {
 _, err := wrapper.IamClient.AttachRolePolicy(context.TODO(),
 &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(roleName string)
 ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(context.TODO(),
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
```

```
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(roleName string, policyArn string)
 error {
 _, err := wrapper.IamClient.DetachRolePolicy(context.TODO(),
 &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(roleName string) ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(context.TODO(),
 &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(roleName string) error {
 _, err := wrapper.IamClient.DeleteRole(context.TODO(), &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
```

```
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

사용자 작업을 래핑하는 구조체를 정의합니다.

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(maxUsers int32) ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(context.TODO(), &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(userName string) (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(context.TODO(), &iam.GetUserInput{
  UserName: aws.String(userName),
 })
```

```
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
  user = result.User
 }
 return user, err
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(userName string) (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(context.TODO(),
 &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}



// CreateUserPolicy adds an inline policy to a user. This example creates a
 policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(userName string, policyName string,
 actions []string,
```

```
  roleArn string) error {
 policyDoc := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Action: actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(context.TODO(),
 &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(userName string) ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(context.TODO(),
 &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
```

```
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(userName string, policyName string)
 error {
 _, err := wrapper.IamClient.DeleteUserPolicy(context.TODO(),
 &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(userName string) error {
 _, err := wrapper.IamClient.DeleteUser(context.TODO(), &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(userName string)
 (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(context.TODO(),
 &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
```

```
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(userName string, keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(context.TODO(),
 &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(context.TODO(),
 &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

IAM 사용자 작업을 래핑하는 함수를 생성합니다.

```
/*
  To run this Java V2 code example, set up your development environment,
 including your credentials.

  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html

  This example performs these operations:
```

```
  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3
 Service client object with the temporary credentials.
  6. Deletes the resources.
 */

public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static String userArn;

    public static void main(String[] args) throws Exception {

        final String usage = """

                Usage:
                    <username> <policyName> <roleName> <roleSessionName>
 <bucketName>\s

                Where:
                    username - The name of the IAM user to create.\s
                    policyName - The name of the policy to create.\s
                    roleName - The name of the role to create.\s
                    roleSessionName - The name of the session required for the
 assumeRole operation.\s
                    bucketName - The name of the Amazon S3 bucket from which
 objects are read.\s
                """;
```

```
        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        String policyName = args[1];
        String roleName = args[2];
        String roleSessionName = args[3];
        String bucketName = args[4];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IAM example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 1. Create the IAM user.");
        User createUser = createIAMUser(iam, userName);

        System.out.println(DASHES);
        userArn = createUser.arn();

        AccessKey myKey = createIAMAccessKey(iam, userName);
        String accessKey = myKey.accessKeyId();
        String secretKey = myKey.secretAccessKey();
        String assumeRolePolicyDocument = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                " \"AWS\": \"" + userArn + "\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        System.out.println(assumeRolePolicyDocument);
        System.out.println(userName + " was successfully created.");
```

```
        System.out.println(DASHES);
        System.out.println("2. Creates a policy.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully
created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the
role.");
        System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
```

```
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);

waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static String createIAMRole(IamClient iam, String rolename, String
json) {

        try {
```

```
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(json)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " +
response.role().arn());
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();
            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

```
   public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
       try {
           ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                   .roleName(roleName)
                   .build();

           ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
           List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
           String polArn;
           for (AttachedPolicy policy : attachedPolicies) {
               polArn = policy.policyArn();
               if (polArn.compareTo(policyArn) == 0) {
                   System.out.println(roleName + " policy is already attached to
this role.");
                   return;
               }
           }

           AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                   .roleName(roleName)
                   .policyArn(policyArn)
                   .build();

           iam.attachRolePolicy(attachRequest);
           System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

       } catch (IamException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   // Invoke an Amazon S3 operation using the Assumed Role.
   public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
           String keySecret) {

       // Use the creds of the new IAM user that was created in this code
example.
```

```
        AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
        StsClient stsClient = StsClient.builder()
                .region(Region.US_EAST_1)

.credentialsProvider(StaticCredentialsProvider.create(credentials))
                .build();

        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();
            String key = myCreds.accessKeyId();
            String secKey = myCreds.secretAccessKey();
            String secToken = myCreds.sessionToken();

            // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
            // invoking assumeRole.
            Region region = Region.US_EAST_1;
            S3Client s3 = S3Client.builder()
                    .credentialsProvider(

StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
                    .region(region)
                    .build();

            System.out.println("Created a S3Client using temp credentials.");
            System.out.println("Listing objects in " + bucketName);
            ListObjectsRequest listObjects = ListObjectsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.println("The name of the key is " + myValue.key());
                System.out.println("The owner is " + myValue.owner());
            }
```

```
        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteRole(IamClient iam, String roleName, String polArn)
{

        try {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                    .policyArn(polArn)
                    .roleName(roleName)
                    .build();

            iam.detachRolePolicy(rolePolicyRequest);

            // Delete the policy.
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(polArn)
                    .build();

            iam.deletePolicy(request);
            System.out.println("*** Successfully deleted " + polArn);

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteKey(IamClient iam, String username, String
accessKey) {
```

```
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("*** Successfully deleted " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey

- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할과 IAM 사용자를 생성합니다. 사용자는 역할을 수임할 수 있는 권한만 있습니다. 역할을 수임한 후 임시 자격 증명을 사용하여 계정의 버킷을 나열합니다.

```
import {
  CreateUserCommand,
  GetUserCommand,
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachRolePolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario/index.js";
```

```
// Set the parameters.
const iamClient = new IAMClient({});
const userName = "test_name";
const policyName = "test_policy";
const roleName = "test_role";

/**
 * Create a new IAM user. If the user already exists, give
 * the option to delete and re-create it.
 * @param {string} name
 */
export const createUser = async (name, confirmAll = false) => {
  try {
    const { User } = await iamClient.send(
      new GetUserCommand({ UserName: name }),
    );
    const input = new ScenarioInput(
      "deleteUser",
      "Do you want to delete and remake this user?",
      { type: "confirm" },
    );
    const deleteUser = await input.handle({}, { confirmAll });
    // If the user exists, and you want to delete it, delete the user
    // and then create it again.
    if (deleteUser) {
      await iamClient.send(new DeleteUserCommand({ UserName: User.UserName }));
      await iamClient.send(new CreateUserCommand({ UserName: name }));
    } else {
      console.warn(
        `${name} already exists. The scenario may not work as expected.`,
      );
      return User;
    }
  } catch (caught) {
    // If there is no user by that name, create one.
    if (caught instanceof Error && caught.name === "NoSuchEntityException") {
      const { User } = await iamClient.send(
        new CreateUserCommand({ UserName: name }),
      );
      return User;
    } else {
      throw caught;
    }
```

```
  }
};

export const main = async (confirmAll = false) => {
  // Create a user. The user has no permissions by default.
  const User = await createUser(userName, confirmAll);

  if (!User) {
    throw new Error("User not created");
  }

  // Create an access key. This key is used to authenticate the new user to
  // Amazon Simple Storage Service (Amazon S3) and AWS Security Token Service
 (AWS STS).
  // It's not best practice to use access keys. For more information, see
 https://aws.amazon.com/iam/resources/best-practices/.
  const createAccessKeyResponse = await iamClient.send(
    new CreateAccessKeyCommand({ UserName: userName }),
  );

  if (
    !createAccessKeyResponse.AccessKey?.AccessKeyId ||
    !createAccessKeyResponse.AccessKey?.SecretAccessKey
  ) {
    throw new Error("Access key not created");
  }

  const {
    AccessKey: { AccessKeyId, SecretAccessKey },
  } = createAccessKeyResponse;

  let s3Client = new S3Client({
    credentials: {
      accessKeyId: AccessKeyId,
      secretAccessKey: SecretAccessKey,
    },
  });

  // Retry the list buckets operation until it succeeds. InvalidAccessKeyId is
  // thrown while the user and access keys are still stabilizing.
  await retry({ intervalInMs: 1000, maxRetries: 300 }, async () => {
    try {
      return await listBuckets(s3Client);
    } catch (err) {
```

```
      if (err instanceof Error && err.name === "InvalidAccessKeyId") {
        throw err;
      }
    }
  });

  // Retry the create role operation until it succeeds. A MalformedPolicyDocument
error
  // is thrown while the user and access keys are still stabilizing.
  const { Role } = await retry(
    {
      intervalInMs: 2000,
      maxRetries: 60,
    },
    () =>
      iamClient.send(
        new CreateRoleCommand({
          AssumeRolePolicyDocument: JSON.stringify({
            Version: "2012-10-17",
            Statement: [
              {
                Effect: "Allow",
                Principal: {
                  // Allow the previously created user to assume this role.
                  AWS: User.Arn,
                },
                Action: "sts:AssumeRole",
              },
            ],
          }),
          RoleName: roleName,
        }),
      ),
  );

  if (!Role) {
    throw new Error("Role not created");
  }

  // Create a policy that allows the user to list S3 buckets.
  const { Policy: listBucketPolicy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyDocument: JSON.stringify({
        Version: "2012-10-17",
```

```
        Statement: [
          {
            Effect: "Allow",
            Action: ["s3:ListAllMyBuckets"],
            Resource: "*",
          },
        ],
      }),
      PolicyName: policyName,
    }),
  );

  if (!listBucketPolicy) {
    throw new Error("Policy not created");
  }

  // Attach the policy granting the 's3:ListAllMyBuckets' action to the role.
  await iamClient.send(
    new AttachRolePolicyCommand({
      PolicyArn: listBucketPolicy.Arn,
      RoleName: Role.RoleName,
    }),
  );

  // Assume the role.
  const stsClient = new STSClient({
    credentials: {
      accessKeyId: AccessKeyId,
      secretAccessKey: SecretAccessKey,
    },
  });

  // Retry the assume role operation until it succeeds.
  const { Credentials } = await retry(
    { intervalInMs: 2000, maxRetries: 60 },
    () =>
      stsClient.send(
        new AssumeRoleCommand({
          RoleArn: Role.Arn,
          RoleSessionName: `iamBasicScenarioSession-${Math.floor(
            Math.random() * 1000000,
          )}`,
          DurationSeconds: 900,
        }),
```

```
    ),
);

if (!Credentials?.AccessKeyId || !Credentials?.SecretAccessKey) {
  throw new Error("Credentials not created");
}

s3Client = new S3Client({
  credentials: {
    accessKeyId: Credentials.AccessKeyId,
    secretAccessKey: Credentials.SecretAccessKey,
    sessionToken: Credentials.SessionToken,
  },
});

// List the S3 buckets again.
// Retry the list buckets operation until it succeeds. AccessDenied might
// be thrown while the role policy is still stabilizing.
await retry({ intervalInMs: 2000, maxRetries: 60 }, () =>
  listBuckets(s3Client),
);

// Clean up.
await iamClient.send(
  new DetachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);

await iamClient.send(
  new DeletePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
  }),
);

await iamClient.send(
  new DeleteRoleCommand({
    RoleName: Role.RoleName,
  }),
);

await iamClient.send(
  new DeleteAccessKeyCommand({
```

```
      UserName: userName,
      AccessKeyId,
    }),
  );

  await iamClient.send(
    new DeleteUserCommand({
      UserName: userName,
    }),
  );
};

/**
 *
 * @param {S3Client} s3Client
 */
const listBuckets = async (s3Client) => {
  const { Buckets } = await s3Client.send(new ListBucketsCommand({}));

  if (!Buckets) {
    throw new Error("Buckets not listed");
  }

  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy

- DetachRolePolicy
- PutUserPolicy

Kotlin

SDK for Kotlin

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

IAM 사용자 작업을 래핑하는 함수를 생성합니다.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
 <bucketName>

    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
 operation.
        fileLocation - The file location to the JSON required to create the role
 (see Readme).
        bucketName - The name of the Amazon S3 bucket from which objects are
 read.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
```

```
    val roleSessionName = args[3]
    val fileLocation = args[4]
    val bucketName = args[5]

    createUser(userName)
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("*** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue: String =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
```

```
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?,
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
```

```
            roleName = roleNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role
 $roleNameVal")
    }
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
```

```
    roleSessionNameVal: String?,
    bucketName: String,
) {
    val stsClient =
        StsClient {
            region = "us-east-1"
        }

    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials =
        StaticCredentialsProvider {
            accessKeyId = key
            secretAccessKey = secKey
            sessionToken = secToken
        }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 =
        S3Client {
            credentialsProvider = staticCredentials
            region = "us-east-1"
        }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")

    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
```

```
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String,
) {
    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
```

```
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

PHP

SDK for PHP

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
echo("---------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("---------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
```

```
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
```

```
    echo "this should now not fail\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- API 세부 정보는 AWS SDK for PHP API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할과 IAM 사용자를 생성합니다. 사용자는 역할을 수임할 수 있는 권한만 있습니다. 역할을 수임한 후 임시 자격 증명을 사용하여 계정의 버킷을 나열합니다.

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError


def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print(".", end="")
        sys.stdout.flush()
    print()


def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.
```

```
    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
resource
                         that has permissions to create users, roles, and
policies
                         in the account.
    :return: The newly created user, user key, and role.
    """
    try:
        user = iam_resource.create_user(UserName=f"demo-user-{uuid4()}")
        print(f"Created user {user.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a user for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user_key = user.create_access_key_pair()
        print(f"Created access key pair for user.")
    except ClientError as error:
        print(
            f"Couldn't create access keys for user {user.name}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    try:
        role = iam_resource.create_role(
            RoleName=f"demo-role-{uuid4()}",
            AssumeRolePolicyDocument=json.dumps(
                {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"AWS": user.arn},
                            "Action": "sts:AssumeRole",
                        }
                    ],
                }
```

```
            ),
        )
        print(f"Created role {role.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a role for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        policy = iam_resource.create_policy(
            PolicyName=f"demo-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "s3:ListAllMyBuckets",
                            "Resource": "arn:aws:s3:::*",
                        }
                    ],
                }
            ),
        )
        role.attach_policy(PolicyArn=policy.arn)
        print(f"Created policy {policy.policy_name} and attached it to the
role.")
    except ClientError as error:
        print(
            f"Couldn't create a policy and attach it to role {role.name}. Here's
why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user.create_policy(
            PolicyName=f"demo-user-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version": "2012-10-17",
                    "Statement": [
```

```
                        {
                            "Effect": "Allow",
                            "Action": "sts:AssumeRole",
                            "Resource": role.arn,
                        }
                    ],
                }
            ),
        )
        print(
            f"Created an inline policy for {user.name} that lets the user assume
 "
            f"the role."
        )
    except ClientError as error:
        print(
            f"Couldn't create an inline policy for user {user.name}. Here's why:
 "
            f"{error.response['Error']['Message']}"
        )
        raise

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, role


def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
                     have permission to list buckets in the account.
    """
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        "s3", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
    )
    try:
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
```

```
        raise RuntimeError("Expected to get AccessDenied error when listing
 buckets!")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise


def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the
 account.
    Uses the temporary credentials from the role to list the buckets that are
 owned
    by the assumed role's account.

    :param user_key: The access key of a user that has permission to assume the
 role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        "sts", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
    )
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name
        )
        temp_credentials = response["Credentials"]
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(
            f"Couldn't assume role {assume_role_arn}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    # Create an S3 resource that can access the account with the temporary
 credentials.
    s3_resource = boto3.resource(
        "s3",
```

```
            aws_access_key_id=temp_credentials["AccessKeyId"],
            aws_secret_access_key=temp_credentials["SecretAccessKey"],
            aws_session_token=temp_credentials["SessionToken"],
        )
        print(f"Listing buckets for the assumed role's account:")
        try:
            for bucket in s3_resource.buckets.all():
                print(bucket.name)
        except ClientError as error:
            print(
                f"Couldn't list buckets for the account. Here's why: "
                f"{error.response['Error']['Message']}"
            )
            raise




def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print(
            "Couldn't detach policy, delete policy, or delete role. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
```

```
        for key in user.access_keys.all():
            key.delete()
            print("Deleted user's access key.")
        user.delete()
        print(f"Deleted {user.name}.")
    except ClientError as error:
        print(
            "Couldn't delete user policy or delete user. Here's why: "
            f"{error.response['Error']['Message']}"
        )


def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the IAM create user and assume role demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn,
 "AssumeRoleDemoSession")
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
        print("Thanks for watching!")


if __name__ == "__main__":
    usage_demo()
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy

- CreateRole
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할과 IAM 사용자를 생성합니다. 사용자는 역할을 수임할 수 있는 권한만 있습니다. 역할을 수임한 후 임시 자격 증명을 사용하여 계정의 버킷을 나열합니다.

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
```

```
    puts("Give AWS time to propagate resources...")
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_client.create_user(user_name: user_name).user
    @logger.info("Created demo user named #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Tried and failed to create demo user.")
    @logger.info("\t#{e.code}: #{e.message}")
    @logger.info("\nCan't continue the demo without a user!")
    raise
  else
    user
  end

  # Creates an access key for a user.
  #
  # @param user [Aws::IAM::User] The user that owns the key.
  # @return [Aws::IAM::AccessKeyPair] The newly created access key.
  def create_access_key_pair(user)
    user_key = @iam_client.create_access_key(user_name:
user.user_name).access_key
    @logger.info("Created accesskey pair for user #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create access keys for user #{user.user_name}.")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    user_key
  end

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
```

```
      Version: "2012-10-17",
      Statement: [{
                   Effect: "Allow",
                   Principal: {'AWS': user.arn},
                   Action: "sts:AssumeRole"
                 }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
    @logger.info("Created role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a role for the demo. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account,
and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
      Version: "2012-10-17",
      Statement: [{
                   Effect: "Allow",
                   Action: "s3:ListAllMyBuckets",
                   Resource: "arn:aws:s3:::*"
                 }]
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
```

```
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role
#{role.role_name}. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an inline policy for a user that lets the user assume a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param user [Aws::IAM::User] The user that owns the policy.
  # @param role [Aws::IAM::Role] The role that can be assumed.
  # @return [Aws::IAM::UserPolicy] The newly created policy.
  def create_user_policy(policy_name, user, role)
    policy_document = {
      Version: "2012-10-17",
      Statement: [{
                    Effect: "Allow",
                    Action: "sts:AssumeRole",
                    Resource: role.arn
                  }]
    }.to_json
    @iam_client.put_user_policy(
      user_name: user.user_name,
      policy_name: policy_name,
      policy_document: policy_document
    )
    puts("Created an inline policy for #{user.user_name} that lets the user
assume role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an Amazon S3 resource with specified credentials. This is separated
into a
  # factory function so that it can be mocked for unit testing.
  #
  # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
```

```
 def create_s3_resource(credentials)
   Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
 end

 # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
 # Because the resource uses credentials with limited access, it may not be able
to
 # list the S3 buckets.
 #
 # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
 def list_buckets(s3_resource)
   count = 10
   s3_resource.buckets.each do |bucket|
     @logger.info "\t#{bucket.name}"
     count -= 1
     break if count.zero?
   end
 rescue Aws::Errors::ServiceError => e
   if e.code == "AccessDenied"
     puts("Attempt to list buckets with no permissions: AccessDenied.")
   else
     @logger.info("Couldn't list buckets for the account. Here's why: ")
     @logger.info("\t#{e.code}: #{e.message}")
     raise
   end
 end

 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
```

```
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: "create-use-assume-role-scenario"
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end

 # Deletes a role. If the role has policies attached, they are detached and
 # deleted before the role is deleted.
 #
 # @param role_name [String] The name of the role to delete.
 def delete_role(role_name)
   @iam_client.list_attached_role_policies(role_name:
role_name).attached_policies.each do |policy|
     @iam_client.detach_role_policy(role_name: role_name, policy_arn:
policy.policy_arn)
     @iam_client.delete_policy(policy_arn: policy.policy_arn)
     @logger.info("Detached and deleted policy #{policy.policy_name}.")
   end
   @iam_client.delete_role({ role_name: role_name })
   @logger.info("Role deleted: #{role_name}.")
 rescue Aws::Errors::ServiceError => e
   @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
why:")
   @logger.info("\t#{e.code}: #{e.message}")
   raise
 end

 # Deletes a user. If the user has inline policies or access keys, they are
deleted
 # before the user is deleted.
 #
 # @param user [Aws::IAM::User] The user to delete.
 def delete_user(user_name)
   user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
   user.each do |key|
     @iam_client.delete_access_key({ access_key_id: key.access_key_id,
user_name: user_name })
```

```
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the IAM create a user and assume a role demo!")
  puts("-" * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-
#{Random.uuid}", role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user,
 role)
  scenario.wait(10)
  puts("Try to list buckets with credentials for a user who has no permissions.")
  puts("Expect AccessDenied from this call.")
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key)))
  puts("Now, assume the role that grants permission.")
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key))
  puts("Here are your buckets:")
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts("Thanks for watching!")
  puts("-" * 88)
rescue Aws::Errors::ServiceError => e
  puts("Something went wrong with the demo.")
```

```
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
 $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{config::Credentials as iamCredentials, config::Region, Client
 as iamClient};
```

```
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document)
 =
        initialize_variables().await;

    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{:?}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = iamClient::new(&shared_config);
    let uuid = Uuid::new_v4().to_string();

    let list_all_buckets_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
    }"
    .to_string();
    let inline_policy_document = "{
                \"Version\": \"2012-10-17\",
```

```
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{}\"}]
    }"
    .to_string();

    (
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}{}",
 "iam_demo_user_", uuid)).await?;
    println!("Created the user with the name: {}", user.user_name());
    let key = iam_service::create_access_key(&client, user.user_name()).await?;

    let assume_role_policy_document = "{
        \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }"
    .to_string()
    .replace("{}", user.arn());

    let assume_role_role = iam_service::create_role(
        &client,
        &format!("{}{}", "iam_demo_role_", uuid),
        &assume_role_policy_document,
    )
    .await?;
    println!("Created the role with the ARN: {}", assume_role_role.arn());
```

```
    let list_all_buckets_policy = iam_service::create_policy(
        &client,
        &format!("{}{}", "iam_demo_policy_", uuid),
        &list_all_buckets_policy_document,
    )
    .await?;
    println!(
        "Created policy: {}",
        list_all_buckets_policy.policy_name.as_ref().unwrap()
    );

    let attach_role_policy_result =
        iam_service::attach_role_policy(&client, &assume_role_role,
&list_all_buckets_policy)
            .await?;
    println!(
        "Attached the policy to the role: {:?}",
        attach_role_policy_result
    );

    let inline_policy_name = format!("{}{}", "iam_demo_inline_policy_", uuid);
    let inline_policy_document = inline_policy_document.replace("{}",
assume_role_role.arn());
    iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
        .await?;
    println!("Created inline policy.");

    //First, fail to list the buckets with the user.
    let creds = iamCredentials::from_keys(key.access_key_id(),
key.secret_access_key(), None);
    let fail_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    println!("Fail config: {:?}", fail_config);
    let fail_client: s3Client = s3Client::new(&fail_config);
    match fail_client.list_buckets().send().await {
        Ok(e) => {
            println!("This should not run. {:?}", e);
        }
        Err(e) => {
            println!("Successfully failed with error: {:?}", e)
```

```
        }
    }

    let sts_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    let sts_client: stsClient = stsClient::new(&sts_config);
    sleep(Duration::from_secs(10)).await;
    let assumed_role = sts_client
        .assume_role()
        .role_arn(assume_role_role.arn())
        .role_session_name(&format!("{}{}", "iam_demo_assumerole_session_",
uuid))
        .send()
        .await;
    println!("Assumed role: {:?}", assumed_role);
    sleep(Duration::from_secs(10)).await;

    let assumed_credentials = iamCredentials::from_keys(
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .access_key_id(),
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .secret_access_key(),
        Some(
            assumed_role
                .as_ref()
                .unwrap()
                .credentials
                .as_ref()
                .unwrap()
                .session_token
                .clone(),
        ),
```

```
    );

    let succeed_config = aws_config::from_env()
        .credentials_provider(assumed_credentials)
        .load()
        .await;
    println!("succeed config: {:?}", succeed_config);
    let succeed_client: s3Client = s3Client::new(&succeed_config);
    sleep(Duration::from_secs(10)).await;
    match succeed_client.list_buckets().send().await {
        Ok(_) => {
            println!("This should now run successfully.")
        }
        Err(e) => {
            println!("This should not run. {:?}", e);
            panic!()
        }
    }

    //Clean up.
    iam_service::detach_role_policy(
        &client,
        assume_role_role.role_name(),
        list_all_buckets_policy.arn().unwrap_or_default(),
    )
    .await?;
    iam_service::delete_policy(&client, list_all_buckets_policy).await?;
    iam_service::delete_role(&client, &assume_role_role).await?;
    println!("Deleted role {}", assume_role_role.role_name());
    iam_service::delete_access_key(&client, &user, &key).await?;
    println!("Deleted key for {}", key.user_name());
    iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
    println!("Deleted inline user policy: {}", inline_policy_name);
    iam_service::delete_user(&client, &user).await?;
    println!("Deleted user {}", user.user_name());

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey

- CreatePolicy
- CreateRole
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 읽기 전용 및 읽기-쓰기 IAM 사용자 생성

다음 코드 예제에서는 사용자를 생성하고 정책을 사용자에게 연결하는 방법을 보여줍니다.

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 자격 증명 공급자를 통한 페더레이션을 사용하세요.

- 두 IAM 사용자를 생성합니다.
- Amazon S3 버킷의 객체를 가져와 넣기 위한 정책을 한 명의 사용자에게 연결합니다.
- 두 번째 사용자가 버킷에서 객체를 가져오도록 하기 위한 정책을 연결합니다.
- 사용자 보안 인증에 따라 버킷에 대해 서로 다른 권한이 부여됩니다.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

IAM 사용자 작업을 래핑하는 함수를 생성합니다.

```
import logging
import time

import boto3
from botocore.exceptions import ClientError

import access_key_wrapper
import policy_wrapper

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user



def update_user(user_name, new_user_name):
```

```
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user



def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users



def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
```

```
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise



def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise



def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

IAM 정책 작업을 래핑하는 함수를 생성합니다.

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::my-bucket/*' to allow actions on all
 objects
                         in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
```

```
        raise
    else:
        return policy



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

IAM 액세스 키 작업을 래핑하는 함수를 생성합니다.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
```

```
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

래퍼 함수를 사용하여 다른 정책을 가진 사용자를 생성하고 자격 증명을 사용하여 Amazon S3 버킷에 액세스합니다.

```
def usage_demo():
    """
    Shows how to manage users, keys, and policies.
    This demonstration creates two users: one user who can put and get objects in
 an
    Amazon S3 bucket, and another user who can only get objects from the bucket.
    The demo then shows how the users can perform only the actions they are
 permitted
    to perform.
```

```
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management user demo.")
    print("-" * 88)
    print(
        "Users can have policies and roles attached to grant them specific "
        "permissions."
    )
    s3 = boto3.resource("s3")
    bucket = s3.create_bucket(
        Bucket=f"demo-iam-bucket-{time.time_ns()}",
        CreateBucketConfiguration={
            "LocationConstraint": s3.meta.client.meta.region_name
        },
    )
    print(f"Created an Amazon S3 bucket named {bucket.name}.")
    user_read_writer = create_user("demo-iam-read-writer")
    user_reader = create_user("demo-iam-reader")
    print(f"Created two IAM users: {user_read_writer.name} and
{user_reader.name}")
    update_user(user_read_writer.name, "demo-iam-creator")
    update_user(user_reader.name, "demo-iam-getter")
    users = list_users()
    user_read_writer = next(
        user for user in users if user.user_id == user_read_writer.user_id
    )
    user_reader = next(user for user in users if user.user_id ==
user_reader.user_id)
    print(
        f"Changed the names of the users to {user_read_writer.name} "
        f"and {user_reader.name}."
    )

    read_write_policy = policy_wrapper.create_policy(
        "demo-iam-read-write-policy",
        "Grants rights to create and get an object in the demo bucket.",
        ["s3:PutObject", "s3:GetObject"],
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(
        f"Created policy {read_write_policy.policy_name} with ARN:
{read_write_policy.arn}"
    )
```

```
    print(read_write_policy.description)
    read_policy = policy_wrapper.create_policy(
        "demo-iam-read-policy",
        "Grants rights to get an object from the demo bucket.",
        "s3:GetObject",
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(f"Created policy {read_policy.policy_name} with ARN:
{read_policy.arn}")
    print(read_policy.description)
    attach_policy(user_read_writer.name, read_write_policy.arn)
    print(f"Attached {read_write_policy.policy_name} to
{user_read_writer.name}.")
    attach_policy(user_reader.name, read_policy.arn)
    print(f"Attached {read_policy.policy_name} to {user_reader.name}.")

    user_read_writer_key = access_key_wrapper.create_key(user_read_writer.name)
    print(f"Created access key pair for {user_read_writer.name}.")
    user_reader_key = access_key_wrapper.create_key(user_reader.name)
    print(f"Created access key pair for {user_reader.name}.")

    s3_read_writer_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_read_writer_key.id,
        aws_secret_access_key=user_read_writer_key.secret,
    )
    demo_object_key = f"object-{time.time_ns()}"
    demo_object = None
    while demo_object is None:
        try:
            demo_object = s3_read_writer_resource.Bucket(bucket.name).put_object(
                Key=demo_object_key, Body=b"AWS IAM demo object content!"
            )
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise
    print(
        f"Put {demo_object_key} into {bucket.name} using "
        f"{user_read_writer.name}'s credentials."
    )
```

```
    read_writer_object = s3_read_writer_resource.Bucket(bucket.name).Object(
        demo_object_key
    )
    read_writer_content = read_writer_object.get()["Body"].read()
    print(f"Got object {read_writer_object.key} using read-writer user's
credentials.")
    print(f"Object content: {read_writer_content}")

    s3_reader_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_reader_key.id,
        aws_secret_access_key=user_reader_key.secret,
    )
    demo_content = None
    while demo_content is None:
        try:
            demo_object =
s3_reader_resource.Bucket(bucket.name).Object(demo_object_key)
            demo_content = demo_object.get()["Body"].read()
            print(f"Got object {demo_object.key} using reader user's
credentials.")
            print(f"Object content: {demo_content}")
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise

    try:
        demo_object.delete()
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("-" * 88)
            print(
                "Tried to delete the object using the reader user's credentials.
"
                "Got expected AccessDenied error because the reader is not "
                "allowed to delete objects."
            )
            print("-" * 88)

    access_key_wrapper.delete_key(user_reader.name, user_reader_key.id)
    detach_policy(user_reader.name, read_policy.arn)
```

```
    policy_wrapper.delete_policy(read_policy.arn)
    delete_user(user_reader.name)
    print(f"Deleted keys, detached and deleted policy, and deleted
{user_reader.name}.")

    access_key_wrapper.delete_key(user_read_writer.name, user_read_writer_key.id)
    detach_policy(user_read_writer.name, read_write_policy.arn)
    policy_wrapper.delete_policy(read_write_policy.arn)
    delete_user(user_read_writer.name)
    print(
        f"Deleted keys, detached and deleted policy, and deleted
{user_read_writer.name}."
    )

    bucket.objects.delete()
    bucket.delete()
    print(f"Emptied and deleted {bucket.name}.")
    print("Thanks for watching!")
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - AttachUserPolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteUser
  - DetachUserPolicy
  - ListUsers
  - UpdateUser

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 IAM 액세스 키 관리

다음 코드 예제에서는 액세스 키를 관리하는 방법을 보여줍니다.

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 자격 증명 공급자를 통한 페더레이션을 사용하세요.

- 액세스 키를 생성하고 나열합니다.
- 액세스 키가 마지막으로 사용된 시기 및 방법을 조회합니다.
- 액세스 키를 업데이트 및 삭제합니다.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

IAM 액세스 키 작업을 래핑하는 함수를 생성합니다.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
```

```
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys



def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
```

```
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response



def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
 deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
 key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate",
 key_id
        )
        raise
```

```
def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

래퍼 함수를 사용하여 현재 사용자에 대한 액세스 키 작업을 수행합니다.

```
def usage_demo():
    """Shows how to create and manage access keys."""

    def print_keys():
        """Gets and prints the current keys for a user."""
        current_keys = list_keys(current_user_name)
        print("The current user's keys are now:")
        print(*[f"{key.id}: {key.status}" for key in current_keys], sep="\n")

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management access key demo.")
    print("-" * 88)
    current_user_name = iam.CurrentUser().user_name
    print(
        f"This demo creates an access key for the current user "
        f"({current_user_name}), manipulates the key in a few ways, and then "
        f"deletes it."
    )
    all_keys = list_keys(current_user_name)
    if len(all_keys) == 2:
```

```
        print(
            "The current user already has the maximum of 2 access keys. To run "
            "this demo, either delete one of the access keys or use a user "
            "that has only 1 access key."
        )
    else:
        new_key = create_key(current_user_name)
        print(f"Created a new key with id {new_key.id} and secret
{new_key.secret}.")
        print_keys()
        existing_key = next(key for key in all_keys if key != new_key)
        last_use = get_last_use(existing_key.id)["AccessKeyLastUsed"]
        print(
            f"Key {all_keys[0].id} was last used to access
{last_use['ServiceName']} "
            f"on {last_use['LastUsedDate']}"
        )
        update_key(current_user_name, new_key.id, False)
        print(f"Key {new_key.id} is now deactivated.")
        print_keys()
        delete_key(current_user_name, new_key.id)
        print_keys()
        print("Thanks for watching!")
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - CreateAccessKey
  - DeleteAccessKey
  - GetAccessKeyLastUsed
  - ListAccessKeys
  - UpdateAccessKey

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 IAM 정책 관리

다음 코드 예시는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 정책을 생성하고 나열합니다.
- 정책 버전을 생성하고 가져옵니다.
- 정책을 이전 버전으로 롤백합니다.
- 정책을 삭제합니다.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

IAM 정책 작업을 래핑하는 함수를 생성합니다.

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
```

```
                          'arn:aws:s3:::my-bucket/*' to allow actions on all
 objects
                          in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy



def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are
 returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies
```

```
def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version



def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
```

```
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement



def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
```

```
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
        logger.info("Set %s as the default version.",
 rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
 default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
                "Default version %s found for %s, but no previous version exists,
 so "
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
        return rollback_version



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

래퍼 함수를 사용하여 정책을 생성하고, 버전을 업데이트하고, 정책에 대한 정보를 얻습니다.

```
def usage_demo():
    """Shows how to use the policy functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management policy demo.")
    print("-" * 88)
    print(
        "Policies let you define sets of permissions that can be attached to "
        "other IAM resources, like users and roles."
    )
    bucket_arn = f"arn:aws:s3:::made-up-bucket-name"
    policy = create_policy(
        "demo-iam-policy",
        "Policy for IAM demonstration.",
        ["s3:ListObjects"],
        bucket_arn,
    )
    print(f"Created policy {policy.policy_name}.")
    policies = list_policies("Local")
    print(f"Your account has {len(policies)} managed policies:")
    print(*[pol.policy_name for pol in policies], sep=", ")
    time.sleep(1)
    policy_version = create_policy_version(
        policy.arn, ["s3:PutObject"], bucket_arn, True
    )
    print(
        f"Added policy version {policy_version.version_id} to policy "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is:")
    pprint.pprint(default_statement)
    rollback_version = rollback_policy_version(policy.arn)
    print(
        f"Rolled back to version {rollback_version.version_id} for "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is now:")
    pprint.pprint(default_statement)
    delete_policy(policy.arn)
    print(f"Deleted policy {policy.policy_name}.")
```

```
    print("Thanks for watching!")
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - CreatePolicy
  - CreatePolicyVersion
  - DeletePolicy
  - DeletePolicyVersion
  - GetPolicyVersion
  - ListPolicies
  - ListPolicyVersions
  - SetDefaultPolicyVersion

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 IAM 역할 관리

다음 코드 예시는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- IAM 역할을 생성합니다.
- 역할에 대한 정책을 연결 및 분리합니다.
- 역할을 삭제합니다.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

IAM 역할 작업을 래핑하는 함수를 생성합니다.

```
import json
import logging
import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role
```

```
def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise



def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise



def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
```

```
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

래퍼 함수를 사용하여 역할을 생성한 다음 정책을 연결하고 분리합니다.

```
def usage_demo():
    """Shows how to use the role functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management role demo.")
    print("-" * 88)
    print(
        "Roles let you define sets of permissions and can be assumed by "
        "other entities, like users and services."
    )
    print("The first 10 roles currently in your account are:")
    roles = list_roles(10)
    print(f"The inline policies for role {roles[0].name} are:")
    list_policies(roles[0].name)
    role = create_role(
        "demo-iam-role", ["lambda.amazonaws.com",
 "batchoperations.s3.amazonaws.com"]
    )
    print(f"Created role {role.name}, with trust policy:")
    pprint.pprint(role.assume_role_policy_document)
    policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
    attach_policy(role.name, policy_arn)
    print(f"Attached policy {policy_arn} to {role.name}.")
    print(f"Policies attached to role {role.name} are:")
    list_attached_policies(role.name)
    detach_policy(role.name, policy_arn)
    print(f"Detached policy {policy_arn} from {role.name}.")
    delete_role(role.name)
    print(f"Deleted {role.name}.")
    print("Thanks for watching!")
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateRole
  - DeleteRole
  - DetachRolePolicy

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 IAM 계정 관리

다음 코드 예시는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 계정 별칭을 가져오고 업데이트합니다.
- 사용자 및 보안 인증에 대한 보고서를 생성합니다.
- 계정 사용량 요약을 가져옵니다.
- 서로의 관계를 포함하여 계정의 모든 사용자, 그룹, 역할 및 정책에 대한 세부 정보를 가져옵니다.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

IAM 계정 작업을 래핑하는 함수를 생성합니다.

```
import logging
import pprint
import sys
import time
import boto3
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]



def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of
 the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise
```

```
def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise



def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four
 hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
 account.")
        raise
    else:
        return response



def get_credential_report():
    """
    Gets the most recently generated credentials report about the current
 account.
```

```
    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]



def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map



def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report,
 such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
```

```
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

래퍼 함수를 호출하여 계정 별칭을 변경하고 계정에 대한 보고서를 가져옵니다.

```
def usage_demo():
    """Shows how to use the account functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management account demo.")
    print("-" * 88)
    print(
        "Setting an account alias lets you use the alias in your sign-in URL "
        "instead of your account number."
    )
    old_aliases = list_aliases()
    if len(old_aliases) > 0:
        print(f"Your account currently uses '{old_aliases[0]}' as its alias.")
    else:
        print("Your account currently has no alias.")
    for index in range(1, 3):
        new_alias = f"alias-{index}-{time.time_ns()}"
        print(f"Setting your account alias to {new_alias}")
        create_alias(new_alias)
    current_aliases = list_aliases()
    print(f"Your account alias is now {current_aliases}.")
    delete_alias(current_aliases[0])
    print(f"Your account now has no alias.")
    if len(old_aliases) > 0:
        print(f"Restoring your original alias back to {old_aliases[0]}...")
        create_alias(old_aliases[0])

    print("-" * 88)
    print("You can get various reports about your account.")
    print("Let's generate a credentials report...")
```

```
    report_state = None
    while report_state != "COMPLETE":
        cred_report_response = generate_credential_report()
        old_report_state = report_state
        report_state = cred_report_response["State"]
        if report_state != old_report_state:
            print(report_state, sep="")
        else:
            print(".", sep="")
        sys.stdout.flush()
        time.sleep(1)
    print()
    cred_report = get_credential_report()
    col_count = 3
    print(f"Got credentials report. Showing only the first {col_count} columns.")
    cred_lines = [
        line.split(",")[:col_count] for line in
 cred_report.decode("utf-8").split("\n")
    ]
    col_width = max([len(item) for line in cred_lines for item in line]) + 2
    for line in cred_report.decode("utf-8").split("\n"):
        print(
            "".join(element.ljust(col_width) for element in line.split(",")
[:col_count])
        )

    print("-" * 88)
    print("Let's get an account summary.")
    summary = get_summary()
    print("Here's your summary:")
    pprint.pprint(summary)

    print("-" * 88)
    print("Let's get authorization details!")
    details = get_authorization_details([])
    see_details = input("These are pretty long, do you want to see them (y/n)? ")
    if see_details.lower() == "y":
        pprint.pprint(details)

    print("-" * 88)
    pw_policy_created = None
    see_pw_policy = input("Want to see the password policy for the account (y/n)?
 ")
    if see_pw_policy.lower() == "y":
```

```
        while True:
            if print_password_policy():
                break
            else:
                answer = input(
                    "Do you want to create a default password policy (y/n)? "
                )
                if answer.lower() == "y":
                    pw_policy_created = iam.create_account_password_policy()
                else:
                    break
    if pw_policy_created is not None:
        answer = input("Do you want to delete the password policy (y/n)? ")
        if answer.lower() == "y":
            pw_policy_created.delete()
            print("Password policy deleted.")

    print("The SAML providers for your account are:")
    list_saml_providers(10)

    print("-" * 88)
    print("Thanks for watching.")
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - CreateAccountAlias
  - DeleteAccountAlias
  - GenerateCredentialReport
  - GetAccountAuthorizationDetails
  - GetAccountSummary
  - GetCredentialReport
  - ListAccountAliases

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 IAM 정책 버전 롤백

다음 코드 예시는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 정책 버전 목록을 날짜별로 순서대로 가져옵니다.
- 기본 정책 버전을 찾습니다.
- 이전 정책 버전을 기본값으로 설정합니다.
- 이전 기본 버전을 삭제합니다.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise
```

```
    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
        logger.info("Set %s as the default version.",
rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
                "Default version %s found for %s, but no previous version exists,
so "
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
        return rollback_version
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - DeletePolicyVersion
  - ListPolicyVersions
  - SetDefaultPolicyVersion

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 IAM 정책 빌더 API 작업

다음 코드 예시는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 객체 지향 API를 사용하여 IAM 정책을 생성합니다.
- IAM 서비스에 IAM 정책 빌더 API를 사용합니다.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

예제에서는 다음 가져오기를 사용합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
```

```
import java.util.Arrays;
import java.util.List;
```

시간 기반 정책을 생성합니다.

```
        public String timeBasedPolicyExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")
                                                .addResource(IamResource.ALL)
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.DATE_GREATER_THAN)

 .key("aws:CurrentTime")

 .value("2020-04-01T00:00:00Z"))
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.DATE_LESS_THAN)

 .key("aws:CurrentTime")

 .value("2020-06-30T23:59:59Z")))
                                .build();

                // Use an IamPolicyWriter to write out the JSON string to a more
 readable
                // format.
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true)
                                .build());
        }
```

여러 조건이 포함된 정책을 생성합니다.

```
        public String multipleConditionsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
```

```
                                                .addAction("dynamodb:GetItem")

 .addAction("dynamodb:BatchGetItem")
                                                .addAction("dynamodb:Query")
                                                .addAction("dynamodb:PutItem")
                                                .addAction("dynamodb:UpdateItem")
                                                .addAction("dynamodb:DeleteItem")

 .addAction("dynamodb:BatchWriteItem")

 .addResource("arn:aws:dynamodb:*:*:table/table-name")

 .addConditions(IamConditionOperator.STRING_EQUALS

 .addPrefix("ForAllValues:"),

 "dynamodb:Attributes",
                                                                    List.of("column-
name1", "column-name2", "column-name3"))
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS

 .addSuffix("IfExists"))

 .key("dynamodb:Select")

 .value("SPECIFIC_ATTRIBUTES")))
                                .build();

                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

정책에 보안 주체를 사용합니다.

```
        public String specifyPrincipalsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.DENY)
                                                .addAction("s3:*")
                                                .addPrincipal(IamPrincipal.ALL)
```

```
 .addResource("arn:aws:s3:::BUCKETNAME/*")

 .addResource("arn:aws:s3:::BUCKETNAME")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.ARN_NOT_EQUALS)

 .key("aws:PrincipalArn")

 .value("arn:aws:iam::444455556666:user/user-name")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

교차 계정 액세스를 허용합니다.

```
        public String allowCrossAccountAccessExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)

 .addPrincipal(IamPrincipalType.AWS, "111122223333")
                                                .addAction("s3:PutObject")
                                                .addResource("arn:aws:s3:::DOC-
EXAMPLE-BUCKET/*")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS)
                                                                        .key("s3:x-amz-
acl")
                                                                        .value("bucket-
owner-full-control")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

IamPolicy를 빌드하고 업로드합니다.

```
       public String createAndUploadPolicyExample(IamClient iam, String
 accountID, String policyName) {
               // Build the policy.
               IamPolicy policy = IamPolicy.builder() // 'version' defaults to
 "2012-10-17".
                               .addStatement(IamStatement.builder()
                                               .effect(IamEffect.ALLOW)
                                               .addAction("dynamodb:PutItem")

 .addResource("arn:aws:dynamodb:us-east-1:" + accountID
                                                                       + ":table/
exampleTableName")
                                               .build())
                               .build();
               // Upload the policy.
               iam.createPolicy(r ->
 r.policyName(policyName).policyDocument(policy.toJson()));
               return
 policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
       }
```

IamPolicy를 다운로드하고 사용합니다.

```
       public String createNewBasedOnExistingPolicyExample(IamClient iam, String
 accountID, String policyName,
                       String newPolicyName) {

               String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
 policyName;
               GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
 r.policyArn(policyArn));

               String policyVersion =
 getPolicyResponse.policy().defaultVersionId();
               GetPolicyVersionResponse getPolicyVersionResponse = iam
                               .getPolicyVersion(r ->
 r.policyArn(policyArn).versionId(policyVersion));

               // Create an IamPolicy instance from the JSON string returned
 from IAM.
               String decodedPolicy =
 URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
```

```
                                StandardCharsets.UTF_8);
                IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

                /*
                 * All IamPolicy components are immutable, so use the copy method
that creates a
                 * new instance that
                 * can be altered in the same method call.
                 *
                 * Add the ability to get an item from DynamoDB as an additional
action.
                 */
                IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

                // Create a new statement that replaces the original statement.
                IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

                // Upload the new policy. IAM now has both policies.
                iam.createPolicy(r -> r.policyName(newPolicyName)
                                .policyDocument(newPolicy.toJson()));

                return
newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

- 자세한 정보는 AWS SDK for Java 2.x 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - CreatePolicy
  - GetPolicy
  - GetPolicyVersion

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용한 AWS STS 코드 예제

다음 코드 예제는 AWS STS를 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여 주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

코드 예시

- AWS SDK를 사용한 AWS STS 코드 예제
  - AWS SDK를 사용한 AWS STS 작업
    - AWS SDK 또는 CLI와 함께 AssumeRole 사용
    - AWS SDK 또는 CLI와 함께 AssumeRoleWithWebIdentity 사용
    - AWS SDK 또는 CLI와 함께 DecodeAuthorizationMessage 사용
    - AWS SDK 또는 CLI와 함께 GetFederationToken 사용
    - AWS SDK 또는 CLI와 함께 GetSessionToken 사용
- AWS SDK를 사용하는 AWS STS 시나리오
  - AWS SDK를 사용하여 AWS STS에서 MFA 토큰이 필요한 IAM 역할 수임
  - AWS SDK를 사용하여 페더레이션 사용자를 위해 AWS STS로 URL 구성
  - AWS SDK를 사용하여 AWS STS에서 MFA 토큰이 필요한 세션 토큰 가져오기

AWS SDK를 사용한 AWS STS 코드 예제

다음 코드 예제에서는 AWS SDK에서 AWS Security Token Service(AWS STS)의 기본 사항을 사용하는 방법을 보여줍니다.

예시

- AWS SDK를 사용한 AWS STS 작업
  - AWS SDK 또는 CLI와 함께 AssumeRole 사용
  - AWS SDK 또는 CLI와 함께 AssumeRoleWithWebIdentity 사용
  - AWS SDK 또는 CLI와 함께 DecodeAuthorizationMessage 사용
  - AWS SDK 또는 CLI와 함께 GetFederationToken 사용

- AWS SDK 또는 CLI와 함께 GetSessionToken 사용

AWS SDK를 사용한 AWS STS 작업

다음 코드 예제는 AWS SDK를 통해 개별 AWS STS 작업을 수행하는 방법을 보여줍니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

이들 발췌문은 AWS STS API를 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. AWS SDK를 사용하는 AWS STS 시나리오 에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 AWS Security Token Service(AWS STS) API 참조를 참조하세요.

예시

- AWS SDK 또는 CLI와 함께 AssumeRole 사용
- AWS SDK 또는 CLI와 함께 AssumeRoleWithWebIdentity 사용
- AWS SDK 또는 CLI와 함께 DecodeAuthorizationMessage 사용
- AWS SDK 또는 CLI와 함께 GetFederationToken 사용
- AWS SDK 또는 CLI와 함께 GetSessionToken 사용

AWS SDK 또는 CLI와 함께 **`AssumeRole`** 사용

다음 코드 예제는 `AssumeRole`의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- MFA 토큰이 필요한 IAM 역할 수임
- 페더레이션 사용자를 위해 URL 구성

.NET

AWS SDK for .NET

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

namespace AssumeRoleExample
{
    class AssumeRole
    {
        /// <summary>
        /// This example shows how to use the AWS Security Token
        /// Service (AWS STS) to assume an IAM role.
        ///
        /// NOTE: It is important that the role that will be assumed has a
        /// trust relationship with the account that will assume the role.
        ///
        /// Before you run the example, you need to create the role you want to
        /// assume and have it trust the IAM account that will assume that role.
        ///
        /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html
        /// for help in working with roles.
        /// </summary>

        private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

        static async Task Main()
        {
            // Create the SecurityToken client and then display the identity of
 the
            // default user.
```

```
            var roleArnToAssume = "arn:aws:iam::123456789012:role/
testAssumeRole";

            var client = new
 Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

            // Get and display the information about the identity of the default
 user.
            var callerIdRequest = new GetCallerIdentityRequest();
            var caller = await client.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"Original Caller: {caller.Arn}");

            // Create the request to use with the AssumeRoleAsync call.
            var assumeRoleReq = new AssumeRoleRequest()
            {
                DurationSeconds = 1600,
                RoleSessionName = "Session1",
                RoleArn = roleArnToAssume
            };

            var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);

            // Now create a new client based on the credentials of the caller
 assuming the role.
            var client2 = new AmazonSecurityTokenServiceClient(credentials:
 assumeRoleRes.Credentials);

            // Get and display information about the caller that has assumed the
 defined role.
            var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"AssumedRole Caller: {caller2.Arn}");
        }
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 AssumeRole을 참조하십시오.

Bash

Bash 스크립트와 함께 AWS CLI사용

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function sts_assume_role
#
# This function assumes a role in the AWS account and returns the temporary
#  credentials.
#
# Parameters:
#       -n role_session_name -- The name of the session.
#       -r role_arn -- The ARN of the role to assume.
#
# Returns:
```

```
#       [access_key_id, secret_access_key, session_token]
#     And:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function sts_assume_role() {
  local role_session_name role_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function sts_assume_role"
    echo "Assumes a role in the AWS account and returns the temporary
 credentials:"
    echo "  -n role_session_name -- The name of the session."
    echo "  -r role_arn -- The ARN of the role to assume."
    echo ""
  }

  while getopts n:r:h option; do
    case "${option}" in
      n) role_session_name=${OPTARG} ;;
      r) role_arn=${OPTARG} ;;
      h)
        usage
        return 0
        ;;
      \?)
        ech o"Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  response=$(aws sts assume-role \
    --role-session-name "$role_session_name" \
    --role-arn "$role_arn" \
    --output text \
    --query "Credentials.[AccessKeyId, SecretAccessKey, SessionToken]")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
```

```
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- API 세부 정보는 AWS CLI명령 참조의 AssumeRole을 참조하십시오.

C++

SDK for C++

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
                             const Aws::String &roleSessionName,
                             const Aws::String &externalId,
                             Aws::Auth::AWSCredentials &credentials,
                             const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
 result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 AssumeRole을 참조하십시오.

CLI

AWS CLI

역할 위임

다음 assume-role 명령은 IAM 역할 s3-access-example에 대한 일련의 단기 보안 인증 정보를 검색합니다.

```
aws sts assume-role \
    --role-arn arn:aws:iam::123456789012:role/xaccounts3access \
    --role-session-name s3-access-example
```

출력:

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:s3-access-example",
        "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-
access-example"
    },
    "Credentials": {
```

```
        "SecretAccessKey": "9drTJvcXLB89EXAMPLELB8923FB892xMFI",
        "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/
qwjzP2iEXAMPLEbw/m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8B
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=",
        "Expiration": "2016-03-15T00:05:07Z",
        "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
    }
}
```

명령의 출력에는 AWS 인증에 사용할 수 있는 액세스 키, 비밀 키 및 세션 토큰이 포함됩니다.

AWS CLI를 사용하는 경우 역할과 연결된 이름이 지정된 프로파일을 설정할 수 있습니다. 프로파일을 사용하면 AWS CLI에서 assume-role을 호출하고 대신 보안 인증 정보를 관리합니다. 자세한 내용은 AWS CLI 사용 설명서의 AWS CLI에서 IAM 역할 사용을 참조하세요.

- API 세부 정보는 AWS CLI명령 참조의 AssumeRole을 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;
```

```
/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 * "Version": "2012-10-17",
 * "Statement": [
 * {
 * "Effect": "Allow",
 * "Principal": {
 * "AWS": "<Specify the ARN of your IAM user you are using in this code
 * example>"
 * },
 * "Action": "sts:AssumeRole"
 * }
 * ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleArn> <roleSessionName>\s

                Where:
                    roleArn - The Amazon Resource Name (ARN) of the role to
 assume (for example, rn:aws:iam::000008047983:role/s3role).\s
                    roleSessionName - An identifier for the assumed role session
 (for example, mysession).\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
                .region(region)
                .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn,
String roleSessionName) {
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();

            // Display the time when the temp creds expire.
            Instant exTime = myCreds.expiration();
            String tokenInfo = myCreds.sessionToken();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(exTime);
            System.out.println("The token " + tokenInfo + "  expires on " +
exTime);

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AssumeRole을 참조하십시오.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

클라이언트를 생성합니다.

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an AWS STS service client object.
export const client = new STSClient({ region: REGION });
```

IAM 역할을 수임합니다.

```
import { AssumeRoleCommand } from "@aws-sdk/client-sts";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Returns a set of temporary security credentials that you can use to
    // access Amazon Web Services resources that you might not normally
    // have access to.
    const command = new AssumeRoleCommand({
      // The Amazon Resource Name (ARN) of the role to assume.
      RoleArn: "ROLE_ARN",
      // An identifier for the assumed role session.
```

```
      RoleSessionName: "session1",
      // The duration, in seconds, of the role session. The value specified
      // can range from 900 seconds (15 minutes) up to the maximum session
      // duration set for the role.
      DurationSeconds: 900,
    });
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 AssumeRole을 참조하십시오.

SDK for JavaScript (v2)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Load the AWS SDK for Node.js
const AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

var roleToAssume = {
  RoleArn: "arn:aws:iam::123456789012:role/RoleName",
  RoleSessionName: "session1",
  DurationSeconds: 900,
};
var roleCreds;

// Create the STS service object
var sts = new AWS.STS({ apiVersion: "2011-06-15" });

//Assume Role
sts.assumeRole(roleToAssume, function (err, data) {
  if (err) console.log(err, err.stack);
  else {
```

```
    roleCreds = {
      accessKeyId: data.Credentials.AccessKeyId,
      secretAccessKey: data.Credentials.SecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    stsGetCallerIdentity(roleCreds);
  }
});

//Get Arn of current identity
function stsGetCallerIdentity(creds) {
  var stsParams = { credentials: creds };
  // Create STS service object
  var sts = new AWS.STS(stsParams);

  sts.getCallerIdentity({}, function (err, data) {
    if (err) {
      console.log(err, err.stack);
    } else {
      console.log(data.Arn);
    }
  });
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 AssumeRole을 참조하십시오.

PowerShell

PowerShell용 도구

요청하는 사용자가 일반적으로 액세스할 수 없는 AWS 리소스에 액세스하는 데 1시간 동안 사용할 수 있는 임시 자격 증명(액세스 키, 비밀 키 및 세션 토큰) 세트를 반환합니다. 반환된 자격 증명에는 수임 중인 역할의 액세스 정책과 제공된 정책에 의해 허용되는 권한이 있습니다. 제공된 정책을 사용하여 수임 중인 역할의 액세스 정책에 의해 정의된 권한을 초과하는 권한을 부여할 수 없습니다.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -Policy "...JSON policy..." -DurationInSeconds 3600
```

예제 2: 수임된 역할의 액세스 정책에 정의된 것과 동일한 권한을 갖고 1시간 동안 유효한 임시 자격 증명 세트를 반환합니다.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600
```

예제 3: cmdlet을 실행하는 데 사용되는 사용자 자격 증명과 연결된 MFA에서 생성된 토큰과 일련 번호를 제공하는 임시 자격 증명 세트를 반환합니다.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

예제 4: 고객 계정에 정의된 역할을 수임한 임시 자격 증명 세트를 반환합니다. 타사에서 수임할 수 있는 각 역할에 대해 고객 계정은 역할이 수임될 때마다 -ExternalID 파라미터로 전달되는 식별자를 사용하여 역할을 생성해야 합니다.

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -ExternalId "ABC123"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 AssumeRole을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

MFA 토큰이 필요한 IAM 역할을 수임하고 임시 자격 증명을 사용하여 계정에 대한 Amazon S3 버킷을 나열합니다.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
```

```
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 AssumeRole를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: "create-use-assume-role-scenario"
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 AssumeRole을 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn assume_role(config: &SdkConfig, role_name: String, session_name:
 Option<String>) {
    let provider = aws_config::sts::AssumeRoleProvider::builder(role_name)
        .session_name(session_name.unwrap_or("rust_sdk_example_session".into()))
        .configure(config)
        .build()
        .await;

    let local_config = aws_config::from_env()
        .credentials_provider(provider)
        .load()
        .await;
    let client = Client::new(&local_config);
    let req = client.get_caller_identity();
    let resp = req.send().await;
    match resp {
        Ok(e) => {
            println!("UserID :               {}",
 e.user_id().unwrap_or_default());
            println!("Account:               {}",
 e.account().unwrap_or_default());
            println!("Arn    :               {}", e.arn().unwrap_or_default());
        }
        Err(e) => println!("{:?}", e),
    }
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 AssumeRole을 참조하십시오.

Swift

SDK for Swift

> **Note**
>
> 이 사전 릴리스 설명서는 평가판 버전 SDK에 관한 것입니다. 내용은 변경될 수 있습니다.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public func assumeRole(role: IAMClientTypes.Role, sessionName: String)
                async throws -> STSClientTypes.Credentials {
    let input = AssumeRoleInput(
        roleArn: role.arn,
        roleSessionName: sessionName
    )
    do {
        let output = try await stsClient.assumeRole(input: input)

        guard let credentials = output.credentials else {
            throw ServiceHandlerError.authError
        }

        return credentials
    } catch {
        throw error
    }
}
```

- API 세부 정보는 Swift용 AWS SDK API 참조의 AssumeRole을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **AssumeRoleWithWebIdentity** 사용

다음 코드 예제는 AssumeRoleWithWebIdentity의 사용 방법을 보여 줍니다.

CLI

AWS CLI

웹 자격 증명으로 인증된 역할에 대한 단기 자격 증명 받기(OAuth 2."0)

다음 assume-role-with-web-identity 명령은 IAM 역할 app1에 대한 일련의 단기 보안 인증 정보를 검색합니다. 요청은 지정된 웹 ID 제공업체가 제공하는 웹 ID 토큰을 사용하여 인증됩니다. 사용자가 수행할 수 있는 작업을 추가로 제한하기 위해 두 가지 추가 정책이 세션에 적용됩니다. 반환된 자격 증명은 생성되고 1시간 후에 만료됩니다.

```
aws sts assume-role-with-web-identity \
    --duration-seconds 3600 \
    --role-session-name "app1" \
    --provider-id "www.amazon.com" \
    --policy-arns "arn:aws:iam::123456789012:policy/
q=webidentitydemopolicy1","arn:aws:iam::123456789012:policy/
webidentitydemopolicy2" \
    --role-arn arn:aws:iam::123456789012:role/FederatedWebIdentityRole \
    --web-identity-token "Atza
%7CIQEBLjAsAhRFiXuWpUXuRvQ9PZL3GMFcYevydwIUFAHZwXZXXXXXXXXJnrulxKDHwy87oGKPznh0D6bEQZTSCz
CrKqjG7nPBjNIL016GGvuS5gSvPRUxWES3VYfm1wl7WTI7jn-Pcb6M-
buCgHhFOzTQxod27L9CqnOLio7N3gZAGpsp6n1-
AJBOCJckcyXe2c6uD0srOJeZlKUm2eTDVMf8IehDVI0r1QOnTV6KzzAI3OY87Vd_cVMQ"
```

출력:

```
{
    "SubjectFromWebIdentityToken": "amzn1.account.AF6RHO7KZU5XRVQJGXK6HB56KR2A"
    "Audience": "client.5498841531868486423.1548@apps.example.com",
    "AssumedRoleUser": {
        "Arn": "arn:aws:sts::123456789012:assumed-role/FederatedWebIdentityRole/
app1",
        "AssumedRoleId": "AROACLKWSDQRAOEXAMPLE:app1"
    }
    "Credentials": {
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
```

```
        "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2020-05-19T18:06:10+00:00"
    },
    "Provider": "www.amazon.com"
}
```

자세한 내용은 AWS IAM 사용 설명서의 임시 보안 자격 증명 요청을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 AssumeRoleWithWebIdentity를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: Login with Amazon ID 제공업체를 통해 인증된 사용자에 대해 1시간 동안 유효한 임시 자격 증명 세트를 반환합니다. 자격 증명은 역할 ARN으로 식별된 역할과 연결된 액세스 정책을 수임합니다. 필요에 따라 액세스 권한을 더욱 세분화하는 -Policy 파라미터에 JSON 정책을 전달할 수 있습니다. 역할과 연결된 권한에서 사용 가능한 것보다 더 많은 권한을 부여할 수는 없습니다. -WebIdentityToken에 제공되는 값은 ID 제공업체가 반환한 고유한 사용자 식별자입니다.

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 AssumeRoleWithWebIdentity를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **DecodeAuthorizationMessage** 사용

다음 코드 예제는 DecodeAuthorizationMessage의 사용 방법을 보여 줍니다.

CLI

AWS CLI

요청에 대한 응답으로 반환된 인코딩된 인증 메시지 디코딩

다음 decode-authorization-message 예제는 Amazon Web Services 요청에 대한 응답으로 반환되는 인코딩 메시지로부터 받은 요청의 권한 부여 상태에 대한 추가 정보를 디코딩합니다.

```
aws sts decode-authorization-message \
    --encoded-message EXAMPLEWodyRNrtlQARDip-
eTA6i6DrlUhHhPQrLWB_lAbl5pAKxl9mPDLexYcGBreyIKQC1BGBIpBKr3dFDkwqeO7e2NMk5j_hmzAiChJN-8oy3
Ojau7BMjOTWwOtHPhV_Zaz87yENdipr745EjQwRd5LaoL3vN8_5ZfA9UiBMKDgVh1gjqZJFUiQoubv78V1RbHNYnK
pOu3FZjwYStfvTb3GHs3-6rLribGO9jZOktkfE6vqxlFzLyeDr4P2ihC1wty9tArCvvGzIAUNmARQJ2VVWPxioqgo
JWP5pwe_mAyqh0NLw-r1S56YC_90onj9A80sNrHlI-
tIiNd7tgNTYzDuPQYD2FMDBnp82V9eVmYGtPp5NIeSpuf3fOHanFuBZgENxZQZ2dlH3xJGMTtYayzZrRXjiq_SfX9
FaoPIb8LmmKVBLpIB0iFhU9sEHPqKHVPi6jdxXqKaZaFGvYVmVOiuQdNQKuyk0p067POFrZECLjjOtNPBOZCcuEKE
```

출력:

```
{
    "DecodedMessage": "{\"allowed\":false,\"explicitDeny\":true,
\"matchedStatements\":{\"items\":[{\"statementId\":\"VisualEditor0\",\"effect
\":\"DENY\",\"principals\":{\"items\":[{\"value\":\"AROA123456789EXAMPLE
\"}]},\"principalGroups\":{\"items\":[]},\"actions\":{\"items\":[{\"value
\":\"ec2:RunInstances\"}]},\"resources\":{\"items\":[{\"value\":\"*
\"}]},\"conditions\":{\"items\":[]}}]},\"failures\":{\"items\":[]},
\"context\":{\"principal\":{\"id\":\"AROA123456789EXAMPLE:Ana\",\"arn
\":\"arn:aws:sts::111122223333:assumed-role/Developer/Ana\"},\"action\":
\"RunInstances\",\"resource\":\"arn:aws:ec2:us-east-1:111122223333:instance/*
\",\"conditions\":{\"items\":[{\"key\":\"ec2:MetadataHttpPutResponseHopLimit\",
\"values\":{\"items\":[{\"value\":\"2\"}]}},{\"key\":\"ec2:InstanceMarketType
\",\"values\":{\"items\":[{\"value\":\"on-demand\"}]}},{\"key\":\"aws:Resource
\",\"values\":{\"items\":[{\"value\":\"instance/*\"}]}},{\"key\":\"aws:Account
\",\"values\":{\"items\":[{\"value\":\"111122223333\"}]}},{\"key\":
\"ec2:AvailabilityZone\",\"values\":{\"items\":[{\"value\":\"us-east-1f\"}]}},
{\"key\":\"ec2:ebsOptimized\",\"values\":{\"items\":[{\"value\":\"false\"}]}},
{\"key\":\"ec2:IsLaunchTemplateResource\",\"values\":{\"items\":[{\"value\":
\"false\"}]}},{\"key\":\"ec2:InstanceType\",\"values\":{\"items\":[{\"value
\":\"t2.micro\"}]}},{\"key\":\"ec2:RootDeviceType\",\"values\":{\"items\":
[{\"value\":\"ebs\"}]}},{\"key\":\"aws:Region\",\"values\":{\"items\":[{\"value
\":\"us-east-1\"}]}},{\"key\":\"ec2:MetadataHttpEndpoint\",\"values\":{\"items
```

```
\":[{\"value\":\"enabled\"}]}},{\"key\":\"aws:Service\",\"values\":{\"items
\":[{\"value\":\"ec2\"}]}},{\"key\":\"ec2:InstanceID\",\"values\":{\"items\":
[{\"value\":\"*\"}]}},{\"key\":\"ec2:MetadataHttpTokens\",\"values\":{\"items
\":[{\"value\":\"required\"}]}},{\"key\":\"aws:Type\",\"values\":{\"items
\":[{\"value\":\"instance\"}]}},{\"key\":\"ec2:Tenancy\",\"values\":{\"items
\":[{\"value\":\"default\"}]}},{\"key\":\"ec2:Region\",\"values\":{\"items
\":[{\"value\":\"us-east-1\"}]}},{\"key\":\"aws:ARN\",\"values\":{\"items\":
[{\"value\":\"arn:aws:ec2:us-east-1:111122223333:instance/*\"}]}}]}}}"
}
```

자세한 내용은 AWS IAM 사용 설명서의 정책 평가 로직을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 DecodeAuthorizationMessage를 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 요청에 대한 응답으로 반환된 제공된 인코딩된 메시지 내용에 포함된 추가 정보를 디코딩합니다. 권한 부여 상태의 세부 정보가 작업을 요청한 사용자가 볼 수 없는 권한 있는 정보로 구성될 수 있기 때문에 추가 정보가 인코딩됩니다.

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 DecodeAuthorizationMessage를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetFederationToken** 사용

다음 코드 예제는 GetFederationToken의 사용 방법을 보여 줍니다.

CLI

AWS CLI

IAM 사용자 액세스 키 자격 증명을 사용하여 임시 보안 자격 증명 세트 반환

다음 get-federation-token 예제는 사용자에 대한 임시 보안 자격 증명 세트(액세스 키 ID, 비밀 액세스 키 및 보안 토큰으로 구성)를 반환합니다. IAM 사용자의 장기 보안 자격 증명을 사용하여 GetFederationToken 작업을 직접적으로 호출해야 합니다.

```
aws sts get-federation-token \
    --name Bob \
    --policy file://myfile.json \
    --policy-arns arn=arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess \
    --duration-seconds 900
```

myfile.json의 콘텐츠:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:Describe*",
            "Resource": "*"
        }
    ]
}
```

출력:

```
{
    "Credentials": {
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "SessionToken": "EXAMPLEpZ2luX2VjEGoaCXVzLXdlc3QtMiJIMEYCIQC/
W9pL5ArQyDD5JwFL3/h5+WGopQ24GEXweNctwhi9sgIhAMkg
+MZE35iWM8s4r5Lr25f9rSTVPFH98G42QQunWMTfKq0DCOP/////////
wEQAxoMNDUyOTI1MTcwNTA3Igxuy3AOpuuoLsk3MJwqgQPg8QOd9HuoClUxq26wnc/nm
+eZLjHDyGf2KUAHK2DuaS/nrGSEXAMPLE",
        "Expiration": "2023-12-20T02:06:07+00:00"
    },
    "FederatedUser": {
        "FederatedUserId": "111122223333:Bob",
        "Arn": "arn:aws:sts::111122223333:federated-user/Bob"
    },
    "PackedPolicySize": 36
}
```

자세한 내용은 AWS IAM 사용 설명서의 임시 보안 자격 증명 요청을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetFederationToken을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 페더레이션 사용자의 이름으로 'Bob'을 사용하여 1시간 동안 유효한 페더레이션 토큰을 요청합니다. 이 이름은 리소스 기반 정책(예: Amazon S3 버킷 정책)에서 페더레이션 사용자 이름을 참조하는 데 사용할 수 있습니다. JSON 형식으로 제공된 IAM 정책은 IAM 사용자가 사용할 수 있는 권한의 범위를 좁히는 데 사용됩니다. 제공된 정책은 요청하는 사용자에게 부여된 것보다 더 많은 권한을 부여할 수 없으며, 페더레이션 사용자에 대한 최종 권한은 전달된 정책과 IAM 사용자 정책의 교차점을 기준으로 가장 제한적인 세트입니다.

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- API 세부 정보는 AWS Tools for PowerShell 명령 참조의 GetFederationToken을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 함께 **GetSessionToken** 사용

다음 코드 예제는 GetSessionToken의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- MFA 토큰이 필요한 세션 토큰 가져오기

CLI

AWS CLI

IAM 자격 증명용 단기 보안 인증 정보 세트 가져오기

다음 get-session-token 명령은 호출을 위한 IAM 자격 증명용 단기 보안 인증 정보 세트를 검색합니다. 정책에 따라 다중 인증(MFA)이 필요한 경우 요청에 이 보안 인증 정보를 사용할 수 있습니다. 보안 인증 정보는 생성되고 15분 후에 만료됩니다.

```
aws sts get-session-token \
    --duration-seconds 900 \
    --serial-number "YourMFADeviceSerialNumber" \
    --token-code 123456
```

출력:

```
{
    "Credentials": {
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2020-05-19T18:06:10+00:00"
    }
}
```

자세한 내용은 AWS IAM 사용 설명서의 임시 보안 자격 증명 요청을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 GetSessionToken을 참조하세요.

PowerShell

PowerShell용 도구

예제 1: 설정된 기간 동안 유효한 임시 자격 증명이 포함된 **Amazon.RuntimeAWSCredentials** 인스턴스를 반환합니다. 임시 자격 증명을 요청하는 데 사용되는 자격 증명은 현재 쉘 기본값에서 유추됩니다. 다른 자격 증명을 지정하려면 -ProfileName 또는 -AccessKey/-SecretKey 파라미터를 사용합니다.

```
Get-STSSessionToken
```

출력:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

예제 2: 1시간 동안 유효한 임시 자격 증명이 포함된 **Amazon.RuntimeAWSCredentials** 인스턴스를 반환합니다. 요청에 사용되는 자격 증명은 지정된 프로파일에서 가져옵니다.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

출력:

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

예제 3: 프로파일 'myprofilename'에 자격 증명이 지정된 계정과 연결된 MFA 디바이스의 식별 번호와 디바이스에서 제공한 값을 사용하여 1시간 동안 유효한 임시 자격 증명이 들어 있는 **Amazon.RuntimeAWSCredentials** 인스턴스를 반환합니다.

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

출력:

```
AccessKeyId                               Expiration
 SecretAccessKey                          SessionToken
-----------                               ----------
 ---------------                          ------------
EXAMPLEACCESSKEYID                        2/16/2015 9:12:28 PM
 examplesecretaccesskey...                SamPleTokeN.....
```

- API 세부 정보는 AWS Tools for PowerShell Cmdlet 참조의 GetSessionToken을 참조하세요.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

MFA 토큰을 전달하여 세션 토큰을 가져와 계정에 대한 Amazon S3 버킷을 나열하는 데 사용합니다.

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
```

```
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- API 세부 정보는 Python용 AWS SDK(Boto3) API 참조의 GetSessionToken을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하는 AWS STS 시나리오

다음 코드 예제는 AWS SDK를 사용하여 AWS STS에서 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이러한 시나리오에서는 AWS STS 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접 호출하여 특정 태스크를 수행하는 방법을 보여줍니다. 각 시나리오에는 전체 소스 코드에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시나리오는 컨텍스트에 맞는 서비스 작업을 이해하는 데 도움이 되도록 중급 수준의 경험을 대상으로 합니다.

예시

- AWS SDK를 사용하여 AWS STS에서 MFA 토큰이 필요한 IAM 역할 수임
- AWS SDK를 사용하여 페더레이션 사용자를 위해 AWS STS로 URL 구성

- AWS SDK를 사용하여 AWS STS에서 MFA 토큰이 필요한 세션 토큰 가져오기

AWS SDK를 사용하여 AWS STS에서 MFA 토큰이 필요한 IAM 역할 수임

다음 코드 예제에서는 MFA 토큰이 필요한 역할을 수임하는 방법을 보여줍니다.

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 자격 증명 공급자를 통한 페더레이션을 사용하세요.

- Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 IAM 역할을 생성합니다.
- MFA 보안 인증이 제공된 경우에만 역할을 수임할 권한이 있는 IAM 사용자를 생성합니다.
- 사용자를 위한 MFA 디바이스를 등록합니다.
- 역할을 수임하고 임시 보안 인증을 사용하여 S3 버킷을 나열합니다.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

IAM 사용자를 생성하고, MFA 디바이스를 등록하고, S3 버킷을 나열할 수 있는 권한을 부여하는 역할을 생성합니다. 사용자는 역할을 수임할 수 있는 권한만 있습니다.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual MFA device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
```

```
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role and requires
MFA.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    For demonstration purposes, the user is created in the same account as the
role,
    but in practice the user would likely be from another account.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
resource
                         that has permissions to create users, roles, and
policies
                         in the account.
    :return: The newly created user, user key, virtual MFA device, and role.
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name("mfa")
    )
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
```

```
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": user.arn},
                        "Action": "sts:AssumeRole",
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent":
True}},
                    }
                ],
            }
        ),
    )
    print(f"Created role {role.name} that requires MFA.")

    policy = iam_resource.create_policy(
        PolicyName=unique_name("policy"),
        PolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                    }
                ],
            }
        ),
```

```
    )
    role.attach_policy(PolicyArn=policy.arn)
    print(f"Created policy {policy.policy_name} and attached it to the role.")

    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "sts:AssumeRole",
                        "Resource": role.arn,
                    }
                ],
            }
        ),
    )
    print(
        f"Created an inline policy for {user.name} that lets the user assume "
        f"the role."
    )

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, virtual_mfa_device, role
```

MFA 토큰 없이 역할을 수임하는 것은 허용되지 않음을 보여줍니다.

```
def try_to_assume_role_without_mfa(assume_role_arn, session_name, sts_client):
    """
    Shows that attempting to assume the role without sending MFA credentials
 results
    in an AccessDenied error.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role to assume.
    :param session_name: The name of the STS session.
```

```
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    print(f"Trying to assume the role without sending MFA credentials...")
    try:
        sts_client.assume_role(RoleArn=assume_role_arn,
 RoleSessionName=session_name)
        raise RuntimeError("Expected AccessDenied error.")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Got AccessDenied.")
        else:
            raise
```

S3 버킷을 나열할 수 있는 권한을 부여하는 역할을 수임하여 필요한 MFA 토큰을 전달하고 버킷을 나열할 수 있음을 보여줍니다.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
 account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
```

```
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

데모용으로 생성된 리소스를 삭제합니다.

```
def teardown(user, virtual_mfa_device, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        policy_name = attached.policy_name
        role.detach_policy(PolicyArn=attached.arn)
        attached.delete()
        print(f"Detached and deleted {policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
```

```
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

이전에 정의한 함수를 사용하여 이 시나리오를 실행합니다.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device, role = setup(iam_resource)
    print(f"Created {user.name} and {role.name}.")
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
        )
        try_to_assume_role_without_mfa(role.arn, "demo-sts-session", sts_client)
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_from_assumed_role_with_mfa(
            role.arn,
            "demo-sts-session",
            virtual_mfa_device.serial_number,
            mfa_totp,
            sts_client,
        )
    finally:
        teardown(user, virtual_mfa_device, role)
        print("Thanks for watching!")
```

- API 세부 정보는 Python용 AWS SDK(Boto3) API 참조의 AssumeRole을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 페더레이션 사용자를 위해 AWS STS로 URL 구성

다음 코드 예시는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 현재 계정의 Amazon S3 리소스에 대한 읽기 전용 액세스 권한을 부여하는 IAM 역할을 생성합니다.
- AWS 페더레이션 엔드포인트에서 보안 토큰을 받습니다.
- 페더레이션형 보안 인증으로 콘솔에 액세스하는 데 사용할 수 있는 URL을 구성합니다.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

현재 계정의 S3 리소스에 대한 읽기 전용 액세스 권한을 부여하는 역할을 생성합니다.

```
def setup(iam_resource):
    """
    Creates a role that can be assumed by the current user.
    Attaches a policy that allows only Amazon S3 read-only access.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 instance
                         that has the permission to create a role.
    :return: The newly created role.
    """
    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
```

```
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": iam_resource.CurrentUser().arn},
                        "Action": "sts:AssumeRole",
                    }
                ],
            }
        ),
    )
    role.attach_policy(PolicyArn="arn:aws:iam::aws:policy/
AmazonS3ReadOnlyAccess")
    print(f"Created role {role.name}.")

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return role
```

AWS 페더레이션 엔드포인트에서 보안 토큰을 받고 페더레이션형 자격 증명으로 콘솔에 액세스하는 데 사용할 수 있는 URL을 구성합니다.

```
def construct_federated_url(assume_role_arn, session_name, issuer, sts_client):
    """
    Constructs a URL that gives federated users direct access to the AWS
 Management
    Console.

    1. Acquires temporary credentials from AWS Security Token Service (AWS STS)
 that
       can be used to assume a role with limited permissions.
    2. Uses the temporary credentials to request a sign-in token from the
       AWS federation endpoint.
    3. Builds a URL that can be used in a browser to navigate to the AWS
 federation
       endpoint, includes the sign-in token for authentication, and redirects to
       the AWS Management Console with permissions defined by the role that was
       specified in step 1.
```

```
    :param assume_role_arn: The role that specifies the permissions that are
granted.
                            The current user must have permission to assume the
role.
    :param session_name: The name for the STS session.
    :param issuer: The organization that issues the URL.
    :param sts_client: A Boto3 STS instance that can assume the role.
    :return: The federated URL.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn, RoleSessionName=session_name
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    session_data = {
        "sessionId": temp_credentials["AccessKeyId"],
        "sessionKey": temp_credentials["SecretAccessKey"],
        "sessionToken": temp_credentials["SessionToken"],
    }
    aws_federated_signin_endpoint = "https://signin.aws.amazon.com/federation"

    # Make a request to the AWS federation endpoint to get a sign-in token.
    # The requests.get function URL-encodes the parameters and builds the query
string
    # before making the request.
    response = requests.get(
        aws_federated_signin_endpoint,
        params={
            "Action": "getSigninToken",
            "SessionDuration": str(datetime.timedelta(hours=12).seconds),
            "Session": json.dumps(session_data),
        },
    )
    signin_token = json.loads(response.text)
    print(f"Got a sign-in token from the AWS sign-in federation endpoint.")

    # Make a federated URL that can be used to sign into the AWS Management
Console.
    query_string = urllib.parse.urlencode(
        {
            "Action": "login",
            "Issuer": issuer,
            "Destination": "https://console.aws.amazon.com/",
```

```
            "SigninToken": signin_token["SigninToken"],
        }
    )
    federated_url = f"{aws_federated_signin_endpoint}?{query_string}"
    return federated_url
```

데모용으로 생성된 리소스를 삭제합니다.

```
def teardown(role):
    """
    Removes all resources created during setup.

    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        role.detach_policy(PolicyArn=attached.arn)
        print(f"Detached {attached.policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
```

이전에 정의한 함수를 사용하여 이 시나리오를 실행합니다.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the AWS Security Token Service federated URL demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    role = setup(iam_resource)
    sts_client = boto3.client("sts")
    try:
        federated_url = construct_federated_url(
            role.arn, "AssumeRoleDemoSession", "example.org", sts_client
        )
        print(
            "Constructed a federated URL that can be used to connect to the "
            "AWS Management Console with role-defined permissions:"
```

```
        )
        print("-" * 88)
        print(federated_url)
        print("-" * 88)
        _ = input(
            "Copy and paste the above URL into a browser to open the AWS "
            "Management Console with limited permissions. When done, press "
            "Enter to clean up and complete this demo."
        )
    finally:
        teardown(role)
        print("Thanks for watching!")
```

- API 세부 정보는 Python용 AWS SDK(Boto3) API 참조의 AssumeRole을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 AWS STS에서 MFA 토큰이 필요한 세션 토큰 가져오기

다음 코드 예제에서는 MFA 토큰이 필요한 세션 토큰을 가져오는 방법을 보여줍니다.

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 자격 증명 공급자를 통한 페더레이션을 사용하세요.

- Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 IAM 역할을 생성합니다.
- MFA 보안 인증이 제공된 경우에만 역할을 수임할 권한이 있는 IAM 사용자를 생성합니다.
- 사용자를 위한 MFA 디바이스를 등록합니다.
- MFA 보안 인증을 제공하여 세션 토큰을 가져오고 임시 보안 인증을 사용하여 S3 버킷을 나열합니다.

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보십시오.

IAM 사용자를 생성하고, MFA 디바이스를 등록하고, MFA 보안 인증이 사용될 때만 사용자가 S3 버킷을 나열할 수 있는 권한을 부여하는 역할을 생성합니다.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual multi-factor authentication (MFA) device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates an inline policy for the user that lets the user list Amazon S3
 buckets,
    but only when MFA credentials are used.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 resource
                         that has permissions to create users, MFA devices, and
                         policies in the account.
    :return: The newly created user, user key, and virtual MFA device.
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name("mfa")
    )
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")
```

```
    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent":
True}},
                    }
                ],
            }
        ),
    )
    print(
```

```
        f"Created an inline policy for {user.name} that lets the user list
 buckets, "
        f"but only when MFA credentials are present."
    )

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, virtual_mfa_device
```

MFA 토큰을 전달하여 임시 세션 보안 인증을 가져오고 보안 인증을 사용하여 계정에 대한 S3 버킷을 나열합니다.

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
```

```
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

데모용으로 생성된 리소스를 삭제합니다.

```
def teardown(user, virtual_mfa_device):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo MFA device.
    """
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

이전에 정의한 함수를 사용하여 이 시나리오를 실행합니다.

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
```

```
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device = setup(iam_resource)
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
        )
        try:
            print("Listing buckets without specifying MFA credentials.")
            list_buckets_with_session_token_with_mfa(None, None, sts_client)
        except ClientError as error:
            if error.response["Error"]["Code"] == "AccessDenied":
                print("Got expected AccessDenied error.")
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_with_session_token_with_mfa(
            virtual_mfa_device.serial_number, mfa_totp, sts_client
        )
    finally:
        teardown(user, virtual_mfa_device)
        print("Thanks for watching!")
```

- API 세부 정보는 AWS SDK for Python(Boto3) API 참조의 GetSessionToken을 참조하세요.

AWS SDK 개발자 가이드 및 코드 예시의 전체 목록은 AWS SDK와 함께 이 서비스 사용 단원을 참조하세요. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

IAM 및 AWS STS의 보안

AWS에서 클라우드 보안은 가장 중요합니다. AWS 고객은 보안에 가장 보안에 민감한 조직의 요구 사항에 부합하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS와 사용자의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드의 보안: AWS는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS는 안전하게 사용할 수 있는 서비스 또한 제공합니다. 서드 파티 감사자는 AWS 규정 준수 프로그램의 일환으로 보안 효과를 정기적으로 테스트하고 검증합니다. AWS Identity and Access Management(IAM)에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램의 범위에 속하는 AWS 서비스를 참조하세요.
- 클라우드 내 보안 – 귀하의 책임은 귀하가 사용하는 AWS 서비스로 결정됩니다. 또한 귀하는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 AWS Identity and Access Management(IAM) 및 AWS Security Token Service(AWS STS)를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목적에 맞게 IAM 및 AWS STS를 구성하는 방법을 보여줍니다. 또한 IAM 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

내용

AWS 보안 인증 정보

AWS와(과) 상호 작용할 경우 AWS 보안 인증 정보를 지정하여 사용자가 누구인지, 요청 중인 리소스에 액세스할 수 있는 권한이 있는지를 확인합니다. AWS에서는 보안 인증 정보를 사용하여 요청을 인증하고 권한을 부여합니다.

예를 들어 Amazon Simple Storage Service(S3) 버킷에서 보호된 파일을 다운로드하려면 보안 인증에서 해당 액세스를 허용해야 합니다. 보안 인증에서 파일 다운로드 권한이 없는 것으로 표시되는 경우 AWS는 요청을 거부합니다. 하지만 공개적으로 공유되는 Amazon S3 버킷에서 파일을 다운로드하는 데에는 AWS 보안 인증이 요구되지 않습니다.

AWS에는 각각 고유한 보안 자격 증명을 가진 다양한 유형의 사용자가 있습니다.

- 계정 소유자(루트 사용자) - AWS 계정을 생성하고 전체 액세스 권한을 갖는 사용자입니다.
- AWS IAM Identity Center 사용자 - AWS IAM Identity Center에서 관리되는 사용자입니다.
- 페더레이션 사용자 - 페더레이션을 통해 AWS에 대한 임시 액세스 권한이 부여된 외부 ID 제공업체의 사용자입니다. 페더레이션 자격 증명에 대한 자세한 내용은 자격 증명 공급자 및 페더레이션 섹션을 참조하세요.
- IAM 사용자 - AWS Identity and Access Management(IAM) 서비스 내에서 생성된 개별 사용자입니다.

사용자는 장기 또는 임시 보안 인증 정보를 가지고 있습니다. 루트 사용자, IAM 사용자 및 액세스 키에는 만료되지 않는 장기 보안 인증 정보가 있습니다. 장기 보안 인증 정보를 보호하기 위해 액세스 키를 관리하고, 암호를 변경하며, MFA를 활성화하는 프로세스를 마련합니다. 자세한 내용은 AWS Identity and Access Management의 보안 모범 사례 및 사용 사례 단원을 참조하십시오.

IAM 역할, AWS IAM Identity Center의 사용자 또는 페더레이션 사용자에게는 임시 보안 인증 정보가 있습니다. 임시 보안 인증 정보는 지정된 기간이 지난 후 또는 사용자가 세션을 종료할 때 만료됩니다. 임시 보안 인증은 다음과 같은 차이점을 제외하고는 장기 보안 인증과 거의 동일한 효력을 지닙니다.

- 임시 보안 자격 증명은 그 이름이 암시하듯 단기적입니다. 이 자격 증명은 몇 분에서 몇 시간까지 지속되도록 구성할 수 있습니다. 자격 증명이 만료된 후 AWS는 더는 그 자격 증명을 인식하지 못하거나 그 자격 증명을 사용한 API 요청으로부터 이루어지는 어떤 종류의 액세스도 허용하지 않습니다.
- 임시 보안 자격 증명은 사용자와 함께 저장되지 않지만 동적으로 생성되어 요청시 사용자에게 제공됩니다. 임시 보안 자격 증명이 만료되었을 때(심지어는 만료 전이라도) 사용자는 새 자격 증명을 요청할 수 있습니다. 단, 자격 증명을 요청하는 해당 사용자에게 그렇게 할 수 있는 권한이 있어야 합니다.

따라서 임시 보안 인증은 장기 보안 인증보다 다음과 같은 이점이 있습니다.

- 애플리케이션으로 장기 AWS 보안 자격 증명을 배포 또는 포함할 필요가 없습니다.
- 사용자에 대한 AWS 자격 증명을 정의하지 않고도 AWS 리소스에 대한 액세스 권한을 사용자에게 제공할 수 있습니다. 임시 자격 증명은 역할 및 아이덴티티 페더레이션을 위한 기초입니다.
- 임시 보안 인증은 수명이 제한되어 있어서, 더 이상 필요하지 않을 때 업데이트하거나 명시적으로 취소할 필요가 없습니다. 임시 보안 자격 증명이 만료된 후에는 다시 사용할 수 없습니다. 그 자격 증명에 대해 유효 기간을 최대 한계까지 지정할 수 있습니다.

보안 고려 사항

AWS 계정의 보안 조건을 결정할 때 다음 정보를 고려하는 것이 좋습니다.

- AWS 계정을 만들면 계정 루트 사용자가 생성됩니다. 루트 사용자(계정 소유자)의 보안 인증 정보는 계정 내 모든 리소스에 대한 모든 액세스 권한을 허용합니다. 루트 사용자를 사용하여 수행할 첫 번째 작업은 다른 사용자에게 AWS 계정 관리 권한을 부여하여 루트 사용자의 사용을 최소화하는 것입니다.
- 다중 인증(MFA)은 AWS 계정에 액세스할 수 있는 사용자의 보안 수준을 강화합니다. 보안 강화를 위해 AWS 계정 루트 사용자 보안 인증 및 모든 IAM 사용자에 대해 MFA를 요구하는 것이 좋습니다. 자세한 내용은 IAM의 AWS 다중 인증 단원을 참조하십시오.
- AWS에 액세스하는 방식과 AWS 사용자 유형에 따라, AWS는 다양한 유형의 보안 인증을 요구합니다. 예를 들어 AWS Management Console에는 로그인 보안 인증 정보를 사용하는 반면, AWS를 프로그래밍 방식으로 호출하는 데에는 액세스 키를 사용합니다. 사용자 유형 및 로그인 페이지를 결정하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 AWS 로그인이란 무엇입니까?를 참조하세요.
- IAM 정책을 사용하여 리소스에 대한 루트 사용자 액세스를 명시적으로 거부할 수는 없습니다. 루트 사용자의 권한을 제한하려면 AWS Organizations 서비스 제어 정책(SCP)을 사용해야 합니다.
- 루트 사용자 암호를 잊어버렸거나 분실한 경우 계정에 연결된 이메일 주소에 액세스할 수 있어야 암호를 재설정할 수 있습니다.
- 루트 사용자 액세스 키를 분실한 경우 루트 사용자로 계정에 로그인하여 새 액세스 키를 생성할 수 있어야 합니다.
- 일상적인 작업에 루트 사용자를 사용하지 마세요. 루트 사용자는 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용하세요. 루트 사용자로 로그인해야 하는 전체 작업 목록을 보려면 루트 사용자 보안 인증이 필요한 작업 섹션을 참조하세요.
- 보안 자격 증명은 계정에 특정합니다. 여러 AWS 계정에 액세스할 수 있는 경우, 각 계정마다 별도의 보안 인증 정보가 있습니다.

- 정책은 사용자, 역할 또는 사용자 그룹 멤버가 수행할 수 있는 작업, 작업의 대상 AWS 리소스 및 작업 수행 조건을 결정합니다. 정책을 사용하면 AWS 계정에서 AWS 서비스와 리소스에 대한 액세스 권한을 안전하게 제어할 수 있습니다. 보안 이벤트에 대응하여 권한을 수정하거나 취소해야 하는 경우, 자격 증명 직접 변경하는 것이 아니라 정책을 삭제하거나 수정합니다.
- 긴급 액세스 IAM 사용자의 로그인 보안 인증 정보와 프로그래밍 방식으로 액세스하기 위해 생성한 액세스 키는 안전한 위치에 저장해야 합니다. 액세스 키를 분실한 경우 계정에 로그인하여 새 액세스 키를 생성해야 합니다.
- IAM 사용자 및 액세스 키가 제공하는 장기 보안 인증 대신 IAM 역할 및 페더레이션 사용자가 제공하는 임시 보안 인증을 사용하는 것이 좋습니다.

AWS 보안 자격 증명을 사용한 프로그래밍 방식 액세스

가능한 경우 단기 액세스 키를 사용하여 AWS를 프로그래밍 방식으로 호출하거나 AWS Command Line Interface 또는 AWS Tools for PowerShell을 사용하는 것이 좋습니다. 하지만 이러한 용도에 장기 AWS 액세스 키를 사용할 수도 있습니다.

장기 액세스 키를 생성할 때는 액세스 키 ID(예: `AKIAIOSFODNN7EXAMPLE`)와 비밀 액세스 키(예: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`)를 세트로 생성합니다. 비밀 액세스 키는 생성할 때만 다운로드할 수 있습니다. 비밀 액세스 키를 다운로드하지 않았거나 분실한 경우, 액세스 키를 새로 생성해야 합니다.

대부분의 경우 IAM 사용자의 액세스 키를 생성할 때 갖는 만료되지 않는 장기 액세스 키가 필요하지 않습니다. 대신, IAM 역할을 만들고 임시 보안 인증 정보를 생성할 수 있습니다. 임시 보안 인증 정보는 액세스 키 ID와 비밀 액세스 키를 포함하지만, 보안 인증 정보가 만료되는 시간을 나타내는 보안 토큰도 포함합니다. 만료된 이후에는 더 이상 유효하지 않습니다. 자세한 내용은 장기 액세스 키의 대안 단원을 참조하세요.

`AKIA`로 시작하는 액세스 키 ID는 IAM 사용자 또는 AWS 계정 루트 사용자를 위한 장기 액세스 키입니다. `ASIA`로 시작하는 액세스 키 ID는 AWS STS 작업을 사용하여 생성하는 임시 보안 인증 액세스 키입니다.

사용자가 AWS Management Console 외부에서 AWS와 상호 작용하려면 프로그래밍 방식의 액세스가 필요합니다. 프로그래밍 방식으로 액세스를 부여하는 방법은 AWS에 액세스하는 사용자 유형에 따라 다릅니다.

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
작업 인력 ID (IAM Identity Center가 관리하는 사용자)	임시 보안 인증 정보를 사용하여 AWS CLI, AWS SDK 또는 AWS API에 대한 프로그래밍 요청에 서명합니다.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • AWS CLI에 대해서는 AWS Command Line Interface 사용 설명서에서 AWS IAM Identity Center을 사용하도록 AWS CLI 구성을 참조하세요. • AWS SDK, 도구,AWS API에 대해서는 AWS SDK 및 도구 참조 가이드에서 IAM Identity Center 인증을 참조하세요.
IAM	임시 보안 인증 정보를 사용하여 AWS CLI, AWS SDK 또는 AWS API에 대한 프로그래밍 요청에 서명합니다.	IAM 사용자 설명서의 AWS 리소스와 함께 임시 보안 인증 정보 사용에 나와 있는 지침을 따르세요.
IAM	(권장되지 않음) 장기 보안 인증 정보를 사용하여 AWS CLI, AWS SDK 또는 AWS API에 대한 프로그래밍 요청에 서명합니다.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • AWS CLI에 대해서는 AWS Command Line Interface 사용 설명서에서 IAM 사용자 보안 인증 정보를 사용한 인증을 참조하세요. • AWS SDK와 도구에 대해서는 AWS SDK 및 도구 참조 가이드에서 장기 보안 인증 정보를 사용한 인증을 참조하세요.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
		• AWS API에 대해서는 IAM 사용자 설명서에서 IAM 사용자의 액세스 키 관리를 참조하세요.

장기 액세스 키의 대안

대부분의 일반적인 사용 사례에서는 장기 액세스 키 대신 사용할 수 있는 대안이 있습니다. 계정 보안을 강화하려면 다음 사항을 고려하세요.

- 애플리케이션 코드나 코드 리포지토리에 장기 액세스 키와 비밀 액세스 키를 포함해서는 안 됨 - 대신, 일반 텍스트로 키를 하드 코딩할 필요가 없도록 AWS Secrets Manager 또는 기타 비밀 관리 솔루션을 사용하세요. 그러면 애플리케이션 또는 클라이언트가 필요할 때 비밀 정보를 검색할 수 있습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 AWS Secrets Manager이란 무엇입니까? 섹션을 참조하십시오.

- 가능하면 IAM 역할을 사용하여 임시 보안 인증 정보 생성 - 가능하면 장기 액세스 키보다는 항상 메커니즘을 사용하여 임시 보안 인증 정보를 발급하세요. 임시 보안 인증 정보는 사용자와 함께 저장되지 않지만 동적으로 생성되어 요청 시 사용자에게 제공되므로 더 안전합니다. 임시 보안 인증 정보는 수명이 제한되어 있으므로 관리하거나 업데이트할 필요가 없습니다. 임시 액세스 키를 제공하는 메커니즘으로는 IAM 역할, IAM Identity Center 사용자의 인증 등이 있습니다. AWS 외부에서 실행되는 시스템의 경우 AWS Identity and Access Management Roles Anywhere를 사용할 수 있습니다.
- AWS Command Line Interface(AWS CLI) 또는 **`aws-shell`**에 대해 장기 액세스 키의 대안 사용 – 다음을 대안으로 사용할 수 있습니다.
  - AWS CloudShell은 브라우저 기반의 사전 인증된 셸로, AWS Management Console에서 직접 실행할 수 있습니다. 원하는 셸(Bash, PowerShell 또는 Z 셸)을 통해 AWS 서비스에 대해 AWS CLI 명령을 실행할 수 있습니다. 이 경우 명령줄 도구를 다운로드하거나 설치할 필요가 없습니다. 자세한 내용은 AWS CloudShell 사용 설명서의 AWS CloudShell이란 무엇입니까? 섹션을 참조하십시오.
  - AWS IAM Identity Center(IAM Identity Center)와 AWS CLI 버전 2의 통합. 사용자를 인증하고, AWS CLI 명령을 실행하기 위한 단기 보안 인증을 제공할 수 있습니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 AWS CLI와 IAM Identity Center 통합 및 AWS Command Line Interface 사용 설명서의 IAM Identity Center를 사용하도록 AWS CLI 구성을 참조하세요.

- 애플리케이션 또는 AWS 서비스에 액세스해야 하는 인간 사용자를 위해 장기 액세스 키를 생성해서는 안 됨 - IAM Identity Center는 외부 IdP 사용자가 AWS 서비스에 액세스할 수 있도록 임시 액세스 보안 인증 정보를 생성할 수 있습니다. 따라서 IAM에서 장기 보안 인증 정보를 생성하고 관리할 필요가 없습니다. IAM Identity Center에서 외부 IdP 사용자에게 액세스 권한을 부여하는 IAM Identity Center 권한 세트를 생성합니다. 그런 다음 선택한 AWS 계정에서 IAM Identity Center의 그룹을 권한 세트에 할당합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center이란 무엇인가요?, 외부 자격 증명 공급자 연결 및 권한 세트를 참조하세요.
- AWS 컴퓨팅 서비스 내에 장기 액세스 키를 저장해서는 안 됨 - 대신, 컴퓨팅 리소스에 IAM 역할을 할당하세요. 그러면 액세스 권한을 부여하기 위한 임시 보안 인증이 자동으로 제공됩니다. 예를 들어 Amazon EC2 인스턴스에 연결된 인스턴스 프로파일을 만들 때, 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 해당 역할을 사용할 수 있도록 할 수 있습니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 Amazon EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증 정보를 얻을 수 있습니다. 자세히 알아보려면 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

AWS 보안 감사 지침

보안 구성을 정기적으로 감사하여 현재 비즈니스 요구를 충족하는지 확인해야 합니다. 감사를 통해 불필요한 IAM 사용자, 역할, 그룹 및 정책을 제거하고 사용자와 소프트웨어에 과도한 권한이 부여되지 않는지 확인할 수 있습니다.

다음은 보안 모범 사례에 따라 AWS 리소스를 체계적으로 검토 및 모니터링하기 위한 지침입니다.

> **Tip**
>
> AWS Security Hub을 사용하여 보안 모범 사례와 관련된 IAM의 사용량을 모니터링하십시오. Security Hub는 보안 제어를 사용하여 리소스 구성 및 보안 표준을 평가하여 다양한 규정 준수 프레임워크를 준수할 수 있도록 지원합니다. Security Hub를 사용하여 IAM 리소스를 평가하는 방법에 대한 자세한 내용은 AWS Security Hub 사용 설명서의 AWS 신원과 액세스 관리 제어를 참조하십시오.

내용

- IAM 사용자 검토
- IAM 그룹 검토
- IAM 역할 검토
- SAML 및 OpenID Connect(OIDC)에 대한 IAM 공급자 검토
- 모바일 앱 검토
- IAM 정책 검토를 위한 팁

보안 감사를 해야 하는 경우

다음과 같은 경우에 보안 구성을 감사해야 합니다.

- 주기적으로. 보안을 위한 모범 사례로 이 문서에 설명한 단계를 정기적으로 수행해야 합니다.
- 조직 내에 변경 사항이 있는 경우(예: 직원 퇴사)
- 계정의 사용자에게 더 이상 필요하지 않은 권한을 제거했는지 확인하기 위해 개별 AWS 서비스를 한 가지 이상 사용 중단한 경우
- 계정에서 소프트웨어를 추가하거나 제거한 경우(예: Amazon EC2 인스턴스, AWS OpsWorks 스택, AWS CloudFormation 템플릿 등의 애플리케이션)
- 권한 없는 사용자가 계정에 액세스한 것으로 의심되는 경우

감사 지침

계정의 보안 구성을 검토할 때 다음 지침을 따르십시오.

- 철저히. 거의 사용하지 않는 구성을 포함해 보안 구성의 모든 측면을 조사합니다.
- 가정 금지. 보안 구성의 일부 측면(예: 특정 정책의 배후 추론, 역할의 존재)에 익숙하지 않은 경우 잠재 위험을 이해할 때까지 비즈니스 요구를 조사합니다.
- 간소화. 감사(및 관리)를 간소화하려면 IAM 그룹, IAM 역할, 일관된 이름 지정 체계 및 간단한 정책을 사용합니다.

AWS 계정 자격 증명 검토

AWS 계정 자격 증명을 감사할 때 다음 단계를 수행합니다.

1. 사용하지 않는 루트 사용자 액세스 키가 있다면 그 키는 삭제해도 됩니다. 일상적인 AWS 작업에는 루트 액세스 키를 사용하지 않는 것이 좋습니다. 대신 AWS IAM Identity Center의 사용자 같은 임시 보안 인증 정보가 할당된 사용자를 사용합니다.
2. 계정에 대해 액세스 키가 필요한 경우 필요할 때 해당 키를 업데이트해야 합니다.

IAM 사용자 검토

기존 IAM 사용자를 감사할 때 다음 단계를 수행합니다.

1. 사용자를 나열한 다음 필요 없는 사용자는 삭제합니다.
2. 액세스가 필요하지 않은 사용자는 그룹에서 제거합니다.
3. 사용자가 소속된 그룹에 연결된 정책을 검토합니다. IAM 정책 검토를 위한 팁 섹션을 참조하세요.
4. 사용자에게 필요하지 않거나 노출된 보안 자격 증명을 삭제합니다. 예를 들어 애플리케이션에 사용되는 IAM 사용자에게는 암호가 필요하지 않습니다(AWS 웹 사이트 로그인 시에만 필요함). 마찬가지로 사용자는 액세스 키가 필요 없는 경우 액세스 키를 가져야 할 이유가 없습니다. 자세한 내용은 IAM 사용자의 암호 관리 및 IAM 사용자의 액세스 키 관리를 참조하세요.

   계정의 모든 IAM 사용자와 해당 사용자의 암호, 액세스 키, MFA 디바이스 등 다양한 자격 증명의 상태를 나열하는 자격 증명 보고서를 생성하고 다운로드할 수 있습니다. 암호와 액세스 키의 경우 보안 인증 정보 보고서를 통해 암호 또는 액세스 키가 마지막으로 사용된 일자와 시각을 알 수 있습니다. 계정에서 최근에 사용하지 않은 보안 인증 정보는 삭제하는 것도 좋습니다. (긴급 액세스 사용자는 제거하지 마세요.) 자세한 내용은 AWS 계정의 보안 인증 보고서 가져오기를 참조하세요.
5. 장기 보안 인증이 필요한 사용 사례에 필요한 경우 암호 및 액세스 키를 업데이트합니다. 자세한 내용은 IAM 사용자의 암호 관리 및 IAM 사용자의 액세스 키 관리를 참조하세요.
6. 가장 좋은 방법은 인간 사용자가 ID 제공업체와의 페더레이션을 사용하여 임시 보안 인증으로 AWS에 액세스하도록 하는 것입니다. 가능하면 IAM 사용자에서 페더레이션 사용자(예: IAM Identity Center 사용자)로 전환하십시오. 애플리케이션에 필요한 최소 IAM 사용자 수를 유지하십시오.

IAM 그룹 검토

IAM 그룹을 감사할 때 다음 단계를 수행합니다.

1. 그룹을 나열한 다음 사용하지 않는 그룹을 삭제합니다.
2. 각 그룹의 사용자를 검토하고 소속되지 않은 사용자를 제거합니다.
3. 그룹에 연결된 정책을 검토합니다. IAM 정책 검토를 위한 팁 섹션을 참조하세요.

IAM 역할 검토

IAM 역할을 감사할 때 다음 단계를 수행합니다.

1. 역할을 나열한 다음 사용하지 않는 역할을 삭제합니다.
2. 역할의 신뢰 정책을 검토합니다. 주체가 누구이고 계정 또는 사용자가 역할을 수임할 수 있어야 하는 이유를 이해해야 합니다.
3. 역할에 대한 액세스 정책을 검토하여 역할을 수임하는 사용자에게 적절한 권한을 부여했는지 확인합니다. IAM 정책 검토를 위한 팁 단원을 참조하십시오.

SAML 및 OpenID Connect(OIDC)에 대한 IAM 공급자 검토

SAML 또는 OIDC ID 제공업체(IdP)와의 신뢰를 구축하기 위해 IAM 엔터티를 만든 경우 다음 단계를 수행합니다.

1. 미사용 공급자를 삭제합니다.
2. 각 SAML IdP에 대한 AWS 메타데이터 문서를 다운로드하여 검토하고 문서에 최신 비즈니스 요구 사항이 반영되어 있는지 확인합니다.
3. SAML IdP에서 최신 메타데이터 문서를 가져온 다음 IAM에서 공급자를 업데이트합니다.

모바일 앱 검토

AWS에 요청하는 모바일 앱을 만든 경우 다음 단계를 수행하십시오.

1. 암호화된 스토리지를 포함하여 모바일 앱에 포함된 액세스 키가 있는지 확인합니다.
2. 해당 목적으로 설계된 API를 사용하여 앱에 대한 임시 보안 인증 정보를 얻습니다.

> **Note**
>
> Amazon Cognito를 사용하여 앱에서 사용자 자격 증명을 관리하는 것이 좋습니다. 이 서비스를 사용하면 Login with Amazon, Facebook, Google 또는 OpenID Connect(OIDC) 호환 자격 증명 공급자를 통해 사용자를 인증할 수 있습니다. 자세한 내용은 Amazon Cognito 개발자 안내서의 Amazon Cognito 자격 증명 풀을 참조하세요.

IAM 정책 검토를 위한 팁

정책은 강력하고 세부적이므로 각 정책에 부여되는 권한을 조사하여 이해하는 것이 중요합니다. 정책을 검토할 때 다음 지침을 사용합니다.

- 개별 사용자가 아니라 그룹 또는 역할에 정책을 연결합니다. 개별 사용자가 정책이 있는 경우 해당 사용자가 정책을 필요로 하는 이유를 이해해야 합니다.
- IAM 사용자, 그룹 및 역할에 필요한 권한이 있고 추가 권한이 없는지 확인하십시오.
- IAM 정책 시뮬레이터를 사용하여 사용자 또는 그룹에 연결되는 정책을 테스트합니다.
- 사용자의 권한은 해당하는 모든 정책, 즉 아이덴티티 기반 정책(사용자, 그룹, 역할)과 리소스 기반 정책(Amazon S3 버킷, Amazon SQS 대기열, Amazon SNS 주제, AWS KMS 키 등 리소스에 관한 정책) 모두의 결과입니다. 사용자에게 적용되는 모든 정책을 검사하여 개별 사용자에게 부여된 전체 권한을 이해하는 것이 중요합니다.
- 사용자가 IAM 사용자, 그룹, 역할 또는 정책을 생성하고 정책을 주체 엔터티에 연결할 수 있도록 허용하여 해당 사용자에게 계정의 모든 리소스에 대한 모든 권한을 효율적으로 부여할 수 있습니다. 정책을 생성하여 사용자, 그룹 또는 역할에 연결할 수 있도록 허용된 사용자는 스스로 모든 권한을 부여할 수 있습니다. 일반적으로 계정의 리소스에 대한 모든 액세스 권한을 가진 신뢰할 수 없는 사용자 또는 역할에는 IAM 권한을 부여하지 않습니다. 보안 감사를 수행할 때 신뢰할 수 있는 ID에 다음 IAM 권한이 부여되었는지 확인하십시오.
  - `iam:PutGroupPolicy`
  - `iam:PutRolePolicy`
  - `iam:PutUserPolicy`
  - `iam:CreatePolicy`
  - `iam:CreatePolicyVersion`
  - `iam:AttachGroupPolicy`
  - `iam:AttachRolePolicy`
  - `iam:AttachUserPolicy`
- 정책에서 사용되지 않는 서비스에 대한 권한을 부여하지 않는지 확인합니다. 예를 들어, AWS 관리형 정책을 사용하는 경우 계정에서 사용 중인 AWS 관리형 정책이 실제로 사용하는 서비스에 대한 정책인지를 확인합니다. AWS 관리형 정책이 계정에서 사용 중인지 확인하려면 IAM `GetAccountAuthorizationDetails` API(AWS CLI 명령: `aws iam get-account-authorization-details`)를 사용합니다.

- 정책에서 사용자에게 Amazon EC2 인스턴스를 시작할 수 있는 권한을 부여할 경우 `iam:PassRole` 작업도 허용할 수 있습니다. 하지만 그럴 경우 사용자가 Amazon EC2 인스턴스에 전달할 수 있는 역할을 명시적으로 나열해야 합니다.
- `*`이 들어 있는 `Action` 또는 `Resource` 요소에 대한 값을 면밀히 검토합니다. 가능하면 사용자에게 필요한 개별 작업 및 리소스에 대해 `Allow` 액세스 권한을 부여하세요. 정책에서 `*`를 사용하는 것이 적합한 이유는 다음과 같습니다.
  - 정책은 관리 수준 권한을 부여하도록 설계되었습니다.
  - 와일드카드 문자는 비슷한 작업(예: `Describe*`)에 사용되며 이러한 방법으로 참조되는 전체 작업 목록에 익숙해지면 편리합니다.
  - 와일드카드 문자는 리소스 클래스 또는 리소스 경로(예: `arn:aws:iam::account-id:users/division_abc/*`)를 나타내는 데 사용되며 해당 클래스 또는 경로에 속하는 모든 리소스에 편리하게 액세스 권한을 부여할 수 있습니다.
  - 서비스 작업에서는 리소스 수준 권한을 지원하지 않습니다. 리소스에는 `*`만 사용할 수 있습니다.
- 정책 이름을 검사하여 정책의 기능이 반영되어 있는지 확인합니다. 예를 들어, 정책 이름에 "읽기 전용"이 포함되어 있지만 실제로 정책에서는 쓰기 또는 변경 권한을 부여할 수도 있습니다.

보안 감사 계획에 대한 자세한 내용은 AWS 아키텍처 센터의 보안, 자격 증명 및 규정 준수를 위한 모범 사례를 참조하세요.

AWS Identity and Access Management의 데이터 보호

AWS 공동 책임 모델은 AWS Identity and Access Management의 데이터 보호에 적용됩니다. 이 모델에서 설명하는 것처럼 AWS는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하십시오.

데이터를 보호하려면 AWS 계정보안 인증 정보를 보호하고 AWS IAM Identity Center또는 AWS Identity and Access Management(IAM)를 통해 개별 사용자 계정을 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신하세요. TLS 1.2는 필수이며 TLS 1.3를 권장합니다.

- AWS CloudTrail로 API 및 사용자 활동 로깅을 설정하세요.
- AWS 암호화 솔루션을 AWS 서비스 내의 모든 기본 보안 컨트롤과 함께 사용하세요.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령행 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-3 검증된 암호화 모듈이 필요한 경우, FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 IAM 또는 기타 AWS 서비스에서 콘솔, API, AWS CLI 또는 AWS SDK를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

IAM 및 AWS STS의 데이터 암호화

데이터 암호화는 일반적으로 저장 데이터 암호화와 전송 중 데이터 암호화의 두 가지 범주로 나뉩니다.

저장 중 암호화

IAM에 의해 수집 및 저장되는 데이터는 유휴 시 암호화됩니다.

- IAM - IAM 내에서 수집되고 저장되는 데이터에는 IP 주소, 고객 계정 메타데이터 및 암호를 포함한 고객 식별 데이터가 포함됩니다. 고객 계정 메타데이터 및 고객 식별 데이터는 유휴 시 AES 256을 사용하여 암호화되거나 SHA 256을 사용하여 해시됩니다.
- AWS STS – AWS STS 는 서비스에 대한 성공, 오류 및 잘못된 요청을 기록하는 서비스 로그를 제외하고 고객 콘텐츠를 수집하지 않습니다.

전송 중 암호화

암호를 포함한 고객 식별 데이터는 TLS 1.2 및 1.3을 사용하여 전송 중 암호화됩니다. 모든 AWS STS 엔드포인트는 전송 중 데이터를 암호화하기 위해 HTTPS를 지원합니다. AWS STS 엔드포인트 목록은 리전 및 엔드포인트 섹션을 참조하세요.

IAM 및 AWS STS의 키 관리

IAM 또는 AWS STS를 사용하여 암호화 키를 관리할 수 없습니다. 암호화 키에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 AWS KMS란 무엇인가요?를 참조하세요.

IAM 및 AWS STS의 인터네트워크 트래픽 개인 정보 보호

전송 계층 보안 프로토콜(TLS)을 사용하여 IAM에 대한 요청을 수행해야 합니다. VPC 엔드포인트를 사용하여 AWS STS 서비스에 대한 연결을 보호할 수 있습니다. 자세한 내용은 인터페이스 VPC 엔드포인트 섹션을 참조하세요.

AWS Identity and Access Management의 로깅 및 모니터링

모니터링은 AWS Identity and Access Management(IAM), AWS Security Token Service(AWS STS) 및 기타 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. AWS는 AWS 리소스를 모니터링하고 잠재적 인시던트에 대응할 수 있는 여러 가지 도구를 제공합니다.

- AWS CloudTrail는 콘솔과 API 호출을 비롯한 모든 IAM 및 AWS STS API 호출을 이벤트로 캡처합니다. CloudTrail을 IAM 및 AWS STS과 함께 사용하는 방법에 대해 자세히 알아보려면 AWS CloudTrail을 사용하여 IAM 및 AWS STS API 호출 로깅 섹션을 참조하세요. CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.
- AWS Identity and Access Management Access Analyzer를 사용하면 외부 엔터티와 공유되는 조직 및 계정 내 리소스(예: Amazon S3 버킷 또는 IAM 역할)를 식별할 수 있습니다. 이렇게 하면 리소스 및 데이터에 대한 의도하지 않은 액세스라는 보안 위험을 식별할 수 있습니다. 자세한 내용은 IAM Access Analyzer란 무엇인가요?를 참조하세요.
- Amazon CloudWatch는 AWS에서 실행하는 AWS 리소스와 애플리케이션을 실시간으로 모니터링합니다. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성할 수 있으며, 지정된 지표가 지정한 임계값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예를 들어 CloudWatch에서 EC2 인스턴스의 CPU 사용량 또는 기타 지표를 추적하고 필요할 때 자동으로 새 인스턴스를 시작할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.
- Amazon CloudWatch Logs로 Amazon EC2 인스턴스, CloudTrail, 기타 소스의 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임곗값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구력 있는 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하십시오.

IAM에 대한 추가 리소스 및 보안 모범 사례는 AWS Identity and Access Management의 보안 모범 사례 및 사용 사례 섹션을 참조하세요.

주제

- AWS CloudTrail을 사용하여 IAM 및 AWS STS API 호출 로깅

AWS CloudTrail을 사용하여 IAM 및 AWS STS API 호출 로깅

IAM 및 AWS STS는 사용자 또는 역할이 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 콘솔과 API 호출을 비롯한 모든 IAM에 대한 API 호출 및 AWS STS을 이벤트로 캡처합니다. 추적을 생성하면 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail을 사용하여 IAM 또는 AWS STS에 요청한 내용의 정보를 얻을 수 있습니다. 예를 들어 어떤 IP 주소에서 요청했는지, 누가 요청했는지, 언제 생성되었는지 등 추가적인 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.

주제

CloudTrail의 IAM 및 AWS STS 정보

CloudTrail은 계정 생성 시 AWS 계정에서 사용되도록 설정됩니다. IAM 또는 AWS STS에서 활동이 발생하면 해당 활동이 이벤트 기록(Event history)의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 정보는 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

IAM 및 AWS STS 이벤트를 비롯하여 AWS 계정의 이벤트 기록을 보유하려면 추적을 생성하세요. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 지역에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 지역의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 또는 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

모든 IAM 및 AWS STS 작업은 CloudTrail에서 로깅되고 IAM API 참조 및 AWS Security Token Service API 참조에 기록됩니다.

IAM 및 AWS STS API 요청 로깅

CloudTrail은 인증된 모든 API 요청을 IAM 및 AWS STS API 작업으로 로그합니다. 또한 CloudTrail은 인증되지 않은 요청을 AWS STS 작업, `AssumeRoleWithSAML` 및 `AssumeRoleWithWebIdentity`에 로그하고 자격 증명 제공자가 제공한 정보를 로그합니다. 그러나 인증되지 않은 일부 AWS STS 요청은 합법적인 요청으로 신뢰할 수 있을 만큼 충분히 유효해야 한다는 최소 기대치를 충족하지 못하기 때문에 기록되지 않을 수 있습니다.

로그된 정보를 사용하여 위임된 역할을 지닌 페더레이션 사용자의 호출을 외부 페더레이션 호출자에 다시 매핑할 수 있습니다. `AssumeRole`의 경우, 호출을 원래 AWS 서비스 또는 원래 사용자의 계정에 다시 매핑할 수 있습니다. CloudTrail 로그 항목에 있는 JSON 데이터의 `userIdentity` 섹션에는 특정 페더레이션 사용자에게 AssumeRole* 요청을 매핑하는 데 필요한 정보가 들어 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail userIdentity 요소를 참조하세요.

예를 들어 IAM `CreateUser`, `DeleteRole`, `ListGroups` 및 기타 API 작업에 대한 호출은 모두 CloudTrail에 로깅됩니다.

이러한 유형의 로그 항목에 대한 예시는 이번 주제의 뒷 부분에 제시됩니다.

다른 AWS 서비스에 대한 API 요청 로깅

다른 AWS 서비스 API 작업에 대해 인증된 요청은 CloudTrail에 로깅되며, 이 로그 항목에는 요청자에 대한 정보가 저장됩니다.

예를 들어 Amazon EC2 인스턴스 나열 요청을 했거나 AWS CodeDeploy 배포 그룹을 생성했다고 가정합시다. 요청한 사람이나 서비스에 대한 세부 내용은 그 요청의 로그 항목에 들어 있습니다. 이 정보로 AWS 계정 루트 사용자, IAM 사용자, 역할, 또는 다른 AWS 서비스에 의한 요청인지 판단할 수 있습니다.

CloudTrail 로그 항목의 사용자 자격 증명 정보에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 userIdentity 요소를 참조하세요.

사용자 로그인 이벤트 로깅

CloudTrail은 로그인 이벤트를 AWS Management Console, AWS 토론 포럼 및 AWS Marketplace에 로그합니다. CloudTrail은 IAM 사용자 및 페더레이션 사용자에 대해 성공 및 실패한 로그인 시도를 기록합니다.

루트 사용자 로그인의 성공 및 실패에 대한 샘플 CloudTrail 이벤트를 보려면 AWS CloudTrail사용 설명서의 루트 사용자에 대한 이벤트 레코드의 예제를 참조하세요.

보안 모범 사례의 일환으로 AWS는 잘못된 사용자 이름으로 인해 로그인에 실패하더라도 입력한 IAM 사용자 이름 텍스트를 로깅하지 않습니다. 사용자 이름 텍스트는 `HIDDEN_DUE_TO_SECURITY_REASONS` 값으로 마스킹 처리됩니다. 마스킹 처리의 예는 이번 주제 후반부의 잘못된 사용자 이름으로 인한 로그인 실패 이벤트의 예 섹션을 참조하세요. 이러한 오류는 사용자 오류로 인해 발생할 수 있으므로 사용자 이름 텍스트는 가려집니다. 이러한 오류를 기록하면 잠재적으로 중요한 정보가 노출될 수 있습니다. 예:

- 우발적으로 사용자 이름 상자에 암호를 입력한 경우
- AWS 계정의 로그인 페이지 링크를 선택하고서 다른 AWS 계정의 계정 번호를 입력하는 경우
- 로그인하려는 계정을 잊고 우발적으로 개인 이메일 계정의 사용자 이름, 금융 서비스 로그인 식별자, 또는 기타 프라이빗 ID를 입력하는 경우

임시 자격 증명에 대한 로그인 이벤트 로깅

보안 주체가 임시 자격 증명을 요청할 때 보안 주체 유형에 의해 이벤트를 CloudTrail 로그에 기록하는 방법이 결정됩니다. 보안 주체가 다른 계정의 역할을 맡는 경우 이는 복잡할 수 있습니다. 역할 크로스 계정 작업과 관련된 작업을 수행하기 위한 여러 API 호출이 있습니다. 먼저 보안 주체는 AWS STS API를 호출하여 임시 자격 증명을 검색합니다. 해당 작업은 호출 계정과 AWS STS 작업이 수행되는 계정에 기록됩니다. 그런 다음 보안 주체는 역할을 사용하여 맡은 역할의 계정에서 다른 API 호출을 수행합니다.

역할 신뢰 정책에서 `sts:SourceIdentity` 조건 키를 사용하여 사용자가 역할을 수임할 때 자격 증명을 지정하도록 요구할 수 있습니다. 예를 들어 IAM 사용자가 자신의 사용자 이름을 소스 자격 증명으로 지정하도록 요구할 수 있습니다. 이를 통해 AWS에서 특정 작업을 수행한 사용자를 확인할 수 있습니다. 자세한 내용은 `sts:SourceIdentity` 단원을 참조하십시오. 역할 신뢰 정책에서 `sts:RoleSessionName`을 사용하여 사용자가 역할을 수임할 때 세션 이름을 지정하도록 요구할 수 있습니다. 이렇게 하면 AWS CloudTrail 로그를 검토할 때 여러 보안 주체가 사용하는 역할의 역할 세션을 구분할 수 있습니다.

다음 표는 임시 자격 증명을 생성하는 각 AWS STS API 호출에 대해 CloudTrail에서 다양한 사용자 자격 증명 정보를 기록하는 방법을 보여줍니다.

보안 주체 유형	STS API	호출자 계정에 대한 CloudTrail 로그의 사용자 자격 증명	수임하는 역할 계정에 대한 CloudTrail 로그의 사용자 자격 증명	역할의 후속 API 호출에 대한 CloudTrail 로그의 사용자 자격 증명
AWS 계정 루트 사용자 보안 인증	GetSessionToken	루트 사용자 자격 증명	역할 소유자 계정은 호출 계정과 동일	루트 사용자 자격 증명
IAM 사용자	GetSessionToken	IAM 사용자 자격 증명	역할 소유자 계정은 호출 계정과 동일	IAM 사용자 자격 증명
IAM 사용자	GetFederationToken	IAM 사용자 자격 증명	역할 소유자 계정은 호출 계정과 동일	IAM 사용자 자격 증명
IAM 사용자	AssumeRole	IAM 사용자 자격 증명	계정 번호 및 보안 주체 ID(사용자인 경우) 또는 AWS 서비스 보안 주체	역할 자격 증명만(사용자 없음)
외부에서 인증된 사용자	AssumeRoleWithSAML	해당 사항 없음	SAML 사용자 자격 증명	역할 자격 증명만(사용자 없음)
외부에서 인증된 사용자	AssumeRoleWithWebIdentity	해당 사항 없음	OIDC/웹 사용자 자격 증명	역할 자격 증명만(사용자 없음)

CloudTrail은 리소스를 변형하지 않는 작업을 읽기 전용으로 간주합니다. 읽기 전용 이벤트를 기록할 때 CloudTrail은 로그에서 `responseElements` 정보를 수정합니다. CloudTrail이 읽기 전용이 아닌 이벤트를 기록하는 경우 로그 항목에 전체 `responseElements`가 표시됩니다. 하지만 AWS STS API

AssumeRole, AssumeRoleWithSAML 및 AssumeRoleWithWebIdentity의 경우 읽기 전용으로 기록되더라도 CloudTrail은 이러한 API의 로그에 전체 responseElements를 포함합니다.

다음 표는 임시 자격 증명을 생성하는 각 AWS STS API 호출에 대해 CloudTrail에서 responseElements 및 readOnly 정보를 기록하는 방법을 보여줍니다.

STS API	응답 요소 정보	읽기 전용
AssumeRole	포함	true
AssumeRoleWithSAML	포함	true
AssumeRoleWithWebIdentity	포함	true
GetFederationToken	포함	false
GetSessionToken	포함	false

CloudTrail 로그의 IAM API 이벤트 예제

CloudTrail 로그 파일에는 JSON 형식의 이벤트 정보가 포함되어 있습니다. API 이벤트는 단일 API 요청을 나타내며 보안 주체, 요청된 작업, 모든 파라미터, 작업 날짜 및 시간에 대한 정보를 포함합니다.

CloudTrail 로그 파일의 IAM API 이벤트 예제

다음은 IAM GetUserPolicy 작업 요청에 대한 CloudTrail 로그 항목 예제입니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/JaneDoe",
    "accountId": "444455556666",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "userName": "JaneDoe",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2014-07-15T21:39:40Z"
      }
```

```
    },
    "invokedBy": "signin.amazonaws.com"
  },
  "eventTime": "2014-07-15T21:40:14Z",
  "eventSource": "iam.amazonaws.com",
  "eventName": "GetUserPolicy",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "signin.amazonaws.com",
  "userAgent": "signin.amazonaws.com",
  "requestParameters": {
    "userName": "JaneDoe",
    "policyName": "ReadOnlyAccess-JaneDoe-201407151307"
  },
  "responseElements": null,
  "requestID": "9EXAMPLE-0c68-11e4-a24e-d5e16EXAMPLE",
  "eventID": "cEXAMPLE-127e-4632-980d-505a4EXAMPLE"
}
```

위 이벤트 정보에서 `ReadOnlyAccess-JaneDoe-201407151307` 요소에 지정한 것처럼 사용자 `JaneDoe`을 의미하는 `requestParameters`이라는 이름의 사용자 정책을 가져오는 요청인 것을 알 수 있습니다. 그 밖에도 요청자는 `JaneDoe`이라는 이름의 IAM 사용자이고, 2014년 7월 15일 오후 9시 40분(UTC)에 생성된 것도 확인 가능합니다. 여기서는 `userAgent` 요소를 통해 요청이 AWS Management Console에서 이루어진 것도 알 수 있습니다.

CloudTrail 로그의 AWS STS API 이벤트 예제

CloudTrail 로그 파일에는 JSON 형식의 이벤트 정보가 포함되어 있습니다. API 이벤트는 단일 API 요청을 나타내며 보안 주체, 요청된 작업, 모든 파라미터, 작업 날짜 및 시간에 대한 정보를 포함합니다.

CloudTrail 로그 파일의 크로스 계정 AWS STS API 이벤트의 예

계정 777788889999에 `JohnDoe`라는 이름의 IAM 사용자가 계정 111122223333의 역할 `EC2-dev`를 수임하기 위해 AWS STS AssumeRole 작업을 호출합니다. 계정 관리자는 사용자가 역할을 수임할 때 해당 사용자 이름과 동일한 소스 자격 증명을 설정하도록 요구합니다. 사용자는 값이 `JohnDoe`인 소스 자격 증명을 전달합니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAQRSTUVWXYZEXAMPLE",
    "arn": "arn:aws:iam::777788889999:user/JohnDoe",
```

```
    "accountId": "777788889999",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "JohnDoe"
  },
  "eventTime": "2014-07-18T15:07:39Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRole",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.11.10 Python/2.7.8
 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67",
  "requestParameters": {
    "roleArn": "arn:aws:iam::111122223333:role/EC2-dev",
    "roleSessionName": "JohnDoe-EC2-dev",
    "sourceIdentity": "JohnDoe",
    "serialNumber": "arn:aws:iam::777788889999:mfa"
  },
  "responseElements": {
    "credentials": {
      "sessionToken": "<encoded session token blob>",
      "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
      "expiration": "Jul 18, 2023, 4:07:39 PM"
      },
    "assumedRoleUser": {
      "assumedRoleId": "AIDAQRSTUVWXYZEXAMPLE:JohnDoe-EC2-dev",
      "arn": "arn:aws:sts::111122223333:assumed-role/EC2-dev/JohnDoe-EC2-dev"
    },
  "sourceIdentity": "JohnDoe"
  },
  "resources": [
    {
      "ARN": "arn:aws:iam::111122223333:role/EC2-dev",
      "accountId": "111122223333",
      "type": "AWS::IAM::Role"
    }
  ],
  "requestID": "4EXAMPLE-0e8d-11e4-96e4-e55c0EXAMPLE",
  "sharedEventID": "bEXAMPLE-efea-4a70-b951-19a88EXAMPLE",
  "eventID": "dEXAMPLE-ac7f-466c-a608-4ac8dEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

두 번째 예제는 동일한 요청에 대해 수임한 역할 계정(111122223333)의 CloudTrail 로그 항목입니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AWSAccount",
    "principalId": "AIDAQRSTUVWXYZEXAMPLE",
    "accountId": "777788889999"
  },
  "eventTime": "2014-07-18T15:07:39Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRole",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.11.10 Python/2.7.8
 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67",
  "requestParameters": {
    "roleArn": "arn:aws:iam::111122223333:role/EC2-dev",
    "roleSessionName": "JohnDoe-EC2-dev",
    "sourceIdentity": "JohnDoe",
    "serialNumber": "arn:aws:iam::777788889999:mfa"
  },
  "responseElements": {
    "credentials": {
      "sessionToken": "<encoded session token blob>",
      "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
      "expiration": "Jul 18, 2014, 4:07:39 PM"
    },
    "assumedRoleUser": {
      "assumedRoleId": "AIDAQRSTUVWXYZEXAMPLE:JohnDoe-EC2-dev",
      "arn": "arn:aws:sts::111122223333:assumed-role/EC2-dev/JohnDoe-EC2-dev"
      },
  "sourceIdentity": "JohnDoe"
  },
  "requestID": "4EXAMPLE-0e8d-11e4-96e4-e55c0EXAMPLE",
  "sharedEventID": "bEXAMPLE-efea-4a70-b951-19a88EXAMPLE",
  "eventID": "dEXAMPLE-ac7f-466c-a608-4ac8dEXAMPLE"
}
```

CloudTrail 로그 파일의 AWS STS 역할 체인 API 이벤트의 예

다음은 111111111111 계정의 John Doe가 만든 요청에 대한 CloudTrail 로그 항목 예제입니다. John은 이전에 자신의 JohnDoe 사용자를 사용하여 JohnRole1 역할을 맡았습니다. 이 요청에 대해 그

는 해당 역할의 자격 증명을 사용하여 JohnRole2 역할을 맡습니다. 이를 역할 체인이라고 합니다. JohnDoe1 역할을 수임할 때 설정한 소스 자격 증명은 JohnRole2에 대한 수임 요청에서 지속됩니다. John이 역할을 수임할 때 다른 소스 자격 증명을 설정하려고 하면 요청이 거부됩니다. John은 요청에 두 개의 세션 태그를 전달합니다. 그는 두 태그를 전이적으로 설정합니다. John은 JohnRole1을 맡을 때 전이적으로 설정했기 때문에 요청은 Department 태그를 전이적으로 상속합니다. 소스 자격 증명에 대한 자세한 내용은 위임된 역할로 수행한 작업 모니터링 및 제어 섹션을 참조하세요. 역할 체인의 전이적 키에 대한 자세한 내용은 세션 태그를 사용하는 역할 체인 섹션을 참조하세요.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIN5ATK5U7KEXAMPLE:JohnRole1",
        "arn": "arn:aws:sts::111111111111:assumed-role/JohnDoe/JohnRole1",
        "accountId": "111111111111",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2019-10-02T21:50:54Z"
            },
                "sessionIssuer": {
                "type": "Role",
               "principalId": "AROAIN5ATK5U7KEXAMPLE",
                "arn": "arn:aws:iam::111111111111:role/JohnRole1",
                "accountId": "111111111111",
                "userName": "JohnDoe"
            },
            "sourceIdentity": "JohnDoe"
        }
    },
    "eventTime": "2019-10-02T22:12:29Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRole",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "123.145.67.89",
    "userAgent": "aws-cli/1.16.248 Python/3.4.7
 Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 botocore/1.12.239",
    "requestParameters": {
        "incomingTransitiveTags": {
            "Department": "Engineering"
        },
```

```
        "tags": [
            {
                "value": "johndoe@example.com",
                "key": "Email"
            },
            {
                "value": "12345",
                "key": "CostCenter"
            }
        ],
        "roleArn": "arn:aws:iam::111111111111:role/JohnRole2",
        "roleSessionName": "Role2WithTags",
        "sourceIdentity": "JohnDoe",
        "transitiveTagKeys": [
            "Email",
            "CostCenter"
        ],
        "durationSeconds": 3600
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
            "expiration": "Oct 2, 2019, 11:12:29 PM",
            "sessionToken": "AgoJb3JpZ2luX2VjEB4aCXVzLXdlc3QtMSJHMEXAMPLETOKEN
+//rJb8Lo30mFc5MlhFCEbubZvEj0wHB/mDMwIgSEe9gk/Zjr09tZV7F1HDTMhmEXAMPLETOKEN/iEJ/
rkqngII9//////////
ARABGgw0MjgzMDc4NjM5NjYiDLZjZFKwP4qxQG5sFCryASO4UPz5qE97wPPH1eLMvs7CgSDBSWfonmRTCfokm2FN1+hWUdQ
+C+WKFZb701eiv9J5La2EXAMPLETOKEN/c7S5Iro1WUJ0q3Cxuo/8HUoSxVhQHM7zF7mWWLhXLEQ52ivL
+F6q5dpXu4aTFedpMfnJa8JtkWwG9x1Axj0Ypy2ok8v5unpQGWych1vwdvj6ez1Dm8Xg1+qIzXILiEXAMPLETOKEN/
vQGqu8H+nxp3kabcrtOvTFTvxX6vsc8OGwUfHhzAfYGEXAMPLETOKEN/
L6v1yMM3B1OwFOrQBno1HEjf1oNI8RnQiMNFdUOtwYj7HUZIOCZmjfN8PPHq77N7GJl9lzvIZKQA0Owcjg
+mc78zHCj8y0siY8C96paEXAMPLETOKEN/
E3cpksxWdgs91HRzJWScjN2+r2LTGjYhyPqcmFzzo2mCE7mBNEXAMPLETOKEN/oJy
+2o83YNW5tOiDmczgDzJZ4UKR84yGYOMfSnF4XcEJrDgAJ3OJFwmTcTQICAlSwLEXAMPLETOKEN"
        },
        "assumedRoleUser": {
            "assumedRoleId": "AROAIFR7WHDTSOYQYHFUE:Role2WithTags",
            "arn": "arn:aws:sts::111111111111:assumed-role/test-role/Role2WithTags"
        },
    "sourceIdentity": "JohnDoe"
    },
    "requestID": "b96b0e4e-e561-11e9-8b3f-7b396EXAMPLE",
    "eventID": "1917948f-3042-46ec-98e2-62865EXAMPLE",
    "resources": [
```

```
        {
            "ARN": "arn:aws:iam::111111111111:role/JohnRole2",
            "accountId": "111111111111",
            "type": "AWS::IAM::Role"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
}
```

CloudTrail 로그 파일의 AWS 서비스 AWS STS API 이벤트의 예

다음은 서비스 역할의 권한을 사용하여 다른 서비스 API를 호출하는 AWS 서비스의 요청에 대한 CloudTrail 로그 항목 예제입니다. 777788889999 계정에서 만든 요청에 대한 CloudTrail 로그 항목을 보여줍니다.

```
{
  "eventVersion": "1.04",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAQRSTUVWXYZEXAMPLE:devdsk",
    "arn": "arn:aws:sts::777788889999:assumed-role/AssumeNothing/devdsk",
    "accountId": "777788889999",
    "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2016-11-14T17:25:26Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAQRSTUVWXYZEXAMPLE",
        "arn": "arn:aws:iam::777788889999:role/AssumeNothing",
        "accountId": "777788889999",
        "userName": "AssumeNothing"
      }
    }
  },
  "eventTime": "2016-11-14T17:25:45Z",
  "eventSource": "s3.amazonaws.com",
  "eventName": "DeleteBucket",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.1",
```

```
  "userAgent": "[aws-cli/1.11.10 Python/2.7.8
 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67]",
  "requestParameters": {
    "bucketName": "my-test-bucket-cross-account"
  },
  "responseElements": null,
  "requestID": "EXAMPLE463D56D4C",
  "eventID": "dEXAMPLE-265a-41e0-9352-4401bEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "777788889999"
}
```

CloudTrail 로그 파일의 SAML AWS STS API 이벤트의 예

다음은 AWS STS AssumeRoleWithSAML 작업 요청에 대한 CloudTrail 로그 항목 예제입니다. 이 요청에는 SAML 어설션을 통해 세션 태그로 전달되는 SAML 속성 `CostCenter` 및 `Project`가 포함됩니다. 이러한 태그는 역할 체인 시나리오에서 지속되도록 전이적으로 설정됩니다. 요청에는 선택적 API 파라미터 `DurationSeconds`이(가) 포함되며 이는 CloudTrail 로그에서 `durationSeconds`(으)로 표시되며 `1800`초로 설정됩니다. 요청에는 SAML 어설션에서 전달되는 SAML 속성 `sourceIdentity`도 포함됩니다. 결과 역할 세션 자격 증명을 사용하여 다른 역할을 수임하는 경우 이 소스 자격 증명은 유지됩니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "SAMLUser",
        "principalId": "SampleUkh1i4+ExamplexL/jEvs=:SamlExample",
        "userName": "SamlExample",
        "identityProvider": "bdGOnTesti4+ExamplexL/jEvs="
    },
    "eventTime": "2023-08-28T18:30:58Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRoleWithSAML",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "aws-internal/3 aws-sdk-java/1.12.479
 Linux/5.10.186-157.751.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/17.0.7+11 java/17.0.7
 kotlin/1.3.72 vendor/Amazon.com_Inc. cfg/retry-mode/standard",
    "requestParameters": {
        "sAMLAssertionID": "_c0046cEXAMPLEb9d4b8eEXAMPLE2619aEXAMPLE",
        "roleSessionName": "MyAssignedRoleSessionName",
        "sourceIdentity": "MySAMLUser",
```

```
        "principalTags": {
            "CostCenter": "987654",
            "Project": "Unicorn",
            "Department": "Engineering"
        },
        "transitiveTagKeys": [
            "CostCenter",
            "Project"
        ],
        "roleArn": "arn:aws:iam::444455556666:role/SAMLTestRoleShibboleth",
        "principalArn": "arn:aws:iam::444455556666:saml-provider/Shibboleth",
        "durationSeconds": 1800
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
            "sessionToken": "<encoded session token blob>",
            "expiration": "Aug 28, 2023, 7:00:58 PM"
        },
        "assumedRoleUser": {
            "assumedRoleId": "AROAD35QRSTUVWEXAMPLE:MyAssignedRoleSessionName",
            "arn": "arn:aws:sts::444455556666:assumed-role/SAMLTestRoleShibboleth/
MyAssignedRoleSessionName"
        },
        "packedPolicySize": 1,
        "subject": "SamlExample",
        "subjectType": "transient",
        "issuer": "https://server.example.com/idp/shibboleth",
        "audience": "https://signin.aws.amazon.com/saml",
        "nameQualifier": "bdGOnTesti4+ExamplexL/jEvs=",
        "sourceIdentity": "MySAMLUser"
    },
    "requestID": "6EXAMPLE-e595-11e5-b2c7-c974fEXAMPLE",
    "eventID": "dEXAMPLE-265a-41e0-9352-4401bEXAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "444455556666",
            "type": "AWS::IAM::Role",
            "ARN": "arn:aws:iam::444455556666:role/SAMLTestRoleShibboleth"
        },
        {
            "accountId": "444455556666",
            "type": "AWS::IAM::SAMLProvider",
```

```
            "ARN": "arn:aws:iam::444455556666:saml-provider/test-saml-provider"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "444455556666",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "sts.us-east-2.amazonaws.com"
    }
}
```

CloudTrail 로그 파일의 예시 OIDC AWS STS API 이벤트

다음은 AWS STS AssumeRoleWithWebIdentity 작업 요청에 대한 CloudTrail 로그 항목 예제입니다. 이 요청에는 OpenID Connect(OIDC) ID 제공업체(IdP) 토큰을 통해 세션 태그로 전달되는 속성 `CostCenter` 및 `Project`가 포함됩니다. 이러한 태그는 역할 체인 시나리오에서 지속되도록 전이적으로 설정됩니다. 요청에는 자격 증명 공급자 토큰의 `sourceIdentity` 특성이 포함됩니다. 결과 역할 세션 자격 증명을 사용하여 다른 역할을 수임하는 경우 이 소스 자격 증명은 유지됩니다.

CloudTrail 로그 항목에는 `identityProviderConnectionVerificationMethod` 속성이 있는 `additionalEventData` 필드도 포함되어 있습니다. 이 속성은 AWS에서 OIDC 공급자와의 연결을 확인하는 데 사용하는 방법을 나타냅니다. 속성 값은 `IAMTrustStore` 또는 `Thumbprint`입니다. `IAMTrustStore` 값은 AWS에서 신뢰할 수 있는 루트 인증 기관(CA) 라이브러리를 사용하여 OIDC IdP와의 연결을 성공적으로 확인했음을 나타냅니다. `Thumbprint` 값은 AWS에서 IdP 구성에 설정된 인증서 지문을 사용하여 OIDC IdP 서버 인증서를 확인했음을 나타냅니다.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "WebIdentityUser",
    "principalId": "arn:aws:iam::444455556666:oidc-provider/<issuer url of OIDC
 provider>:<id of application>:<id of user>",
    "userName": "<id of user>",
    "identityProvider": "arn:aws:iam::444455556666:oidc-provider/<issuer url of OIDC
 provider>"
  },
  "eventTime": "2024-07-09T15:41:37Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRoleWithWebIdentity",
```

```
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/2.13.29 Python/3.11.6 Windows/10 exe/AMD64 prompt/off command/
sts.assume-role-with-web-identity",
  "requestParameters": {
    "roleArn": "arn:aws:iam::444455556666:role/FederatedWebIdentityRole",
    "roleSessionName": "<assigned role session name>",
    "sourceIdentity": "MyWebIdentityUser",
    "durationSeconds": 3600,
    "principalTags": {
      "CostCenter": "24680",
      "Project": "Pegasus"
    },
    "transitiveTagKeys": [
      "CostCenter",
      "Project"
    ]
  },
  "responseElements": {
    "credentials": {
      "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
      "sessionToken": "<encoded session token blob>",
      "expiration": "Jul 9, 2024, 4:41:37 PM"
    },
    "subjectFromWebIdentityToken": "<id of user>",
    "sourceIdentity": "MyWebIdentityUser",
    "assumedRoleUser": {
      "assumedRoleId": "AROA123456789EXAMPLE:<assigned role session name>",
      "arn": "arn:aws:sts::444455556666:assumed-role/FederatedWebIdentityRole/<assigned
 role session name>"
    },
    "provider": "arn:aws:iam::444455556666:oidc-provider/<issuer url of OIDC
 provider>",
    "audience": "<id of application>"
  },
  "additionalEventData": {
    "identityProviderConnectionVerificationMethod": "IAMTrustStore"
  },
  "requestID": "aEXAMPLE-0b26-40df-8973-c7012EXAMPLE",
  "eventID": "aEXAMPLE-ee29-4ac0-a0ed-3f5c5EXAMPLE",
  "readOnly": true,
  "resources": [
    {
      "accountId": "444455556666",
```

```
      "type": "AWS::IAM::Role",
      "ARN": "arn:aws:iam::444455556666:role/FederatedWebIdentityRole"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "444455556666",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "sts.us-east-2.amazonaws.com"
  }
}
```

CloudTrail 로그의 로그인 이벤트의 예

CloudTrail 로그 파일에는 JSON 형식의 이벤트 정보가 포함되어 있습니다. 로그인 이벤트는 단일 로그인 요청을 나타내며 로그인 보안 주체, 리전, 작업 날짜 및 시간에 대한 정보를 포함합니다.

CloudTrail 로그 파일의 로그인 성공 이벤트의 예

다음은 성공한 로그인 이벤트에 대한 CloudTrail 로그 항목을 나타낸 예제입니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn":"arn:aws:iam::111122223333:user/JohnDoe",
    "accountId": "111122223333",
    "userName": "JohnDoe"
  },
  "eventTime": "2014-07-16T15:49:27Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.110",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101
 Firefox/24.0",
  "requestParameters": null,
  "responseElements": {
    "ConsoleLogin": "Success"
```

```
  },
  "additionalEventData": {
    "MobileVersion": "No",
    "LoginTo": "https://console.aws.amazon.com/s3/  ",
    "MFAUsed": "No"
  },
  "eventID": "3fcfb182-98f8-4744-bd45-10a395ab61cb"
}
```

CloudTrail 로그 파일에 저장된 정보에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 이벤트 참조를 참조하세요.

CloudTrail 로그 파일의 로그인 실패 이벤트의 예

다음은 실패한 로그인 이벤트에 대한 CloudTrail 로그 항목을 나타낸 예제입니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn":"arn:aws:iam::111122223333:user/JaneDoe",
    "accountId": "111122223333",
    "userName": "JaneDoe"
  },
  "eventTime": "2014-07-08T17:35:27Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.100",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101
 Firefox/24.0",
  "errorMessage": "Failed authentication",
  "requestParameters": null,
  "responseElements": {
    "ConsoleLogin": "Failure"
  },
  "additionalEventData": {
    "MobileVersion": "No",
    "LoginTo": "https://console.aws.amazon.com/sns",
    "MFAUsed": "No"
  },
  "eventID": "11ea990b-4678-4bcd-8fbe-62509088b7cf"
```

```
}
```

이 정보에서 userIdentity 요소에도 나와 있듯이 JaneDoe라는 이름의 IAM 사용자가 로그인을 시도한 것을 알 수 있습니다. 또한 responseElements 요소를 보면 로그인 시도가 실패한 것도 확인됩니다. 그리고 JaneDoe가 Amazon SNS 콘솔에 로그인하려고 시도한 일시는 2014년 7월 8일 오후 5시 35분(UTC)입니다.

잘못된 사용자 이름으로 인한 로그인 실패 이벤트의 예

다음은 잘못된 사용자 이름을 입력하여 로그인을 실패한 이벤트의 CloudTrail 로그 항목을 나타낸 예제입니다. 이때 AWS는 userName 텍스트를 HIDDEN_DUE_TO_SECURITY_REASONS로 마스킹 처리하여 잠재적으로 민감한 정보의 노출을 차단합니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "accountId": "123456789012",
    "accessKeyId": "",
    "userName": "HIDDEN_DUE_TO_SECURITY_REASONS"
  },
  "eventTime": "2015-03-31T22:20:42Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101
 Firefox/24.0",
  "errorMessage": "No username found in supplied account",
  "requestParameters": null,
  "responseElements": {
    "ConsoleLogin": "Failure"
  },
  "additionalEventData": {
    "LoginTo": "https://console.aws.amazon.com/console/home?state=hashArgs
%23&isauthcode=true",
    "MobileVersion": "No",
    "MFAUsed": "No"
  },
  "eventID": "a7654656-0417-45c6-9386-ea8231385051",
  "eventType": "AwsConsoleSignin",
  "recipientAccountId": "123456789012"
```

```
}
```

IAM 역할 신뢰 정책 동작

2022년 9월 21일, AWS는 역할 신뢰 정책을 명시적으로 허용하도록 IAM 역할 신뢰 정책 동작을 변경했습니다. 레거시 동작 허용 목록의 IAM 역할에는 `AssumeRole` 이벤트에 대한 ExplicitTrustGrant를 위한 additionalEventData 필드가 있습니다. 레거시 허용 목록에 있는 역할이 레거시 동작을 사용하여 자기 자신의 역할을 수행할 때 `explicitTrustGrant`의 값은 false입니다. 레거시 허용 목록에 있는 역할이 자기 자신의 역할을 수행하지만 역할 신뢰 정책 동작이 이를 명시적으로 허용하도록 업데이트된 경우 `explicitTrustGrant`의 값은 true입니다.

레거시 동작의 허용 목록에는 포함된 IAM 역할은 매우 제한적이며, 자기 자신의 역할을 맡을 때만 이 필드가 CloudTrail 로그에 표시됩니다. 대부분의 경우 IAM 역할이 자기 자신의 역할을 수행할 필요는 없습니다. AWS는 프로세스, 코드 또는 구성을 업데이트하여 이러한 동작을 제거하거나 이러한 동작을 명시적으로 허용하도록 역할 신뢰 정책을 업데이트하는 것을 권장합니다. 자세한 내용은 IAM 역할 신뢰 정책 동작에 대한 업데이트 발표를 참조하세요.

AWS Identity and Access Management의 규정 준수 확인

서드 파티 감사자는 여러 AWS 규정 준수 프로그램의 일환으로 AWS Identity and Access Management(IAM)의 보안 및 규정 준수 상태를 평가합니다. 여기에는 SOC, PCI, FedRAMP, ISO 등이 포함됩니다.

AWS 서비스(이)가 특정 규정 준수 프로그램의 범위에 포함되는지 알아보려면 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반적인 정보는 AWS 규정 준수 프로그램을 참조하십시오.

AWS Artifact(을)를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드를 참조하십시오.

AWS 서비스 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다.AWS에서는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 가이드 – 이 배포 가이드에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수에 중점을 둔 기본 AWS 환경을 배포하기 위한 단계를 제공합니다.
- Amazon Web Service에서 HIPAA 보안 및 규정 준수 기술 백서 설계 - 이 백서는 기업이 AWS를 사용하여 HIPAA에 적합한 애플리케이션을 만드는 방법에 대해 설명합니다.

> **Note**
>
> 모든 AWS 서비스에 HIPAA 자격이 있는 것은 아닙니다. 자세한 내용은 HIPAA 적격 서비스 참조를 참조하십시오.

- AWS 규정 준수 리소스 - 고객 조직이 속한 산업 및 위치에 적용될 수 있는 워크북 및 가이드 컬렉션입니다.
- AWS 고객 규정 준수 가이드 - 규정 준수의 관점에서 공동 책임 모델을 이해합니다. 이 가이드에서는 AWS 서비스를 보호하기 위한 모범 사례를 요약하고 여러 프레임워크(미국 표준 기술 연구소(NIST), 결제 카드 산업 보안 표준 위원회(PCI), 국제 표준화기구(ISO) 등)에서 보안 제어에 대한 지침을 매핑합니다.
- AWS Config 개발자 가이드의 규칙을 사용하여 리소스 평가 – AWS Config 서비스는 내부 사례, 산업 지침 및 규제에 대한 리소스 구성의 준수 상태를 평가합니다.
- AWS Security Hub – 이 AWS 서비스(은)는 AWS 내의 보안 상태에 대한 포괄적인 보기를 제공합니다. Security Hub는 보안 제어를 사용하여 AWS리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하십시오.
- Amazon GuardDuty - 이 AWS 서비스는 의심스럽고 악의적인 활동이 있는지 환경을 모니터링하여 AWS 계정, 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 탐지합니다. GuardDuty는 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양한 규정 준수 요구 사항을 따르는 데 도움을 줄 수 있습니다.
- AWS Audit Manager - 이 AWS 서비스은(는) AWS 사용을 지속적으로 감사하여 리스크를 관리하고 규정 및 업계 표준을 준수하는 방법을 간소화할 수 있도록 지원합니다.

AWS Identity and Access Management의 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전에는 물리적으로 분리되고 격리된 다수의 가용 영역이 있으며 이러한 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하십시오.

AWS Identity and Access Management(IAM)와 AWS Security Token Service(AWS STS)는 전 세계에서 사용할 수 있는 자체 유지형 리전 기반 서비스입니다.

IAM은 중요한 AWS 서비스입니다. AWS에서 수행되는 모든 작업은 IAM의 인증과 승인을 받아야 합니다. IAM은 IAM에 저장된 아이덴티티와 정책에 대해 각 요청을 확인하여 요청이 허용되는지 아니면 거

부되는지 결정합니다. IAM은 예상치 못한 장애 발생 시에도 서비스가 인증되도록 별도의 컨트롤 플레인과 데이터 영역으로 설계되었습니다. 역할 및 정책과 같이 권한 부여에 사용되는 IAM 리소스는 컨트롤 플레인에 저장됩니다. IAM 고객은 `DeletePolicy` 및 `AttachRolePolicy`와 같은 IAM 작업을 사용하여 이러한 리소스의 구성을 변경할 수 있습니다. 이러한 구성 변경 요청은 컨트롤 플레인으로 이동합니다. 미국 동부(버지니아 북부) 리전에 위치한 모든 상용 AWS 리전에 대해 하나의 IAM 컨트롤 플레인이 있습니다. 그런 다음 IAM 시스템은 모든 활성화된 AWS 리전의 IAM 데이터 영역에 구성 변경 사항을 전파합니다. IAM 데이터 영역은 기본적으로 IAM 컨트롤 플레인 구성 데이터의 읽기 전용 복제본입니다. 각 AWS 리전에는 동일한 리전의 요청에 대한 인증 및 권한 부여를 수행하는 IAM 데이터 영역의 완전히 독립적인 인스턴스가 있습니다. 각 리전에서 IAM 데이터 영역은 최소 3개의 가용 영역에 분산되어 있으며 고객 장애 없이 가용 영역의 손실을 견딜 수 있는 충분한 용량을 갖추고 있습니다. IAM 컨트롤 플레인과 데이터 영역 모두 계획된 가동 중지가 없도록 구축되었으며 모든 소프트웨어 업데이트 및 확장 작업은 고객에게 보이지 않는 방식으로 수행됩니다.

AWS STS기본적으로 요청은 항상 단일 글로벌 엔드포인트로 이동합니다. 리전 AWS STS 엔드포인트를 사용하여 지연 시간을 줄이거나 애플리케이션에 추가 중복성을 제공할 수 있습니다. 자세한 내용은 AWS STS에서 AWS 리전 관리 단원을 참조하십시오.

특정 이벤트로 인해 네트워크를 통한 AWS 리전 간 통신이 중단될 수 있습니다. 그러나 글로벌 IAM 엔드포인트와 통신할 수 없는 경우에도 AWS STS는 여전히 IAM 보안 주체를 인증할 수 있으며 IAM은 요청을 승인할 수 있습니다. 통신을 중단하는 이벤트의 특정 세부 정보에 따라 AWS 서비스에 대한 액세스 가능성이 결정됩니다. 대부분의 경우 AWS 환경에서 IAM 자격 증명을 계속 사용할 수 있습니다. 통신을 중단하는 이벤트에는 다음과 같은 조건이 적용될 수 있습니다.

IAM 사용자의 액세스 키

IAM 사용자를 위한 장기 액세스 키를 사용하여 리전에서 무기한 인증할 수 있습니다. AWS Command Line Interface와 API를 사용할 때 AWS가 프로그래밍 방식의 요청에서 아이덴티티를 확인할 수 있도록 AWS 액세스 키를 제공할 수 있습니다.

> **Important**
>
> 모범 사례로서 사용자가 장기 액세스 키 대신 임시 보안 인증을 사용하여 로그인하는 것이 좋습니다.

임시 자격 증명

최소 24시간 동안 AWS STS 리전 서비스 엔드포인트를 사용하여 새 임시 자격 증명을 요청할 수 있습니다. 다음 API 작업은 임시 자격 증명을 생성합니다.

- AssumeRole
- AssumeRoleWithWebIdentity
- AssumeRoleWithSAML
- GetFederationToken
- GetSessionToken

보안 주체 및 권한

- IAM에서 보안 주체 또는 권한을 추가, 수정 또는 제거하지 못할 수 있습니다.
- 자격 증명에 최근에 IAM에서 적용한 권한 변경 사항이 반영되지 않을 수 있습니다. 자세한 내용은 변경 사항이 매번 즉시 표시되는 것은 아닙니다 섹션을 참조하세요.

AWS Management Console

- 리전 로그인 엔드포인트를 사용하여 IAM 사용자로 AWS Management Console에 로그인할 수 있습니다. 리전 로그인 엔드포인트의 URL 형식은 다음과 같습니다.

  `https://{Account ID}.signin.aws.amazon.com/console?region={Region}`

  예: https://111122223333.signin.aws.amazon.com/console?region=us-west-2
- Universal 2nd Factor(U2F) 다중 인증(MFA)을 완료하지 못할 수 있습니다.

IAM 복원성 모범 사례

AWS는 AWS 리전과 가용 영역에 탄력성을 더했습니다. 환경과 상호 작용하는 시스템에서 다음과 같은 IAM 모범 사례가 관찰되면 해당 복원력을 활용할 수 있습니다.

1. 기본 글로벌 엔드포인트 대신 AWS STS 리전 서비스 엔드포인트를 사용합니다.
2. IAM 리소스를 자주 생성하거나 수정하는 필수 리소스에 대한 환경 구성을 검토하고 기존 IAM 리소스를 사용하는 대체 솔루션을 준비합니다.

AWS Identity and Access Management에서 인프라 보안

관리형 서비스인 AWS Identity and Access Management는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스와 AWS의 인프라 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하십시오. 인프라 보안에 대한 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS에서 게시한 API 호출을 사용하여 네트워크를 통해 IAM에 액세스합니다. 고객은 다음을 지원해야 합니다.

- Transport Layer Security(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

서비스로 직접 HTTPS 요청을 실행할 수 있는 IAM HTTPS API를 사용하여 프로그래밍 방식으로 IAM에 액세스할 수 있습니다. 쿼리 API는 보안 자격 증명을 비롯한 민감한 정보를 반환합니다. 따라서 모든 API 요청과 함께 HTTPS를 사용해야 합니다. HTTPS API를 사용할 때는 자격 증명을 사용하여 요청에 디지털 방식으로 서명하는 코드를 포함해야 합니다.

이러한 API 작업은 어떤 네트워크 위치에서든 호출할 수 있지만, IAM은 소스 IP 주소에 따른 제한 사항을 포함할 수 있는 리소스 기반 액세스 정책을 지원합니다. IAM 정책을 사용하여 특정 Amazon Virtual Private Cloud(Amazon VPC) 엔드포인트 또는 특정 VPC에서 액세스를 제어할 수도 있습니다. 그러면 AWS 네트워크의 특정 VPC에서만 특정 IAM 리소스에 대한 네트워크 액세스가 효과적으로 격리됩니다.

AWS Identity and Access Management의 구성 및 취약성 분석

AWS가 게스트 운영 체제(OS), 데이터베이스 패치, 방화벽 구성, 재해 복구 등의 기본 보안 작업을 처리합니다. 적합한 제3자가 이 절차를 검토하고 인증하였습니다. 자세한 내용은 다음 리소스를 참조하세요.

- 공동 책임 모델
- Amazon Web Services: 보안 프로세스의 개요(백서)

다음 리소스는 AWS Identity and Access Management(IAM)의 구성 및 취약성 분석도 해결합니다.

- AWS Identity and Access Management의 규정 준수 확인
- AWS Identity and Access Management의 보안 모범 사례 및 사용 사례

AWS Identity and Access Management Access Analyzer에 대한 AWS 관리형 정책

AWS 관리형 정책은 AWS에 의해 생성되고 관리되는 독립 실행형 정책입니다. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있기 때문에 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에서 정의한 권한은 변경할 수 없습니다. 만약 AWS가 AWS 관리형 정책에 정의된 권한을 업데이트할 경우 정책이 연결되어 있는 모든 보안 주체 엔터티(사용자, 그룹 및 역할)에도 업데이트가 적용됩니다. 새로운 AWS 서비스를 시작하거나 새로운 API 작업을 기존 서비스에 이용하는 경우 AWS가 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하십시오.

IAMReadOnlyAccess

IAM 리소스에 대한 읽기 전용 액세스를 허용하려면 `IAMReadOnlyAccess` 관리형 정책을 사용합니다. 이 정책은 모든 IAM 리소스를 가져오고 나열할 수 있는 권한을 부여합니다. 이를 통해 사용자, 그룹, 역할, 정책, 자격 증명 공급자 및 MFA 디바이스에 대한 세부 정보 및 활동 보고서를 볼 수 있습니다. 리소스를 생성 또는 삭제하거나 IAM Access Analyzer 리소스에 대한 액세스 권한은 포함되지 않습니다. 이 정책이 지원하는 서비스 및 작업의 전체 목록에 대한 정책을 봅니다.

IAMUserChangePassword

IAM 사용자가 자신의 암호를 변경하도록 허용하려면 이 `IAMUserChangePassword` 관리형 정책을 사용합니다.

IAM Account settings(계정 설정) 및 Password policy(암호 정책)을 구성하여 IAM 사용자가 IAM 계정 암호를 변경할 수 있도록 허용합니다. 이 작업을 허용하면 IAM은 각 사용자에게 다음 정책을 연결합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:ChangePassword"
            ],
            "Resource": [
                "arn:aws:iam::*:user/${aws:username}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

IAMAccessAnalyzerFullAccess

`IAMAccessAnalyzerFullAccess` AWS관리형 정책을 사용하여 관리자가 IAM Access Analyzer에 액세스할 수 있도록 합니다.

권한 그룹화

이 정책은 제공된 권한에 따라 명령문으로 그룹화됩니다.

- IAM Access Analyzer - IAM Access Analyzer의 모든 리소스에 대한 전체 관리 권한을 허용합니다.
- 서비스 연결 역할 생성 – 관리자가 서비스 연결 역할을 생성하도록 허용하여 IAM Access Analyzer를 통해 사용자를 대신하여 다른 서비스의 리소스를 분석할 수 있습니다. 이 권한은 IAM Access Analyzer에서만 사용 가능한 서비스 연결 역할만 생성할 수 있습니다.
- AWS Organizations - 관리자가 IAM Access Analyzer를 AWS Organizations의 조직에 대해 사용하도록 허용합니다. AWS Organizations의 IAM Access Analyzer에 신뢰할 수 있는 액세스를 활성화한 후 관리 계정의 멤버는 조직 전체의 결과를 볼 수 있습니다.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "access-analyzer:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "access-analyzer.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization",
        "organizations:DescribeOrganizationalUnit",
        "organizations:ListAccounts",
        "organizations:ListAccountsForParent",
        "organizations:ListAWSServiceAccessForOrganization",
        "organizations:ListChildren",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:ListParents",
        "organizations:ListRoots"
      ],
      "Resource": "*"
    }
  ]
}
```

IAMAccessAnalyzerReadOnlyAccess

IAM Access Analyzer에 대한 읽기 전용 액세스를 허용하려면 `IAMAccessAnalyzerReadOnlyAccess` AWS 관리형 정책을 사용합니다.

AWS Organizations의 IAM Access Analyzer에 대한 읽기 전용 액세스를 허용하려면 IAMAccessAnalyzerFullAccess AWS관리형 정책에서 설명 및 나열 작업을 허용하는 고객 관리형 정책을 생성합니다.

서비스 수준 권한

이 정책은 IAM Access Analyzer에 대한 읽기 전용 액세스를 제공합니다. 이 정책에 다른 서비스 권한은 포함되지 않습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMAccessAnalyzerReadOnlyAccess",
      "Effect": "Allow",
      "Action": [
        "access-analyzer:CheckAccessNotGranted",
        "access-analyzer:CheckNoNewAccess",
        "access-analyzer:Get*",
        "access-analyzer:List*",
        "access-analyzer:ValidatePolicy"
      ],
      "Resource": "*"
    }
  ]
}
```

AccessAnalyzerServiceRolePolicy

IAM 엔터티에 AccessAnalyzerServiceRolePolicy를 연결할 수 없습니다. 이 정책은 IAM 액세스 분석기에서 사용자를 대신하여 작업을 수행할 수 있도록 서비스 연결 역할에 연결됩니다. 자세한 내용은 AWS Identity and Access Management Access Analyzer에 서비스 연결 역할 사용을 참조하세요.

권한 그룹화

이 정책을 통해 IAM Access Analyzer에 액세스하여 여러 AWS 서비스에서 리소스 메타데이터를 분석합니다.

- Amazon DynamoDB – DynamoDB 스트림과 테이블을 볼 수 있는 권한을 허용합니다.
- Amazon Elastic Compute Cloud – IP 주소, 스냅샷, VPC를 설명할 수 있는 권한을 허용합니다.

- Amazon Elastic Container Registry – 이미지 리포지토리를 설명하고 리포지토리 정책을 검색할 수 있는 권한을 허용합니다.
- Amazon Elastic File System – Amazon EFS 파일 시스템 설명을 확인하고 Amazon EFS 파일 시스템의 리소스 수준 정책을 확인할 수 있는 권한을 허용합니다.
- AWS Identity and Access Management – 지정된 역할에 대한 정보를 검색하고 지정된 경로 접두사가 있는 IAM 역할을 나열할 수 있는 권한을 허용합니다. 사용자, 사용자 그룹, 로그인 프로필, 액세스 키 및 서비스에서 마지막으로 액세스한 데이터에 대한 정보를 검색할 수 있는 권한을 허용합니다.
- AWS Key Management Service – KMS 키와 키 정책, 부여된 권한에 대한 자세한 정보를 볼 수 있는 권한을 허용합니다.
- AWS Lambda – Lambda 별칭, 함수, 레이어 및 별칭(들)에 대한 정보를 볼 수 있는 권한을 허용합니다.
- AWS Organizations - Organizations에 대한 권한을 허용하고 신뢰 영역인 AWS 조직 내에서 분석기 생성을 허용합니다.
- Amazon Relational Database Service – Amazon RDS DB 스냅샷 및 Amazon RDS DB 클러스터 스냅샷에 대한 자세한 정보를 볼 수 있는 권한을 허용합니다.
- Amazon Simple Storage Service – Amazon S3 Express One 스토리지 클래스를 사용하는 Amazon S3 액세스 포인트, 버킷 및 Amazon S3 디렉터리 버킷에 대한 자세한 정보를 볼 수 있는 권한을 허용합니다.
- AWS Secrets Manager – 보안 암호와 보안 암호와 연결된 리소스 정책에 대한 자세한 정보를 볼 수 있는 권한을 허용합니다.
- Amazon 단순 알림 서비스 — 주제에 대한 세부 정보를 볼 수 있는 권한을 허용합니다.
- Amazon Simple Queue Service – 지정된 대기열에 대한 자세한 정보를 볼 수 있는 권한을 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AccessAnalyzerServiceRolePolicy",
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetResourcePolicy",
        "dynamodb:ListStreams",
        "dynamodb:ListTables",
        "ec2:DescribeAddresses",
        "ec2:DescribeByoipCidrs",
```

```
        "ec2:DescribeSnapshotAttribute",
        "ec2:DescribeSnapshots",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs",
        "ec2:GetSnapshotBlockPublicAccessState",
        "ecr:DescribeRepositories",
        "ecr:GetRepositoryPolicy",
        "elasticfilesystem:DescribeFileSystemPolicy",
        "elasticfilesystem:DescribeFileSystems",
        "iam:GetRole",
        "iam:ListEntitiesForPolicy",
        "iam:ListRoles",
        "iam:ListUsers",
        "iam:GetUser",
        "iam:GetGroup",
        "iam:GenerateServiceLastAccessedDetails",
        "iam:GetServiceLastAccessedDetails",
        "iam:ListAccessKeys",
        "iam:GetLoginProfile",
        "iam:GetAccessKeyLastUsed",
        "iam:ListRolePolicies",
        "iam:GetRolePolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListUserPolicies",
        "iam:GetUserPolicy",
        "iam:ListAttachedUserPolicies",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:ListGroupsForUser",
        "kms:DescribeKey",
        "kms:GetKeyPolicy",
        "kms:ListGrants",
        "kms:ListKeyPolicies",
        "kms:ListKeys",
        "lambda:GetFunctionUrlConfig",
        "lambda:GetLayerVersionPolicy",
        "lambda:GetPolicy",
        "lambda:ListAliases",
        "lambda:ListFunctions",
        "lambda:ListLayers",
        "lambda:ListLayerVersions",
        "lambda:ListVersionsByFunction",
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization",
```

```
        "organizations:DescribeOrganizationalUnit",
        "organizations:ListAccounts",
        "organizations:ListAccountsForParent",
        "organizations:ListAWSServiceAccessForOrganization",
        "organizations:ListChildren",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:ListParents",
        "organizations:ListRoots",
        "rds:DescribeDBClusterSnapshotAttributes",
        "rds:DescribeDBClusterSnapshots",
        "rds:DescribeDBSnapshotAttributes",
        "rds:DescribeDBSnapshots",
        "s3:DescribeMultiRegionAccessPointOperation",
        "s3:GetAccessPoint",
        "s3:GetAccessPointPolicy",
        "s3:GetAccessPointPolicyStatus",
        "s3:GetAccountPublicAccessBlock",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation",
        "s3:GetBucketPolicyStatus",
        "s3:GetBucketPolicy",
        "s3:GetBucketPublicAccessBlock",
        "s3:GetMultiRegionAccessPoint",
        "s3:GetMultiRegionAccessPointPolicy",
        "s3:GetMultiRegionAccessPointPolicyStatus",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets",
        "s3:ListMultiRegionAccessPoints",
        "s3express:GetBucketPolicy",
        "s3express:ListAllMyDirectoryBuckets",
        "sns:GetTopicAttributes",
        "sns:ListTopics",
        "secretsmanager:DescribeSecret",
        "secretsmanager:GetResourcePolicy",
        "secretsmanager:ListSecrets",
        "sqs:GetQueueAttributes",
        "sqs:ListQueues"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS 관리형 정책에 대한 IAM 및 IAM Access Analyzer 업데이트

서비스가 이러한 변경 내용을 추적하기 시작한 이후부터 IAM 및 AWS 관리형 정책 업데이트에 대한 세부 정보를 확인합니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 IAM 및 IAM Access Analyzer 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
AccessAnalyzerServiceRolePolicy - 추가된 권한	IAM Access Analyzer가 `AccessAnalyzerServiceRolePolicy` 의 서비스 수준 권한에 IAM 사용자 및 역할 정책에 대한 정보를 검색할 수 있는 권한에 대한 지원을 추가했습니다.	2024년 5월 30일
AccessAnalyzerServiceRolePolicy - 추가된 권한	IAM Access Analyzer가 `AccessAnalyzerServiceRolePolicy` 의 서비스 수준 권한에 Amazon EC2 스냅샷에 대한 퍼블릭 액세스 차단의 현재 상태를 검색할 수 있는 권한에 대한 지원을 추가했습니다.	2024년 1월 23일
AccessAnalyzerServiceRolePolicy - 추가된 권한	IAM Access Analyzer가 `AccessAnalyzerServiceRolePolicy` 의 서비스 수준 권한에 DynamoDB 스트림과 테이블에 대한 지원을 추가했습니다.	2024년 1월 11일
AccessAnalyzerServiceRolePolicy - 추가된 권한	IAM Access Analyzer가 `AccessAnalyzerServiceRolePolicy` 의 서비스	2023년 12월 1일

변경 사항	설명	날짜
	수준 권한에 Amazon S3 디렉터리 버킷에 대한 지원을 추가했습니다.	
IAMAccessAnalyzerReadOnlyAccess - 추가된 권한	IAM Access Analyzer에 정책 업데이트가 추가 액세스를 허용하는지 여부를 확인할 수 있도록 권한을 추가했습니다. 이 권한은 IAM Access Analyzer에서 정책에 대한 정책 확인을 수행하는 데 필요합니다.	2023년 11월 26일
AccessAnalyzerServiceRolePolicy - 추가된 권한	IAM Access Analyzer에 IAM 작업을 추가하여 `AccessAnalyzerServiceRolePolicy` 의 서비스 수준 권한에 대해 다음의 작업을 지원했습니다. • 정책 엔터티 나열 • 마지막으로 액세스한 서비스 세부 정보 생성 • 액세스 키 정보 나열	2023년 11월 26일

변경 사항	설명	날짜
AccessAnalyzerServiceRolePolicy - 추가된 권한	IAM Access Analyzer는 `AccessAnalyzerServiceRolePolicy` 의 서비스 수준 권한에 대해 다음의 리소스 유형에 대한 지원을 추가했습니다. • Amazon EBS 볼륨 스냅샷 • Amazon ECR 리포지토리 • Amazon EFS 파일 시스템 • Amazon RDS DB 스냅샷 • Amazon RDS DB 클러스터 스냅샷 • Amazon SNS 주제	2022년 10월 25일
AccessAnalyzerServiceRolePolicy - 추가된 권한	IAM Access Analyzer는 `lambda:GetFunctionUrlConfig` 작업을 `AccessAnalyzerServiceRolePolicy` 의 서비스 수준 권한에 추가했습니다.	2022년 4월 6일
AccessAnalyzerServiceRolePolicy - 추가된 권한	IAM 액세스 분석기가 다중 리전 액세스 포인트와 연결된 메타데이터를 분석하기 위해 새로운 Amazon S3 작업을 추가했습니다.	2021년 9월 2일

변경 사항	설명	날짜
IAMAccessAnalyzerReadOnlyAccess - 추가된 권한	IAM Access Analyzer에서 검증을 위해 정책 확인을 사용하도록 허용하는 `ValidatePolicy` 권한을 부여하는 새 작업을 추가했습니다. 이 권한은 IAM Access Analyzer에서 정책에 대한 정책 확인을 수행하는 데 필요합니다.	2021년 3월 16일
IAM Access Analyzer 변경 내용 추적 시작	IAM Access Analyzer가 AWS 관리형 정책에 대한 변경 내용 추적을 시작했습니다.	2021년 3월 1일

IAM 외부의 보안 기능

IAM을 사용하여 AWS Management Console을 사용한 작업, AWS 명령줄 도구 작업 또는 AWS SDK를 통한 서비스 API 작업에 대한 액세스를 제어할 수 있습니다. 일부 AWS 제품은 리소스 보안을 위한 다른 방법도 지원합니다. 다음 목록은 전체는 아니지만 몇 가지 예에 해당합니다.

Amazon EC2

Amazon Elastic Compute Cloud에서는 키 페어를 사용하거나(Linux 인스턴스의 경우) 사용자 이름 및 암호를 사용해(Microsoft Windows 인스턴스의 경우) 인스턴스에 로그인합니다.

자세한 내용은 다음 설명서를 참조하세요.

- Amazon EC2 사용 설명서의 Amazon EC2 Linux 인스턴스 시작하기
- Amazon EC2 사용 설명서의 Amazon EC2 Windows 인스턴스 시작하기

Amazon RDS

Amazon Relational Database Service에서는 데이터베이스와 연결되어 있는 사용자 이름 및 암호를 사용해 데이터베이스 엔진에 로그인합니다.

자세한 내용은 Amazon RDS 사용 설명서의 Amazon RDS 시작하기를 참조하세요.

Amazon EC2 및 Amazon RDS

Amazon EC2 와 Amazon RDS에서는 보안 그룹을 사용하여 인스턴스나 데이터베이스에 대한 트래픽을 제어합니다.

자세한 내용은 다음 설명서를 참조하세요.

- Amazon EC2 사용 설명서의 Linux 인스턴스용 Amazon EC2 보안 그룹
- Amazon EC2 사용 설명서의 Windows 인스턴스용 Amazon EC2 보안 그룹
- Amazon RDS 사용 설명서의 Amazon RDS 보안 그룹

WorkSpaces

Amazon WorkSpaces에서는 사용자 이름과 암호를 사용해 데스크톱에 로그인합니다.

자세한 내용은 Amazon WorkSpaces 관리자 안내서의 WorkSpaces로 시작하기를 참조하세요.

Amazon WorkDocs

Amazon WorkDocs에서는 사용자 이름과 암호를 사용해 로그인하여 공유 문서에 액세스합니다.

자세한 내용은 Amazon WorkDocs 관리자 안내서의 Amazon WorkDocs로 시작하기를 참조하세요.

위와 같은 액세스 제어 방법들은 IAM과 다릅니다. IAM을 사용하면 Amazon EC2 인스턴스를 생성 또는 종료하거나 새로운 WorkSpaces 데스크톱을 설정하는 등 AWS 제품을 관리하는 방법을 제어할 수 있습니다. 다시 말해서, IAM은 Amazon Web Services 요청을 통한 작업을 제어하는 데 효과적일 뿐만 아니라 AWS Management Console에 대한 액세스를 제어하는 데도 이상적입니다. 단, IAM은 운영 체제(Amazon EC2), 데이터베이스(Amazon RDS), 데스크톱(Amazon WorkSpaces) 또는 협업 사이트(Amazon WorkDocs)에 로그인하는 등의 작업에 대해서는 보안 관리를 지원하지 않습니다.

특정 AWS 제품을 이용해 작업할 때는 반드시 설명서를 읽고 해당 제품에 속한 모든 리소스의 보안 옵션을 살펴보시기 바랍니다.

AWS Identity and Access Management Access Analyzer 사용하기

AWS Identity and Access Management Access Analyzer에서 다음 기능이 제공됩니다.

- IAM Access Analyzer 외부 액세스 분석기를 사용하면 외부 엔터티와 공유되는 조직 및 계정의 리소스 식별할 수 있습니다.
- IAM Access Analyzer 미사용 액세스 분석기는 조직 및 계정 내 미사용 액세스 식별에 도움이 됩니다.
- IAM Access Analyzer는 정책 문법과 AWS 모범 사례를 기준으로 IAM 정책을 검증합니다.
- IAM Access Analyzer 사용자 지정 정책 확인을 통해 지정된 보안 표준에 따라 IAM 정책을 검증할 수 있습니다.
- IAM Access Analyzer는 AWS CloudTrail 로그의 액세스 활동을 기반으로 IAM 정책을 생성합니다.

외부 엔티티와 공유되는 리소스 식별

IAM Access Analyzer를 사용하면 외부 엔터티와 공유되는 조직 및 계정 내 리소스(예: Amazon S3 버킷 또는 IAM 역할)를 식별할 수 있습니다. 이렇게 하면 리소스 및 데이터에 대한 의도하지 않은 액세스라는 보안 위험을 식별할 수 있습니다. IAM Access Analyzer는 외부 보안 주체와 공유되는 리소스를 식별하기 위해 AWS 환경에서 리소스 기반 정책을 분석하는 로직 기반 추론을 사용합니다. 계정 외부에서 공유되는 리소스의 각 인스턴스에 대해 IAM Access Analyzer는 결과를 생성합니다. 결과에는 액세스에 대한 정보와 액세스 권한이 부여되는 외부 보안 주체에 대한 정보가 포함됩니다. 결과를 검토하여 액세스가 의도한 안전한 액세스인지 또는 액세스가 의도하지 않은 보안 위험인지 확인할 수 있습니다. 외부 엔터티와 공유되는 리소스를 식별하는 데 도움이 될 뿐 아니라 리소스 권한을 배포하기 전에 IAM Access Analyzer 검색 결과를 사용하여 정책이 리소스에 대한 퍼블릭 및 크로스 계정 액세스에 미치는 영향을 미리 볼 수 있습니다. 조사 결과는 시각적 요약 대시보드에 정리되어 있습니다. 대시보드는 공개 액세스 조사 결과와 교차 계정 액세스 조사 결과 간의 차이를 강조 표시하고 리소스 유형별로 조사 결과를 분류하여 제공합니다. 대시보드에 대한 자세한 내용은 IAM Access Analyzer 조사 결과 대시보드 보기 섹션을 참조하세요.

> **Note**
>
> 외부 엔터티는 다른 AWS 계정, 루트 사용자, IAM 사용자 또는 역할, 페더레이션 사용자, 익명 사용자 또는 필터 생성에 사용할 수 있는 다른 엔터티일 수 있습니다. 자세한 내용은 AWS JSON 정책 요소: 보안 주체를 참조하세요.

IAM Access Analyzer를 활성화할 때 전체 조직 또는 계정에 대한 분석기를 생성하게 됩니다. 선택한 조직 또는 계정을 분석기의 신뢰 영역이라고 합니다. 분석기는 신뢰 영역 내에서 지원되는 모든 리소스를 모니터링합니다. 신뢰 영역 내에 있는 보안 주체에 의한 리소스 액세스는 신뢰할 수 있는 것으로 간주됩니다. IAM Access Analyzer가 활성화되면 신뢰 영역에서 지원되는 모든 리소스에 적용되는 정책이 분석됩니다. 첫 번째 분석 후 IAM Access Analyzer는 정기적으로 이들 정책을 분석합니다. 새 정책을 추가하거나 기존 정책을 변경한 경우, IAM Access Analyzer는 약 30분 내에 새 정책 또는 업데이트된 정책을 분석합니다.

정책을 분석할 때 IAM Access Analyzer가 신뢰 영역 내에 없는 외부 보안 주체에 액세스 권한을 부여하는 정책을 식별하면 결과가 생성됩니다. 각 결과에는 리소스에 대한 세부 정보, 리소스에 대한 액세스 권한이 있는 외부 엔터티, 적절한 작업을 수행할 수 있도록 부여된 권한에 대한 세부 정보가 포함되어 있습니다. 결과에 포함된 세부 정보를 확인하여 리소스 액세스가 의도적인지, 아니면 확인해야 할 잠재적 위험 요소인지 확인할 수 있습니다. 리소스에 정책을 추가하거나 기존 정책을 업데이트할 때 IAM Access Analyzer가 정책을 분석합니다. 또한 IAM Access Analyzer는 주기적으로 모든 리소스 기반 정책을 분석합니다.

드문 경우지만 특정 조건에서 IAM Access Analyzer가 추가되거나 업데이트된 정책에 대한 알림을 받지 못해 생성된 결과가 지연될 수 있습니다. Amazon S3 버킷과 연결된 다중 리전 액세스 포인트를 생성 또는 삭제하거나 다중 리전 액세스 포인트에 대한 정책을 업데이트하는 경우 IAM Access Analyzer에서 검색 결과를 생성하거나 해결하는 데 최대 6시간까지 걸릴 수 있습니다. 또한 AWS CloudTrail 로그 전달에 전송 문제가 있는 경우, 정책을 변경해도 결과에 보고된 리소스의 다시 검색을 트리거하지 않습니다. 이런 경우 IAM Access Analyzer는 다음 주기적 검색 중에(24시간 이내) 새 정책 또는 업데이트된 정책을 분석합니다. 정책을 변경하면 결과에 보고된 액세스 문제가 해결되는지 확인하려는 경우 결과 세부 정보 페이지의 다시 검색(Rescan) 링크를 사용하거나 IAM Access Analyzer API의 `StartResourceScan` 작업을 사용하여 결과에 보고된 리소스를 다시 검색할 수 있습니다. 자세한 내용은 IAM Access Analyzer 조사 결과 해결을 참조하십시오.

> **Important**
>
> IAM Access Analyzer에서는 해당 Access Analyzer가 활성화된 AWS 리전의 리소스에 적용되는 정책만 분석합니다. AWS 환경의 모든 리소스를 모니터링하려면 지원되는 AWS 리소스를 사용 중인 각 리전에서 IAM Access Analyzer를 활성화하기 위해 분석기를 생성해야 합니다.

IAM Access Analyzer는 다음과 같은 리소스 유형을 분석합니다.

- Amazon Simple Storage Service 버킷
- Amazon Simple Storage Service 디렉터리 버킷
- AWS Identity and Access Management 역할
- AWS Key Management Service 키
- AWS Lambda 함수 및 계층
- Amazon Simple Queue Service 대기열
- AWS Secrets Manager 보안 암호
- Amazon Simple Notification Service(Amazon SNS) 주제
- Amazon Elastic Block Store 볼륨 스냅샷
- Amazon Relational Database Service DB 스냅샷
- Amazon Relational Database Service DB 클러스터 스냅샷
- Amazon Elastic 컨테이너 레지스트리 리포지토리
- Amazon Elastic File System 파일 시스템
- Amazon DynamoDB Streams
- Amazon DynamoDB 테이블

IAM 사용자 및 역할에 부여된 미사용 액세스 권한 식별

IAM Access Analyzer를 사용하면 AWS 조직 및 계정 내 미사용 액세스를 식별하고 검토할 수 있습니다. IAM Access Analyzer는 AWS 조직 및 계정의 모든 IAM 역할과 사용자를 지속적으로 모니터링하고 미사용 액세스에 대한 조사 결과를 생성합니다. 조사 결과에는 미사용 역할, IAM 사용자의 미사용 액세스 키, IAM 사용자의 미사용 암호가 강조 표시됩니다. 활성 IAM 역할 및 사용자의 경우, 조사 결과를 통해 미사용 서비스 및 작업을 파악할 수 있습니다.

외부 액세스 및 미사용 액세스 분석기의 조사 결과는 시각적 요약 대시보드에 정리되어 있습니다. 대시보드에는 조사 결과가 가장 많은 AWS 계정이 강조 표시되며 유형별로 조사 결과를 분류하여 제공합니다. 대시보드에 대한 자세한 내용은 IAM Access Analyzer 조사 결과 대시보드 보기 섹션을 참조하세요.

IAM Access Analyzer는 미사용 액세스를 식별할 수 있도록 AWS 조직 및 계정의 모든 역할에 대해 마지막으로 액세스한 정보를 검토합니다. IAM 작업 마지막으로 액세스한 정보는 AWS 계정 내 역할에 대한 미사용 작업을 식별하는 데 도움이 됩니다. 자세한 내용은 마지막으로 액세스한 정보를 사용하여 AWS에서의 권한 재정의 단원을 참조하십시오.

AWS 모범 사례와 비교하여 정책 검증

IAM Access Analyzer 정책 검증에서 제공하는 기본 정책 확인을 사용하여 IAM 정책 문법 및 AWS 모범 사례에 따라 정책을 검증할 수 있습니다. AWS CLI, AWS API 또는 IAM 콘솔의 JSON 정책 편집기를 사용하여 정책을 생성 또는 편집할 수 있습니다. 보안 경고, 오류, 일반 경고 및 정책에 대한 제안 사항이 포함된 정책 유효성 검사 결과를 볼 수 있습니다. 이러한 조사 결과는 AWS 모범 사례 준수 기능을 갖춘 정책을 작성하는 데 도움이 되는 실행 가능한 권장 사항을 제공합니다. 정책 검증을 사용하여 정책을 검증하는 방법에 대한 자세한 내용은 IAM Access Analyzer 정책 검증 섹션을 참조하세요.

지정된 보안 표준에 따라 정책을 검증

IAM Access Analyzer 사용자 지정 정책 확인을 사용하여 지정된 보안 표준에 따라 정책을 검증할 수 있습니다. AWS CLI, AWS API 또는 IAM 콘솔의 JSON 정책 편집기를 사용하여 정책을 생성 또는 편집할 수 있습니다. 콘솔을 통해 업데이트된 정책이 기존 버전과 비교하여 새 액세스를 허용하는지 확인할 수 있습니다. AWS CLI 및 AWS API를 통해 중요하다고 생각되는 특정 IAM 작업이 정책에서 허용되지 않는지 확인할 수도 있습니다. 이러한 확인에서 새 액세스 권한을 부여하는 정책 내용을 강조 표시합니다. 정책이 보안 표준을 준수할 때까지 정책 내용을 업데이트하고 확인을 다시 실행할 수 있습니다. 사용자 지정 정책 확인을 통해 정책을 검증하는 방법에 대한 자세한 내용은 IAM Access Analyzer 정책 확인 섹션을 참조하세요.

정책 생성

IAM Access Analyzer는 AWS CloudTrail 로그를 분석하여 지정된 날짜 범위 내에서 IAM 엔터티(사용자 또는 역할)가 사용한 작업 및 서비스를 식별합니다. 그런 다음 해당 액세스 활동을 기반으로 하는 IAM 정책을 생성합니다. 생성된 정책을 IAM 사용자 또는 역할에 연결하여 엔터티의 권한을 세분화하는 데 사용할 수 있습니다. IAM Access Analyzer를 사용하여 정책을 생성하는 방법에 대한 자세한 내용은 IAM Access Analyzer 정책 생성 섹션을 참조하세요.

IAM Access Analyzer 요금

IAM Access Analyzer는 매월 분석기별로 분석된 IAM 역할 및 사용자 수를 기준으로 미사용 액세스 분석에 대한 요금을 부과합니다.

- 생성한 각 미사용 액세스 분석기에 대한 요금이 청구됩니다.
- 여러 리전에 걸쳐 미사용 액세스 분석기를 생성하면 각 분석기에 대한 요금이 부과됩니다.
- 서비스 연결 역할은 미사용 액세스 활동이 분석되지 않으며 분석된 총 IAM 역할 수에 포함되지 않습니다.

IAM Access Analyzer는 새 액세스를 확인하기 위해 IAM Access Analyzer에 보낸 API 요청 수를 기준으로 사용자 지정 정책 확인에 대한 요금을 부과합니다.

IAM Access Analyzer에 관련된 전체적인 요금 및 가격 목록은 IAM Access Analyzer 요금을 참조하세요.

청구 요금은 AWS Billing and Cost Management 콘솔의 청구 및 비용 관리 대시보드에서 확인할 수 있습니다. 청구서에는 요금 내역을 자세하게 확인할 수 있는 사용 보고서 링크가 포함됩니다. AWS 계정 결제에 대한 자세한 내용은 AWS Billing 사용 설명서를 참조하세요.

AWS 결제, 계정 및 이벤트에 관련된 질문은 AWS Support에 문의하세요.

외부 및 미사용 액세스에 대한 조사 결과

IAM Access Analyzer는 사용자의 AWS 계정 또는 조직 내 외부 액세스 및 미사용 액세스에 대한 조사 결과를 생성합니다. 외부 액세스의 경우, IAM Access Analyzer는 신뢰 영역 내에 있지 않은 보안 주체에 대한 신뢰 영역의 리소스에 액세스 권한을 부여하는 리소스 기반 정책의 각 인스턴스에 대한 결과를 생성합니다. 외부 액세스 분석기를 생성할 때 분석할 조직 또는 AWS 계정을 선택합니다. 분석기에 대해 선택한 조직 또는 계정의 모든 보안 주체는 신뢰할 수 있는 것으로 간주됩니다. 동일한 조직 또는 계정의 보안 주체는 신뢰할 수 있으므로 조직 또는 계정 내의 리소스 및 보안 주체는 분석기의 신뢰 영역을 구성합니다. 신뢰 영역 내에서의 모든 공유는 안전한 것으로 간주되므로 IAM Access Analyzer에서 결과가 생성되지 않습니다. 예를 들어 조직을 분석기의 신뢰 영역으로 선택하면 조직의 모든 리소스 및 보안 주체가 신뢰 영역 내에 있습니다. 조직 구성원 계정 중 하나의 Amazon S3 버킷에 대한 권한을 다른 조직 구성원 계정의 보안 주체에게 부여하면 IAM Access Analyzer가 결과를 생성하지 않습니다. 그러나 조직의 멤버가 아닌 계정의 보안 주체에게 권한을 부여하면 IAM Access Analyzer가 결과를 생성합니다.

IAM Access Analyzer는 AWS 조직 및 계정에 부여된 미사용 액세스에 대한 조사 결과도 생성합니다. 미사용 액세스 분석기를 생성하면 IAM Access Analyzer는 AWS 조직 및 계정의 모든 IAM 역할과 사용자를 지속적으로 모니터링하고 미사용 액세스에 대한 조사 결과를 생성합니다. IAM Access Analyzer는 미사용 액세스에 대해 다음과 같은 유형의 조사 결과를 생성합니다.

- 미사용 역할 - 지정된 사용 창 내에 액세스 활동이 없는 역할입니다.
- 미사용 IAM 사용자 액세스 키 및 암호 - IAM 사용자가 AWS 계정에 액세스할 수 있게 해주는 IAM 사용자 소유 자격 증명서입니다.
- 미사용 권한 - 지정된 사용 기간 내에 역할이 사용하지 않은 서비스 수준 및 작업 수준 권한입니다. IAM Access Analyzer는 역할에 연결된 자격 기반 정책을 통해 해당 역할이 액세스할 수 있는 서비스와 작업을 결정합니다. IAM Access Analyzer는 모든 서비스 수준 권한에 대한 미사용 권한 검토를 지원합니다. 미사용 액세스 조사 결과에 대해 지원되는 작업 수준 권한의 전체 목록은 IAM 작업에서 마지막으로 액세스한 정보와 관련된 서비스 및 작업을 참조하세요.

> **Note**
>
> IAM Access Analyzer는 외부 액세스 조사 결과를 무료로 제공하며, 매월 분석기별로 분석된 IAM 역할 및 사용자 수를 기준으로 미사용 액세스 조사 결과에 대해 요금을 부과합니다. 요금에 대한 자세한 내용은 IAM Access Analyzer 요금을 참조하세요.

주제

- IAM Access Analyzer 조사 결과 작동 방식 이해
- AWS Identity and Access Management Access Analyzer 결과 시작하기
- IAM Access Analyzer 조사 결과 대시보드 보기
- 조사 결과 작업
- 결과 검토
- 조사 결과 필터링
- 결과 아카이브
- IAM Access Analyzer 조사 결과 해결
- 외부 액세스를 위한 IAM Access Analyzer 리소스 유형
- IAM Access Analyzer 설정
- 아카이브 규칙
- Amazon EventBridge로 AWS Identity and Access Management Access Analyzer 모니터링

- AWS Security Hub와 IAM Access Analyzer의 통합
- AWS CloudTrail을 사용하여 IAM Access Analyzer API 호출 로깅
- IAM Access Analyzer 필터 키
- AWS Identity and Access Management Access Analyzer에 서비스 연결 역할 사용

IAM Access Analyzer 조사 결과 작동 방식 이해

이 항목에서는 IAM Access Analyzer에서 AWS 리소스에 대한 액세스를 모니터링하는 방법을 익히는 데 도움이 되도록 IAM Access Analyzer에서 사용되는 개념과 용어에 대해 설명합니다.

IAM Access Analyzer가 외부 액세스에 대한 조사 결과를 생성하는 방법

AWS Identity and Access Management Access Analyzer는 Zelkova라는 기술을 사용하여 IAM 정책을 분석하고 리소스에 대한 외부 액세스를 식별합니다.

Zelkova는 IAM 정책을 동등한 논리적 명령문으로 변환하여 일련의 범용 및 특수 논리 해석기(만족성 모듈로 이론)를 통해 실행합니다. IAM Access Analyzer는 정책의 내용에 따라 허용되는 액세스 유형을 특성화하기 위해 점점 더 구체적인 쿼리를 사용하여 Zelkova를 정책에 반복적으로 적용합니다. 만족성 모듈로 이론에 대한 자세한 내용은 만족성 모듈로 이론을 참조하세요.

외부 액세스 결과의 경우, IAM Access Analyzer는 외부 엔터티가 신뢰 영역 내의 리소스에 실제로 액세스했는지를 확인하기 위해 액세스 로그를 검사하지 않습니다. 대신, 외부 엔터티가 리소스에 액세스했는지 여부와 관계없이 리소스 기반 정책에서 리소스에 대한 액세스를 허용할 때는 결과를 생성합니다.

또한 IAM Access Analyzer는 또한 결정을 내릴 때 외부 계정의 상태를 고려하지 않습니다. 계정 111122223333이 Amazon S3 버킷에 액세스할 수 있음을 나타내는 경우에는 해당 계정의 사용자, 역할, 서비스 제어 정책(SCP) 또는 기타 관련 구성에 대해서는 알 수 없습니다. 이는 고객 개인 정보 보호를 위한 것입니다. IAM Access Analyzer는 누가 다른 계정을 소유하고 있는지 알 수 없기 때문입니다. 이는 보안을 위한 것이기도 합니다. 현재 이를 사용할 수 있는 활성 보안 주체가 없더라도 잠재적인 외부 액세스에 대해 아는 것이 중요하기 때문입니다.

IAM Access Analyzer에서는 외부 사용자가 직접 영향을 줄 수 없거나 권한 부여에 영향력을 행사하는 특정 IAM 조건 키만 고려합니다. IAM Access Analyzer에서 고려하는 조건 키의 예는 IAM Access Analyzer 필터 키를 참조하세요.

현재 IAM Access Analyzer에서는 AWS 서비스 보안 주체 또는 내부 서비스 계정에서 조사 결과를 보고하지 않습니다. 드물지만 Access Analyzer에서 정책문이 외부 엔터티에 대한 액세스 권한을 부여하

는지 여부를 완전히 확인할 수 없는 경우에는 거짓 긍정 결과를 선언하는 오류가 발생합니다. 이는 IAM Access Analyzer가 계정의 리소스 공유에 대한 포괄적인 보기를 제공하고 거짓 부정을 최소화하도록 설계되었기 때문입니다.

IAM Access Analyzer가 미사용 액세스에 대한 조사 결과를 생성하는 방법

미사용 액세스를 분석하려면 리소스에 대한 외부 액세스 조사 결과를 생성하는 분석기를 이미 생성했더라도 역할에 미사용 액세스 조사 결과에 대한 별도의 분석기를 만들어야 합니다.

IAM Access Analyzer는 미사용 액세스 분석기를 생성한 후 액세스 활동을 검토하여 미사용 액세스를 식별합니다. IAM Access Analyzer는 AWS 조직 및 계정의 모든 역할, 사용자 액세스 키, 사용자 암호에 대해 마지막으로 액세스한 정보를 검토합니다. 이는 미사용 액세스를 식별하는 데 도움이 됩니다.

활성 IAM 역할 및 사용자의 경우 IAM Access Analyzer는 IAM 서비스 및 작업이 마지막으로 액세스한 정보를 사용하여 미사용 권한을 식별합니다. 이렇게 하면 AWS 조직 및 계정 수준에서 검토 프로세스를 확장할 수 있습니다. 또한 마지막으로 액세스한 작업 정보를 사용하여 개별 역할을 심층적으로 조사할 수도 있습니다. 이를 통해 어떤 특정 권한이 사용되지 않는지 더 세밀하게 파악할 수 있습니다.

미사용 액세스 전용 분석기를 만들면 AWS 환경 전반의 미사용 액세스를 종합적으로 검토 및 식별하여 기존 외부 액세스 분석기에서 생성된 조사 결과를 보완할 수 있습니다.

요약 대시보드에서 IAM Access Analyzer 조사 결과 보기

IAM Access Analyzer는 외부 액세스 조사 결과와 미사용 액세스 조사 결과를 모두 요약 대시보드에 정리합니다. 외부 액세스 조사 결과:

- 대시보드에 퍼블릭 액세스 조사 결과와 크로스 계정 액세스 조사 결과 간의 차이가 강조 표시됩니다.
- 대시보드는 조사 결과를 리소스 유형별로 분류하여 제공합니다.

미사용 액세스 조사 결과:

- 대시보드에 미사용 액세스 조사 결과가 가장 많은 AWS 계정이 강조 표시됩니다.
- 대시보드는 조사 결과를 유형별로 분류하여 제공합니다.

외부 또는 미사용 액세스에 대한 분석기를 만들면 IAM Access Analyzer는 새로운 조사 결과를 대시보드에 자동으로 추가합니다. 이를 통해 보안 문제가 가장 많은 영역을 식별하고 우선 순위를 정할 수 있습니다.

요약 대시보드를 통해 AWS 환경 전반에서 IAM Access Analyzer가 탐지한 액세스 문제를 개괄적으로 파악할 수 있습니다. 그런 다음 개별 조사 결과를 드릴다운하여 더 자세히 조사하고 적절한 조치를 취하여 문제를 해결할 수 있습니다.

AWS Identity and Access Management Access Analyzer 결과 시작하기

이 주제의 정보를 사용하여 AWS Identity and Access Management Access Analyzer를 사용하고 관리하는 데 필요한 요구 사항과 IAM Access Analyzer를 활성화하는 방법에 대해 알아봅니다. IAM Access Analyzer의 서비스 연결 역할에 대한 자세한 내용은 AWS Identity and Access Management Access Analyzer에 서비스 연결 역할 사용 섹션을 참조하세요.

IAM Access Analyzer를 사용하는 데 필요한 권한

IAM Access Analyzer를 성공적으로 구성하고 사용하려면 사용하는 계정에 필요한 권한을 부여해야 합니다.

IAM Access Analyzer의 AWS 관리형 정책

AWS Identity and Access Management Access Analyzer에서는 빠르게 시작하는 데 도움이 되는 AWS 관리형 정책을 제공합니다.

- IAMAccessAnalyzerFullAccess - 관리자에게 IAM Access Analyzer에 대한 전체 액세스 권한을 허용합니다. 또한 이 정책을 통해 IAM Access Analyzer가 사용자 계정 또는 AWS 조직의 리소스를 분석할 수 있도록 하는 데 필요한 서비스 연결 역할을 생성할 수 있습니다.
- IAMAccessAnalyzerReadOnlyAccess - IAM Access Analyzer에 대한 읽기 전용 액세스 권한을 허용합니다. IAM 자격 증명(사용자, 사용자 그룹 또는 역할)이 결과를 볼 수 있도록 허용하는 정책을 추가해야 합니다.

IAM Access Analyzer에서 정의한 리소스

IAM Access Analyzer에서 정의한 리소스를 보려면 서비스 권한 부여 참조의 IAM Access Analyzer에서 정의한 리소스 유형을 참조하세요.

필요한 IAM Access Analyzer 서비스 권한

IAM Access Analyzer는 `AWSServiceRoleForAccessAnalyzer`라는 서비스 연결 역할(SLR)을 사용합니다. 해당 SLR은 사용자를 대신해 리소스 기반 정책으로 AWS 리소스를 분석하고 미사용 액세스를 분석할 수 있도록 서비스에 읽기 전용 액세스 권한을 부여합니다. 다음 시나리오에서 서비스는 계정에 역할을 생성합니다.

- 계정을 신뢰 영역으로 하여 외부 액세스 분석기를 만듭니다.
- 사용자 계정을 선택한 계정으로 사용하여 미사용 액세스 분석기를 생성합니다.

자세한 내용은 AWS Identity and Access Management Access Analyzer에 서비스 연결 역할 사용 단원을 참조하십시오.

> **Note**
>
> IAM Access Analyzer는 리전별로 적용됩니다. 외부 액세스를 위해 각 리전에서 독립적으로 IAM Access Analyzer를 활성화해야 합니다.
> 미사용 액세스의 경우 분석기에 대한 조사 결과는 리전에 따라 변경되지 않습니다. 리소스가 있는 각 리전에 분석기를 만들 필요는 없습니다.

경우에 따라 IAM Access Analyzer에서 외부 액세스 또는 미사용 액세스 분석기를 만들면 조사 결과 또는 요약 없이 조사 결과 페이지 또는 대시보드가 로드됩니다. 이는 결과를 채우기 위해 콘솔에서 발생하는 지연으로 인한 것일 수 있습니다. 조사 결과 또는 요약을 보려면 브라우저를 수동으로 새로고침하거나 나중에 다시 확인해야 할 수 있습니다. 외부 엔터티가 액세스할 수 있는 지원 리소스가 계정에 없으면 외부 액세스 분석기에 대한 조사 결과가 표시되지 않습니다. 외부 엔터티에 대한 액세스 권한을 부여하는 정책이 리소스에 적용되는 경우 IAM Access Analyzer에서 결과를 생성합니다.

> **Note**
>
> 외부 액세스 분석기의 경우, 정책이 수정되고 최대 30분 후에 IAM Access Analyzer에서 리소스를 분석한 다음 리소스 액세스에 대한 새 외부 액세스 결과를 생성하거나 기존 결과를 업데이트할 수 있습니다. 외부 액세스 분석기와 미사용 액세스 분석기 모두에서 조사 결과에 대한 업데이트가 대시보드에 즉시 반영되지 않을 수 있습니다.

조사 결과 대시보드를 보기 위한 IAM Access Analyzer 필수 권한

IAM Access Analyzer 조사 결과 대시보드를 보려면 사용하는 계정에 다음 필수 작업을 수행할 수 있도록 액세스 권한을 부여해야 합니다.

- `GetAnalyzer`
- `ListAnalyzers`
- `GetFindingsStatistics`

IAM Access Analyzer에서 정의한 모든 작업을 보려면 서비스 권한 부여 참조에서 IAM Access Analyzer에서 정의한 작업을 참조하세요.

IAM Access Analyzer 활성화

AWS 계정을 신뢰 영역으로 설정하여 분석기를 생성하려면

리전에서 외부 액세스 분석기를 활성화하려면 해당 리전에서 분석기를 생성해야 합니다. 리소스에 대한 액세스 권한을 모니터링하려는 각 리전에서 분석기를 생성해야 합니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. Access analyzer(분석기 액세스)를 선택합니다.
3. 분석기 설정을 선택합니다.
4. Create analyzer(분석기 생성)를 선택합니다.
5. 분석 섹션에서 외부 액세스 분석을 선택합니다.
6. 분석기 상세 정보 섹션에서 표시된 리전이 IAM Access Analyzer를 활성화하려는 리전인지 확인합니다.
7. 분석기의 이름을 입력합니다.
8. 분석기의 신뢰 영역으로 현재 AWS 계정을 선택합니다.

> **Note**
>
> 계정이 AWS Organizations 관리 계정 또는 위임된 관리자 계정이 아닌 경우, 계정을 신뢰 영역으로 하여 분석기를 하나만 생성할 수 있습니다.

9. 선택 사항입니다. 분석기에 적용할 태그를 추가합니다.
10. 제출을 선택합니다.

IAM Access Analyzer를 활성화하기 위해 외부 액세스 분석기를 생성하면 `AWSServiceRoleForAccessAnalyzer`라는 이름의 서비스 연결 역할이 계정에서 생성됩니다.

조직을 신뢰 영역으로 하여 외부 액세스 분석기를 생성하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. Access analyzer(분석기 액세스)를 선택합니다.
3. 분석기 설정을 선택합니다.

4. Create analyzer(분석기 생성)를 선택합니다.
5. 분석 섹션에서 외부 액세스 분석을 선택합니다.
6. 분석기 상세 정보 섹션에서 표시된 리전이 IAM Access Analyzer를 활성화하려는 리전인지 확인합니다.
7. 분석기의 이름을 입력합니다.
8. 현재 조직을 분석기의 신뢰 영역으로 선택합니다.
9. 선택 사항입니다. 분석기에 적용할 태그를 추가합니다.
10. 제출을 선택합니다.

조직을 신뢰 영역으로 하여 외부 액세스 분석기를 만들면 조직의 각 계정에 `AWSServiceRoleForAccessAnalyzer`이라는 서비스 연결 역할이 만들어집니다.

현재 계정에 미사용 액세스 분석기를 생성하려면

다음 절차를 통해 단일 AWS 계정을 위한 미사용 액세스 분석기를 생성합니다. 미사용 액세스의 경우 분석기에 대한 조사 결과는 리전에 따라 변경되지 않습니다. 리소스가 있는 각 리전에 분석기를 만들 필요는 없습니다.

IAM Access Analyzer는 분석기별로 매월 분석된 IAM 역할 및 사용자 수를 기준으로 미사용 액세스 분석에 대한 요금을 부과합니다. 요금에 대한 자세한 내용은 IAM Access Analyzer 요금을 참조하세요.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. Access analyzer(분석기 액세스)를 선택합니다.
3. 분석기 설정을 선택합니다.
4. Create analyzer(분석기 생성)를 선택합니다.
5. 분석 섹션에서 미사용 액세스 분석을 선택합니다.
6. 분석기의 이름을 입력합니다.
7. 추적 기간에는 미사용 권한에 대한 조사 결과를 생성할 기간(일수)을 입력합니다. 예를 들어 90일을 입력하면 분석기는 마지막 스캔 이후 90일 이상 미사용 권한에 대해 선택한 계정 내의 IAM 엔터티에 대한 조사 결과를 생성합니다. 1일에서 180일 사이의 값을 선택할 수 있습니다.
8. 선택한 계정에서 현재 AWS 계정을 선택합니다.

> **Note**
>
> 계정이 AWS Organizations 관리 계정 또는 위임된 관리자 계정이 아닌 경우, 선택된 계정으로 분석기를 하나만 생성할 수 있습니다.

9. 선택 사항입니다. 분석기에 적용할 태그를 추가합니다.
10. 제출을 선택합니다.

IAM Access Analyzer를 활성화하기 위해 미사용 액세스 분석기를 생성하면 `AWSServiceRoleForAccessAnalyzer`라는 이름의 서비스 연결 역할이 계정에서 생성됩니다.

현재 조직에서 미사용 액세스 분석기를 생성하려면

다음 절차를 따라 조직에서 미사용 액세스 분석기를 만들어 조직의 모든 AWS 계정을 중앙에서 검토할 수 있습니다. 미사용 액세스 분석의 경우 분석기에 대한 조사 결과는 리전에 따라 변경되지 않습니다. 리소스가 있는 각 리전에 분석기를 만들 필요는 없습니다.

IAM Access Analyzer는 분석기별로 매월 분석된 IAM 역할 및 사용자 수를 기준으로 미사용 액세스 분석에 대한 요금을 부과합니다. 요금에 대한 자세한 내용은 IAM Access Analyzer 요금을 참조하세요.

> **Note**
>
> 구성원 계정이 조직에서 제거되면 미사용 액세스 분석기는 24시간 후부터 새 조사 결과를 생성하지 않으며 해당 계정에 대한 기존 조사 결과 업데이트를 중단합니다. 조직에서 제거된 구성원 계정과 관련된 조사 결과는 90일 후에 영구적으로 제거됩니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. Access analyzer(분석기 액세스)를 선택합니다.
3. 분석기 설정을 선택합니다.
4. Create analyzer(분석기 생성)를 선택합니다.
5. 분석 섹션에서 미사용 액세스 분석을 선택합니다.
6. 분석기의 이름을 입력합니다.
7. 추적 기간에는 미사용 권한에 대한 조사 결과를 생성할 기간(일수)을 입력합니다. 예를 들어 90일을 입력하면 분석기는 분석기의 마지막 스캔 이후 90일 이상 미사용 권한에 대해 선택한 조직의 계

정 내 IAM 엔터티에 대한 조사 결과를 생성합니다. 1일에서 180일 사이의 값을 선택할 수 있습니다.

8. 선택된 계정의 경우 분석기의 선택된 계정으로 현재 조직을 선택합니다.
9. 선택 사항입니다. 분석기에 적용할 태그를 추가합니다.
10. 제출을 선택합니다.

IAM Access Analyzer를 활성화하기 위해 미사용 액세스 분석기를 생성하면 `AWSServiceRoleForAccessAnalyzer`라는 이름의 서비스 연결 역할이 계정에서 생성됩니다.

IAM Access Analyzer 상태

분석기의 상태를 보려면 분석기를 선택합니다. 조직 또는 계정에 대해 생성된 분석기의 상태는 다음과 같습니다.

상태 표시기	설명
활성	외부 액세스 분석기에 대해 분석기는 신뢰 영역 내에서 리소스를 적극적으로 모니터링합니다. 분석기는 새로운 결과를 적극적으로 생성하고 기존 결과를 업데이트합니다. 미사용 액세스 분석기의 경우 분석기는 선택한 조직 내에서 또는 지정된 추적 기간의 AWS 계정 내에서 미사용 액세스를 능동적으로 모니터링합니다. 분석기는 새로운 결과를 적극적으로 생성하고 기존 결과를 업데이트합니다.
[생성 중]	분석기 생성이 아직 진행 중입니다. 생성이 완료되면 분석기가 활성화됩니다.
Disabled(비활성)	AWS Organizations 관리자가 수행한 작업으로 인해 분석기가 비활성화되었습니다. 예를 들어 IAM Access Analyzer의 위임된 관리자로서 분석기의 계정을 제거합니다. 분석기가 비활성화된 상태면 새 조사 결과를 생성하거나 기존 조사 결과를 업데이트하지 않습니다.

상태 표시기	설명
Failed	구성 문제로 인해 분석기를 만들지 못했습니다. 분석기는 결과를 생성하지 않습니다. 분석기를 삭제하고 새 분석기를 만듭니다.

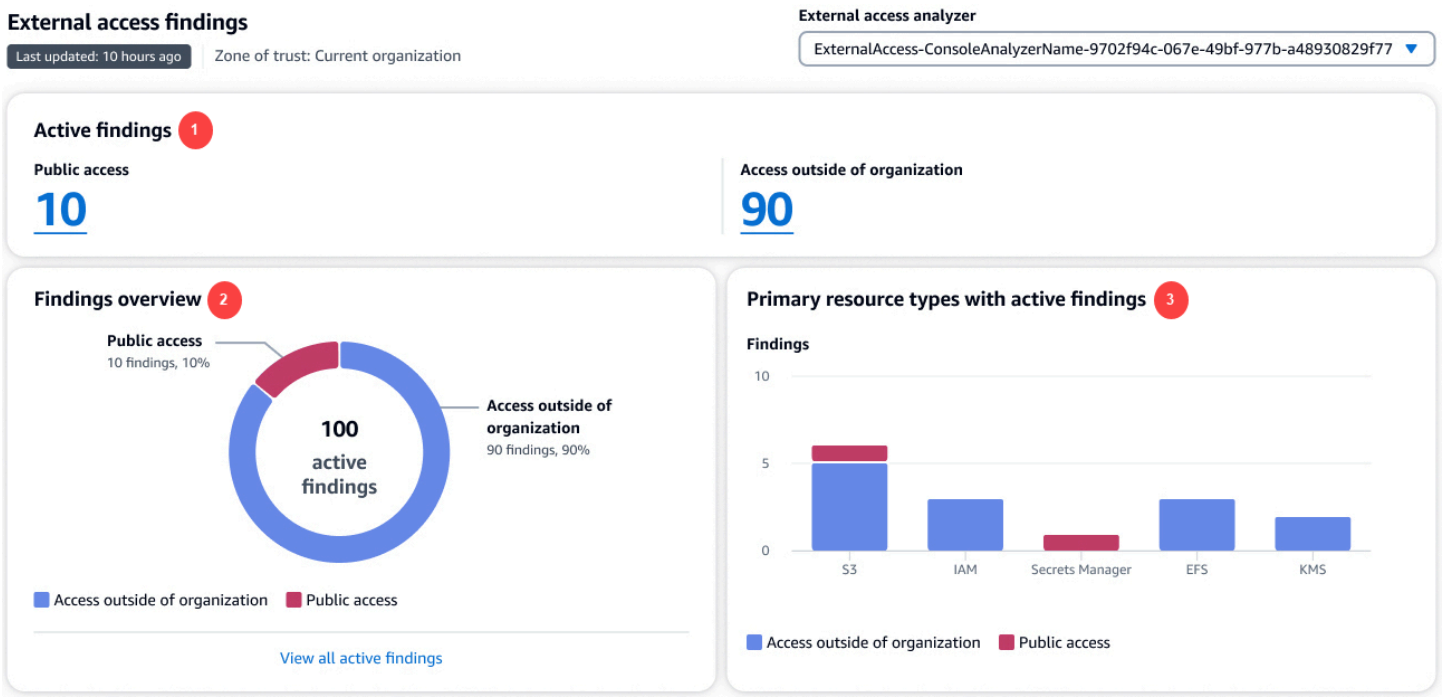
IAM Access Analyzer 조사 결과 대시보드 보기

AWS Identity and Access Management Access Analyzer는 외부 액세스 및 미사용 액세스 조사 결과를 시각적 요약 대시보드로 정리합니다. 대시보드를 통해 대규모 권한의 효과적인 사용을 가시적으로 확인할 수 있고 주의가 필요한 계정을 식별할 수 있습니다. 대시보드를 사용하여 AWS 조직, 계정 및 검색 유형별로 조사 결과를 검토할 수 있습니다.

외부 액세스 분석기의 요약 대시보드를 보려면

> **Note**
>
> 분석기를 만들거나 업데이트한 후 요약 대시보드에 조사 결과 업데이트가 반영되는 데 시간이 걸릴 수 있습니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. Access analyzer(분석기 액세스)를 선택합니다. 요약 창이 표시됩니다.
3. 외부 액세스 분석기 드롭다운에서 분석기를 선택합니다. 분석기에 대한 조사 결과 요약이 외부 액세스 조사 결과 섹션에 표시됩니다.



이전 이미지에서 외부 액세스 조사 결과 대시보드 화면은 요약 페이지에 표시되어 있습니다.

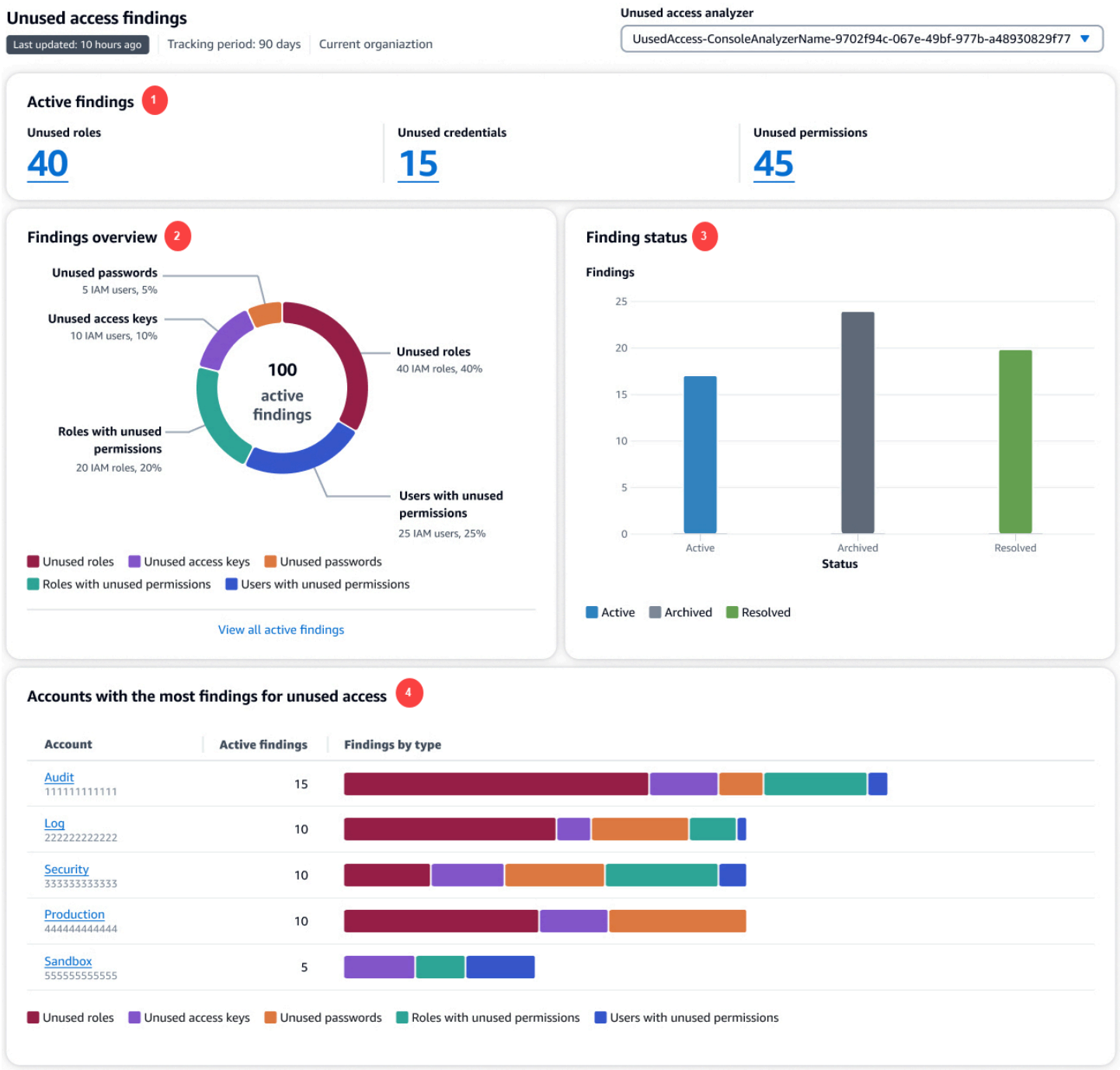
1. 활성 조사 결과 섹션에는 공개 액세스를 위한 활성 조사 결과 수와 계정 또는 조직 외부에서 액세스를 제공하는 활성 조사 결과 수가 포함됩니다. 숫자를 선택하면 각 유형의 활성 조사 결과를 모두 나열할 수 있습니다.
2. 조사 결과 개요 섹션에는 활성 결과 유형의 분류가 포함되어 있습니다. 분석기 계정 또는 조직에 대한 활성 조사 결과의 전체 목록을 보려면 모든 활성 조사 결과 보기를 선택합니다.
3. 활성 조사 결과가 포함된 기본 리소스 유형 섹션에는 활성 조사 결과가 포함된 기본 리소스 유형의 분류가 포함되어 있습니다. 이 정보는 기본 리소스에 대한 조사 결과의 우선 순위를 정하는 데 도움이 됩니다. Amazon S3, DynamoDB 및 AWS KMS를 예로 들 수 있습니다. 이 목록은 모든 리소스 유형의 전체 목록이 아닙니다. 분석기가 이 섹션에 나열되지 않은 리소스 유형에 대한 활성 조사 결과를 포함할 수 있습니다.

미사용 액세스 분석기의 요약 대시보드를 보려면

IAM Access Analyzer는 매월 분석된 IAM 역할 및 사용자 수를 기준으로 미사용 액세스 분석에 대해 요금을 부과합니다. 요금에 대한 자세한 내용은 IAM Access Analyzer 요금을 참조하세요.

> **Note**
>
> 분석기를 만들거나 업데이트한 후에는 사용자 및 역할의 수에 따라 요약 대시보드에 조사 결과 업데이트가 반영되는 데 시간이 소요될 수 있습니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. Access analyzer(분석기 액세스)를 선택합니다. 요약 창이 표시됩니다.
3. 미사용 액세스 분석기 드롭다운에서 분석기를 선택합니다. 분석기에 대한 조사 결과 요약이 미사용 액세스 조사 결과 섹션에 표시됩니다.



이전 이미지에서 외부 액세스 조사 결과 대시보드 화면은 요약 페이지에 표시되어 있습니다.

1. 활성 조사 결과 섹션에는 계정이나 조직 내 미사용 역할, 미사용 자격 증명, 미사용 권한에 대한 활성 조사 결과 수가 포함됩니다. 미사용 자격 증명에는 미사용 액세스 키와 미사용 암호 조사 결과가 모두 포함됩니다. 미사용 권한에는 미사용 권한을 가진 사용자와 역할이 모두 포함됩니다. 숫자를 선택하면 각 유형의 활성 조사 결과를 모두 나열할 수 있습니다.

2. 조사 결과 개요 섹션에는 활성 결과 유형의 분류가 포함되어 있습니다. 분석기 계정 또는 조직에 대한 활성 결과의 전체 목록을 보려면 모든 활성 조사 결과 보기를 선택합니다.
3. 스캔 조사 결과 상태 섹션에는 계정 또는 조직의 조사 결과 상태(활성, 아카이브됨, 해결됨)에 대한 분류가 포함되어 있습니다.
4. 미사용 액세스에 대한 조사 결과가 가장 많은 계정 섹션은 미사용 액세스 분석기에서 선택한 계정이 조직 수준에 있는 경우에만 표시됩니다. 여기에는 가장 활성화된 조사 결과와 조직 내 계정의 세부 정보가 포함됩니다. 해당 목록은 조직 내 모든 계정이 표시된 목록이 아닙니다. 분석기에 이 섹션에 나열되지 않은 다른 계정에 대한 활성 조사 결과가 있을 수 있습니다.

조사 결과 작업

외부 액세스 조사 결과

외부 액세스 조사 결과는 신뢰 영역 외부에서 공유되는 리소스의 각 인스턴스에 대해 한 번만 생성됩니다. 리소스 기반 정책이 수정될 때마다 IAM Access Analyzer가 정책을 분석합니다. 업데이트된 정책이 결과에서 이미 식별되었지만 사용 권한 또는 조건이 다른 리소스를 공유하는 경우, 리소스를 공유하는 해당 인스턴스에 대해 새 결과가 생성됩니다. 첫 번째 조사 결과에서의 액세스가 제거되면 해당 조사 결과는 해결됨 상태로 업데이트됩니다.

조사 결과를 아카이브하거나 조사 결과를 생성한 액세스 권한을 제거할 때까지 모든 조사 결과는 활성 상태로 유지됩니다. 액세스 권한을 제거하면 조사 결과 상태가 해결됨으로 업데이트됩니다.

> **Note**
>
> IAM Access Analyzer에 대한 정책을 수정한 후 리소스를 분석하고 외부 액세스 조사 결과를 업데이트하기까지 최대 30분이 걸릴 수 있습니다.

미사용 액세스 조사 결과

미사용 액세스 조사 결과는 분석기를 생성하는 동안 지정된 일수를 기준으로 선택한 계정 또는 조직 내의 IAM 엔터티에 대해 생성됩니다. 다음 조건 중 하나가 충족되는 경우 다음 번에 분석기가 엔터티를 스캔할 때 새로운 조사 결과가 생성됩니다.

- 지정된 일수 동안 역할이 비활성 상태입니다.
- 미사용 권한, 미사용 사용자 암호 또는 미사용 사용자 액세스 키가 지정된 일수를 초과했습니다.

계정의 모든 조사 결과를 검토하여 외부 액세스 또는 미사용 액세스가 예상 및 승인되었는지 여부를 확인해야 합니다. 조사 결과에서 식별된 외부 액세스 또는 미사용 액세스가 예상되는 경우 조사 결과를 아카이브할 수 있습니다. 조사 결과를 아카이브하면 상태가 아카이브됨으로 변경되고 조사 결과는 활성 조사 결과 목록에서 제거됩니다. 결과는 삭제되지 않습니다. 아카이브된 결과를 언제든지 볼 수 있습니다. 활성 결과가 0이 될 때까지 계정의 모든 결과를 살펴봅니다. 조사 결과가 0이 되면 새로 생성된 모든 활성 조사 결과가 환경의 최근 변경에서 나온 것이라는 뜻입니다.

> **Note**
>
> 사용되지 않은 액세스 조사 결과는 ListFindingsV2 API 작업을 통해서만 사용할 수 있습니다.

결과 검토

IAM Access Analyzer를 활성화한 후 다음 단계 결과를 검토하여 결과에서 식별된 액세스가 의도적인지 아닌지 여부를 확인하는 것입니다. 또한 조사 결과를 검토하여 의도된 액세스에 대한 유사한 조사 결과를 확인한 다음, 해당 조사 결과를 자동으로 아카이브하도록 아카이브 규칙을 생성할 수 있습니다. 아카이브 및 확인이 완료된 결과를 검토할 수도 있습니다.

결과를 검토하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. Access analyzer(분석기 액세스)를 선택합니다.
3. 조사 결과 대시보드가 표시됩니다. 외부 액세스 분석기 또는 미사용 액세스 분석기의 활성 조사 결과를 선택합니다.

   조사 결과 대시보드 보기에 대한 자세한 내용은 IAM Access Analyzer 조사 결과 대시보드 보기 섹션을 참조하세요.

> **Note**
>
> 분석기에 대한 결과를 볼 수 있는 권한이 있는 경우에만 결과가 표시됩니다.

분석기에서는 모든 조사 결과가 표시됩니다. 분석기에서 생성된 다른 조사 결과를 보려면 상태 드롭다운에서 적절한 분석 유형을 선택합니다.

- 분석기에서 생성된 모든 활성 결과를 보려면 활성화를 선택합니다.

- 아카이브가 완료된 분석기에서 생성된 결과만 보려면 아카이브 완료를 선택합니다. 자세한 내용은 결과 아카이브을 참조하십시오.
- 확인이 완료된 분석기에서 생성된 결과만 보려면 Resolved(확인 완료)를 선택합니다. 결과를 생성한 문제를 수정하면 결과 상태가 해결됨으로 변경됩니다.

> **Important**
>
> 확인 완료된 결과는 결과를 마지막으로 업데이트하고 90일 후에 삭제됩니다. 활성화 및 아카이브 완료 상태의 결과는 결과를 생성한 분석기를 삭제하지 않는 한 삭제되지 않습니다.

- 상태에 관계 없이 분석기에서 생성된 모든 결과를 보려면 모두를 선택합니다.

외부 액세스 조사 결과

외부 액세스를 선택한 다음 분석기 보기 드롭다운에서 외부 액세스 분석기를 선택합니다. 외부 액세스 분석기에 대한 결과 페이지에는 조사 결과를 생성한 공유 리소스 및 정책 설명에 대해 다음과 같은 세부 정보가 표시됩니다.

결과 ID

결과에 할당된 고유 ID입니다. 결과를 생성한 리소스 및 정책 설명에 대한 추가적인 세부 정보를 표시하려면 결과 ID를 선택합니다.

리소스

신뢰 영역 내에 있지 않은 외부 엔터티에 액세스 권한을 부여하는 정책이 적용된 리소스의 유형 및 이름의 일부입니다.

리소스 소유자 계정

이 열은 조직을 신뢰 영역으로 사용하는 경우에만 표시됩니다. 결과에 보고된 리소스를 소유한 조직의 계정입니다.

외부 보안 주체

신뢰 영역 내에 있지 않지만 분석된 정책에서 액세스 권한을 부여하는 보안 주체입니다. 유효한 값으로는 다음이 포함됩니다.

- AWS 계정 - 해당 계정의 관리자로부터 권한을 부여받은 나열된 AWS 계정의 모든 보안 주체는 리소스에 액세스할 수 있습니다.

- 모든 보안 주체 - 조건 열에 포함된 조건을 충족하는 AWS 계정의 모든 보안 주체는 리소스에 액세스할 수 있는 권한을 가집니다. 예를 들어 VPC가 나열되어 있을 경우 나열된 VPC에 액세스할 수 있는 권한이 있는 계정의 모든 보안 주체가 리소스에 액세스할 수 있다는 뜻입니다.
- 정식 사용자 - 나열된 정식 사용자 ID를 가진 AWS 계정의 모든 보안 주체는 리소스에 액세스할 수 있는 권한을 가집니다.
- IAM 역할 - 나열된 IAM 역할은 리소스에 액세스할 수 있는 권한을 가집니다.
- IAM 사용자 - 나열된 IAM 사용자는 리소스에 액세스할 수 있는 권한을 가집니다.

Condition

액세스 권한을 부여하는 정책 설명의 조건입니다. 예를 들어 조건 필드에 Source VPC(소스 VPC)가 포함되어 있으면 나열된 VPC에 대한 액세스 권한이 있는 보안 주체와 리소스를 공유한다는 뜻입니다. 조건에는 글로벌 조건과 서비스별 조건이 있습니다. Global condition keys(전역 조건 키)에는 `aws:` 접두사가 있습니다.

Shared through(공유 방식)

Shared through(공유 방식) 필드는 결과를 생성한 액세스 권한이 어떻게 부여되는지 나타냅니다. 유효한 값으로는 다음이 포함됩니다.

- 버킷 정책 - Amazon S3 버킷에 연결된 버킷 정책입니다.
- 액세스 제어 목록 - Amazon S3 버킷에 연결된 액세스 제어 목록(ACL)입니다.
- 액세스 포인트 - Amazon S3 버킷과 연결된 액세스 포인트 또는 다중 리전 액세스 포인트입니다. 액세스 포인트의 ARN이 결과 세부 정보에 표시됩니다.

액세스 레벨

리소스 기반 정책의 작업에 의해 외부 엔터티에 부여된 액세스 수준입니다. 자세한 내용은 결과의 세부 정보를 참조하십시오. 액세스 수준 값은 다음과 같습니다.

- 목록 - 객체가 존재하는지 여부를 판단하도록 서비스 내 리소스를 나열할 수 있는 권한입니다. 이 액세스 레벨의 작업은 객체를 나열할 수 있으나 리소스의 내용을 확인할 수 없습니다.
- 읽기 - 서비스에서 리소스 내용과 속성을 읽을 수 있으나 편집할 수 없는 권한입니다.
- 쓰기 - 서비스에서 리소스를 생성, 삭제 또는 수정할 수 있는 권한입니다.
- 권한 - 서비스에서 리소스 권한을 부여하거나 수정할 수 있는 권한입니다.
- 태그 지정 - 리소스 태그의 상태를 변경만 하는 작업을 수행할 수 있는 권한입니다.

Updated

결과 상태를 가장 최근에 업데이트한 타임스탬프 또는 결과가 생성된 시간과 날짜(업데이트가 수행되지 않은 경우)입니다.

> **Note**
>
> IAM Access Analyzer에 대한 정책을 수정한 후 리소스를 다시 분석하고 결과를 업데이트하기까지 최대 30분이 걸릴 수 있습니다.

상태

결과의 상태는 활성화, 아카이브 완료, 확인 완료 중 하나입니다.

미사용 액세스 조사 결과

IAM Access Analyzer는 매월 분석된 IAM 역할 및 사용자 수를 기준으로 미사용 액세스 분석에 대해 요금을 부과합니다. 요금에 대한 자세한 내용은 IAM Access Analyzer 요금을 참조하세요.

미사용 액세스를 선택한 다음 분석기 보기 드롭다운에서 미사용 액세스 분석기를 선택합니다. 미사용 Access Analyzer에 대한 조사 결과 페이지에는 조사 결과를 생성한 IAM 엔터티에 대해 다음과 같은 세부 정보가 표시됩니다.

결과 ID

결과에 할당된 고유 ID입니다. 결과를 생성한 IAM 엔터티에 대한 추가적인 세부 정보를 표시하려면 결과 ID를 선택합니다.

찾기 유형

미사용 액세스 결과의 유형은 미사용 액세스 키, 미사용 암호, 미사용 권한 또는 미사용 역할 중 하나입니다.

IAM 엔터티

결과에 보고된 IAM 엔터티입니다. 이는 IAM 사용자 또는 역할일 수 있습니다.

AWS 계정 ID

해당 열은 조직 내 모든 AWS 계정에 대해 분석기을 설치한 경우에만 표시됩니다. 결과에 보고된 IAM 엔터티를 소유한 조직의 AWS 계정입니다.

최종 업데이트 날짜

결과에 보고된 IAM 엔터티가 마지막으로 업데이트된 시간 또는 업데이트가 없는 경우 엔터티가 생성된 시점입니다.

상태

결과의 상태는 활성, 아카이브됨, 해결됨 중 하나입니다.

조사 결과 필터링

조사 결과 페이지의 기본 필터링은 모든 조사 결과를 표시하는 것입니다. 활성 조사 결과를 보려면 상태 드롭다운에서 활성 상태를 선택합니다. 아카이브된 조사 결과를 보려면 상태 드롭다운에서 아카이브됨 상태를 선택합니다. IAM Access Analyzer를 처음 사용하기 시작할 때는 아카이브된 검색 결과가 없습니다.

필터를 사용하여 지정된 속성 기준을 충족하는 조사 결과만 표시할 수 있습니다. 필터를 생성하려면 필터링할 속성을 선택한 다음 해당 속성이 값과 동일한 경우 또는 포함하는 경우 중에서 선택하고 필터링할 속성 값을 입력하거나 선택합니다. 예를 들어 특정 AWS 계정에 대한 조사 결과만 표시하는 필터를 생성하려면 속성에 대해 AWS 계정을 선택한 다음, AWS 계정 =을 선택하고, 조사 결과를 보려는 AWS 계정의 계정 번호를 입력합니다.

아카이브 규칙을 생성하거나 업데이트하는 데 사용할 수 있는 필터 키 목록은 IAM Access Analyzer 필터 키 단원을 참조하세요.

외부 액세스 조사 결과 필터링

외부 액세스 조사 결과를 필터링하려면

1. 외부 액세스를 선택한 다음 분석기 보기 드롭다운에서 분석기를 선택합니다.
2. 검색 상자를 선택하면 사용 가능한 속성 목록이 표시됩니다.
3. 표시된 결과를 필터링하는 데 사용할 속성을 선택합니다.
4. 속성에 대해 일치시킬 값을 선택합니다. 결과에 해당 값을 가진 결과만 표시됩니다.

   예를 들어 속성으로 리소스를 선택한 후 리소스:를 선택하고, 버킷 이름의 일부 또는 전체를 입력한 다음 Enter 키를 누릅니다. 필터 기준과 일치하는 버킷에 대한 조사 결과만 표시됩니다. 퍼블릭 액세스를 허용하는 리소스에 대한 조사 결과만 표시하는 필터를 생성하려면 퍼블릭 액세스 속성을 선택한 다음 퍼블릭 액세스 =을 선택하고 퍼블릭 액세스 = true를 선택합니다.

속성을 추가하여 표시된 결과를 추가로 필터링할 수 있습니다. 속성을 추가하면 필터의 모든 조건과 일치하는 결과만 표시됩니다. 특정 속성 또는 다른 속성과 일치하는 결과를 표시하도록 필터를 정의하는 것은 지원되지 않습니다. 필터 지우기를 선택하여 정의한 필터를 모두 지우고 분석기에 지정된 상태와 함께 모든 조사 결과를 표시합니다.

일부 필드는 조직을 신뢰 영역으로 하는 분석기의 결과를 보는 경우에만 표시됩니다.

다음 속성은 필터를 정의하는 데 사용할 수 있습니다.

- 퍼블릭 액세스 - 퍼블릭 액세스를 허용하는 리소스에 대한 결과를 기준으로 필터링하려면 퍼블릭 액세스를 기준으로 필터링한 다음 Public access: true를 선택합니다.
- 리소스 - 리소스를 기준으로 필터링하려면 리소스 이름의 전체 또는 일부를 입력합니다.
- 리소스 유형 - 리소스 유형을 기준으로 필터링하려면 표시된 목록에서 유형을 선택합니다.
- 리소스 소유자 계정 - 결과에 보고된 리소스를 소유한 조직의 계정별로 필터링하려면 이 속성을 사용합니다.
- AWS 계정 - 이 속성을 사용하여 정책 설명의 보안 주체 섹션에서 액세스 권한이 부여된 AWS 계정별로 필터링합니다. AWS 계정별로 필터링하려면 12자리 AWS 계정 ID의 전부 또는 일부를 입력하거나, 현재 계정의 리소스에 액세스할 수 있는 외부 AWS 사용자 또는 역할의 전체 계정 ARN의 전부 또는 일부를 입력합니다.
- 정식 사용자 - 정식 사용자를 기준으로 필터링하려면 Amazon S3 버킷에 정의된 정식 사용자 ID를 입력합니다. 자세한 내용은 AWS 계정 식별자를 참조하세요.
- 페더레이션 사용자 - 페더레이션 사용자를 기준으로 필터링하려면 연동 ID의 ARN 전체 또는 일부를 입력합니다. 자세한 내용은 ID 공급자 및 연동을 참조하십시오.
- 조사 결과 ID - 조사 결과 ID를 기준으로 필터링하려면 조사 결과 ID의 전체 또는 일부를 입력합니다.
- 보안 주체 ARN - 이 속성을 사용하여 aws:PrincipalArn 조건 키에 사용되는 보안 주체(IAM 사용자, 역할 또는 그룹)의 ARN을 필터링합니다. 보안 주체 ARN을 기준으로 필터링하려면 조사 결과에 보고된 외부 AWS 계정에서 IAM 사용자, 역할 또는 그룹의 ARN 전체 또는 일부를 입력합니다.
- 보안 주체 OrgID - 보안 주체 OrgID를 기준으로 필터링하려면 결과에 조건으로 지정된 AWS 조직에 속하는 외부 보안 주체와 연결된 조직 ID 전체 또는 일부를 입력합니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.
- 보안 주체 조직 경로 - 보안 주체 조직 경로를 기준으로 필터링하려면 정책의 조건으로 지정된 조직 또는 조직 단위(OU)의 계정 구성원인 모든 외부 보안 주체에 액세스할 수 있는 AWS 조직 또는 OU의 ID 전체 또는 일부를 입력합니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.
- 소스 계정 - 소스 계정을 기준으로 필터링하려면 AWS의 일부 교차 서비스 권한에서 사용된 대로 리소스와 연결된 AWS 계정 ID 전체 또는 일부를 입력합니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.
- 소스 ARN - 소스 ARN을 기준으로 필터링하려면 결과에 조건으로 지정된 ARN 전체 또는 일부를 입력합니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.

- 소스 IP - 소스 IP를 기준으로 필터링하려면 지정된 IP 주소를 사용할 때 외부 엔터티가 현재 계정의 리소스에 액세스할 수 있도록 허용하는 IP 주소의 전체 또는 일부를 입력합니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.
- 소스 VPC - 소스 VPC를 기준으로 필터링하려면 지정된 VPC를 사용할 때 외부 엔터티가 현재 계정의 리소스에 액세스할 수 있도록 허용하는 VPC ID의 전체 또는 일부를 입력합니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.
- 소스 OrgID - 소스 OrgID를 기준으로 필터링하려면 AWS의 일부 교차 서비스 권한에서 사용된 대로 리소스와 연결된 조직 ID 전체 또는 일부를 입력합니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.
- 소스 OrgPaths - 소스 OrgPaths를 기준으로 필터링하려면 AWS의 일부 교차 서비스 권한에서 사용된 대로 리소스와 연결된 OU(조직 단위) 전체 또는 일부를 입력합니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.
- 사용자 ID - 사용자 ID를 기준으로 필터링하려면 현재 계정의 리소스에 대한 액세스가 허용되는 외부 AWS 계정에서 IAM 사용자의 ID 전체 또는 일부를 입력합니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.
- KMS 키 ID - KMS 키 ID를 기준으로 필터링하려면 현재 계정에서 AWS KMS 암호화 Amazon S3 객체 액세스 조건으로 지정된 KMS 키의 ID 전체 또는 일부를 입력합니다.
- Google 대상 - Google 대상을 기준으로 필터링하려면 현재 계정에서 IAM 역할 액세스 조건으로 지정된 Google 애플리케이션 ID의 전체 또는 일부를 입력합니다. 자세한 내용은 IAM 및 AWS STS 조건 컨텍스트 키를 참조하세요.
- Cognito 대상 - Amazon Cognito 대상을 기준으로 필터링하려면 현재 계정에서 IAM 역할 액세스 조건으로 지정된 Amazon Cognito 자격 증명 풀 ID의 전체 또는 일부를 입력합니다. 자세한 내용은 IAM 및 AWS STS 조건 컨텍스트 키를 참조하세요.
- 호출자 계정 - IAM 역할, 사용자 또는 계정 루트 사용자와 같이 호출 엔터티를 소유하거나 포함하는 계정의 AWS 계정 ID입니다. 이 기능은 AWS KMS를 호출하는 서비스에서 사용됩니다. 호출자 계정을 기준으로 필터링하려면 AWS 계정 ID의 전체 또는 일부를 입력합니다.
- Facebook 앱 ID - Facebook 앱 ID를 기준으로 필터링하려면 현재 계정의 IAM 역할에 대한 Facebook 연동 액세스 권한을 이용하여 로그인할 수 있도록 조건으로 지정된 Facebook 애플리케이션 ID(또는 사이트 ID)의 전체 또는 일부를 입력합니다. 자세한 내용은 IAM 및 AWS STS 조건 컨텍스트 키의 id 섹션을 참조하세요.
- Amazon 앱 ID - Amazon 앱 ID를 기준으로 필터링하려면 현재 계정의 IAM 역할에 대한 Login with Amazon 연동 액세스를 허용하도록 조건이 지정된 Amazon 애플리케이션 ID(또는 사이트 ID)의 전체 또는 일부를 입력합니다. 자세한 내용은 IAM 및 AWS STS 조건 컨텍스트 키의 id 섹션을 참조하세요.

- Lambda 이벤트 소스 토큰 - Alexa 통합 시 전달된 Lambda 이벤트 소스 토큰을 기준으로 필터링하려면 토큰 문자열 전체 또는 일부를 입력합니다.

미사용 액세스 조사 결과를 필터링하기

미사용 액세스 조사 결과를 필터링하려면

1. 미사용 액세스를 선택한 다음 분석기 보기 드롭다운에서 분석기를 선택합니다.
2. 검색 상자를 선택하면 사용 가능한 속성 목록이 표시됩니다.
3. 표시된 결과를 필터링하는 데 사용할 속성을 선택합니다.
4. 속성에 대해 일치시킬 값을 선택합니다. 결과에 해당 값을 가진 결과만 표시됩니다.

   예를 들어, 조사 결과 유형을 속성으로 선택한 후, 조사 결과 유형 =을 선택한 다음 조사 결과 유형 = UnusedIAMRole을 선택하면 UnusedIAMRole 유형의 조사 결과만 표시됩니다.

속성을 추가하여 표시된 결과를 추가로 필터링할 수 있습니다. 속성을 추가하면 필터의 모든 조건과 일치하는 결과만 표시됩니다. 특정 속성 또는 다른 속성과 일치하는 결과를 표시하도록 필터를 정의하는 것은 지원되지 않습니다. 필터 지우기를 선택하여 정의한 필터를 모두 지우고 분석기에 지정된 상태와 함께 모든 조사 결과를 표시합니다.

미사용 액세스를 모니터링하는 분석기의 조사 결과를 보는 경우에만 다음 필드가 표시됩니다.

- 조사 결과 유형 - 조사 결과 유형별로 필터링하려면 조사 결과 유형별로 필터링한 다음, 조사 결과 유형을 선택합니다.
- 리소스 - 리소스를 기준으로 필터링하려면 리소스 이름의 전체 또는 일부를 입력합니다.
- 리소스 유형 - 리소스 유형을 기준으로 필터링하려면 표시된 목록에서 유형을 선택합니다.
- 리소스 소유자 계정 - 조사 결과에 보고된 리소스를 소유한 조직의 계정별로 필터링하려면 이 속성을 사용합니다.
- 조사 결과 ID - 조사 결과 ID를 기준으로 필터링하려면 조사 결과 ID의 전체 또는 일부를 입력합니다.

결과 아카이브

의도적인 리소스 액세스에 대한 조사 결과를 얻으면 조사 결과를 아카이브할 수 있습니다. 승인된 워크플로에서 여러 명이 사용하는 IAM 역할의 외부 액세스 분석이나 필요할 수 있는 액세스 키에 대한 미사용 액세스 조사 결과를 예로 들 수 있습니다. 조사 결과를 보관하면 활성 조사 결과 목록에서 삭제됩니

다. 보관된 결과는 삭제되지 않습니다. 아카이브된 조사 결과를 표시하도록 조사 결과 페이지를 필터링할 수 있고, 언제든지 아카이브를 취소할 수 있습니다.

결과 페이지에서 결과를 아카이브하려면

1. 아카이브할 하나 이상의 결과 옆에 있는 확인란을 선택합니다.
2. 작업을 선택한 후 아카이브를 선택합니다.

   화면 상단에 확인 메시지가 표시됩니다.

조사 결과 세부 정보 페이지에서 조사 결과를 아카이브하려면

1. 아카이브할 결과에 대한 Finding ID(결과 ID)를 선택합니다.
2. Archive(아카이브)를 선택합니다.

   화면 상단에 확인 메시지가 표시됩니다.

결과의 아카이브를 해제하려면 앞의 단계를 반복하되 아카이브(Archive) 대신 아카이브 해제(Unarchive)를 선택합니다. 결과의 아카이브를 해제하면 상태가 활성으로 설정됩니다.

IAM Access Analyzer 조사 결과 해결

외부 액세스 조사 결과 해결

의도하지 않은 액세스로부터 생성된 외부 액세스 조사 결과를 확인하려면 식별된 리소스에 대한 액세스를 허용하는 권한을 제거하도록 정책문을 수정해야 합니다.

Amazon S3 버킷에 대한 조사 결과의 경우 Amazon S3 콘솔을 사용하여 버킷에 대한 권한을 구성합니다.

IAM 역할의 경우 IAM 콘솔을 사용하여 나열된 IAM 역할에 대해 신뢰 정책을 수정합니다.

다른 지원되는 리소스의 경우 콘솔을 사용하여 생성된 조사 결과를 초래한 정책문을 수정합니다.

IAM 역할에 적용된 정책을 수정하는 등 외부 액세스 조사 결과를 확인하기 위해 변경을 수행한 후 IAM Access Analyzer에서 리소스를 다시 검색합니다. 리소스가 신뢰 영역 외부에서 더 이상 공유되지 않으면 결과 상태가 해결됨으로 변경됩니다. 그러면 조사 결과는 활성 조사 결과 목록 대신에 해결된 조사 결과 목록에 표시됩니다.

> **Note**
>
> 오류 조사 결과에는 적용되지 않습니다. IAM Access Analyzer가 리소스를 분석할 수 없는 경우 오류 조사 결과가 생성됩니다. IAM Access Analyzer에서 리소스를 분석할 수 없는 문제를 해결하면, 해결된 조사 결과로 변경되는 대신, 오류 조사 결과가 완전히 제거됩니다.

수행한 변경으로 인해 리소스가 신뢰 영역 외부에서 공유되지만 다른 보안 주체 또는 다른 권한에서와 같이 다른 방식으로 공유되는 경우, IAM Access Analyzer에서 활성 조사 결과가 새로 생성됩니다.

> **Note**
>
> IAM Access Analyzer에 대한 정책을 수정한 후 리소스를 다시 분석하고 결과를 업데이트하기까지 최대 30분이 걸릴 수 있습니다. 확인 완료된 결과는 결과 상태를 마지막으로 업데이트하고 90일 후에 삭제됩니다.

미사용 액세스 조사 결과 해결

IAM Access Analyzer는 조사 결과 유형에 따라 미사용 액세스 분석기 조사 결과 해결을 위한 권장 단계를 제공합니다.

미사용 액세스 결과를 확인하기 위해 변경한 후에는 추후 미사용 액세스 분석기를 실행할 때 결과의 상태가 해결됨으로 변경됩니다. 조사 결과는 더 이상 활성 조사 결과 목록에 표시되지 않고 대신에 해결된 조사 결과 목록에 표시됩니다. 미사용 액세스 검색을 일부만 처리하는 내용을 변경하는 경우 기존 결과는 해결됨으로 변경되지만 새 결과가 생성됩니다. 예를 들어 조사 결과의 미사용 권한을 전부가 아니라 일부만 제거합니다.

IAM Access Analyzer는 매월 분석된 IAM 역할 및 사용자 수를 기준으로 미사용 액세스 분석에 대해 요금을 부과합니다. 요금에 대한 자세한 내용은 IAM Access Analyzer 요금을 참조하세요.

미사용 권한 조사 결과 해결

미사용 권한 조사 결과의 경우 IAM Access Analyzer는 IAM 사용자 또는 역할에서 제거할 정책을 권장하고 기존 권한 정책을 대체할 새 정책을 제공할 수 있습니다. 다음 시나리오에는 정책 권장 사항이 지원되지 않습니다.

- 미사용 권한 조사 결과가 사용자 그룹에 속한 IAM 사용자에 대한 것입니다.
- 미사용 권한 조사 결과가 IAM Identity Center의 IAM 역할에 대한 것입니다.

- 미사용 권한 조사 결과에 `notAction` 요소가 포함된 기존 권한 정책이 있습니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 미사용 액세스를 선택합니다.
3. 결과 유형이 미사용 권한인 조사 결과를 선택합니다.
4. 권장 사항 섹션에서 권장 정책 열에 나열된 정책이 있는 경우 정책 미리 보기를 선택하여 기존 정책을 대체할 권장 정책과 함께 기존 정책을 볼 수 있습니다. 권장 정책이 여러 개 있는 경우 다음 정책과 이전 정책을 선택하여 기존 정책과 권장 정책을 각각 볼 수 있습니다.
5. JSON 다운로드를 선택하여 모든 권장 정책의 JSON 파일이 들어 있는 .zip 파일을 다운로드합니다.
6. 권장 정책을 생성하고 IAM 사용자 또는 역할에 연결합니다. 자세한 내용은 사용자 권한 변경(콘솔) 및 역할 권한 정책 수정(콘솔)을 참조하세요.
7. 기존 권한 정책 열에 나열된 정책을 IAM 사용자 또는 역할에서 제거합니다. 자세한 내용은 사용자 권한 제거(콘솔) 및 역할 권한 정책 수정(콘솔)을 참조하세요.

미사용 역할 조사 결과 해결

미사용 역할 조사 결과의 경우 IAM Access Analyzer는 미사용 IAM 역할을 삭제할 것을 권장합니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 미사용 액세스를 선택합니다.
3. 결과 유형이 미사용 역할인 조사 결과를 선택합니다.
4. 권장 사항 섹션에서 IAM 역할의 세부 정보를 검토합니다.
5. IAM 역할을 삭제합니다. 자세한 내용은 IAM 역할 삭제(콘솔)를 참조하세요.

미사용 액세스 키 조사 결과 해결

미사용 액세스 키 조사 결과의 경우 IAM Access Analyzer는 미사용 액세스 키를 비활성화하거나 삭제할 것을 권장합니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 미사용 액세스를 선택합니다.
3. 결과 유형이 미사용 액세스 키인 조사 결과를 선택합니다.
4. 권장 사항 섹션에서 액세스 키의 세부 정보를 검토합니다.

5. 액세스 키를 비활성화 또는 삭제합니다. 자세한 내용은 액세스 키 관리(콘솔)를 참조하세요.

미사용 암호 조사 결과 해결

미사용 암호 조사 결과의 경우 IAM Access Analyzer는 IAM 사용자의 미사용 암호를 삭제할 것을 권장합니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 미사용 액세스를 선택합니다.
3. 결과 유형이 미사용 암호인 조사 결과를 선택합니다.
4. 권장 사항 섹션에서 IAM 사용자의 세부 정보를 검토합니다.
5. IAM 사용자의 암호를 삭제합니다. 자세한 내용은 IAM 사용자 암호 생성, 변경 또는 삭제(콘솔)를 참조하세요.

외부 액세스를 위한 IAM Access Analyzer 리소스 유형

외부 액세스 분석기의 경우 IAM Access Analyzer에서는 IAM Access Analyzer를 활성화한 리전의 AWS 리소스에 적용되는 리소스 기반 정책을 분석합니다. 리소스 기반 정책만 분석됩니다. IAM Access Analyzer에서 각 리소스 유형에 대한 결과를 생성하는 방법을 자세히 알아보려면 각 리소스에 대한 정보를 검토합니다.

> **Note**
>
> 나열된 리소스 유형은 외부 액세스 분석기를 지원합니다. 미사용 액세스 분석기는 IAM 사용자 및 역할만 지원합니다. 자세한 내용은 조사 결과 작업 단원을 참조하십시오.

외부 액세스에 대해 지원하는 리소스 유형:

- Amazon Simple Storage Service 버킷
- Amazon Simple Storage Service 디렉터리 버킷
- AWS Identity and Access Management 역할
- AWS Key Management Service 키
- AWS Lambda 함수 및 계층
- Amazon Simple Queue Service 대기열
- AWS Secrets Manager 보안 암호

- Amazon Simple Notification Service(Amazon SNS) 주제
- Amazon Elastic Block Store 볼륨 스냅샷
- Amazon Relational Database Service DB 스냅샷
- Amazon Relational Database Service DB 클러스터 스냅샷
- Amazon Elastic 컨테이너 레지스트리 리포지토리
- Amazon Elastic File System 파일 시스템
- Amazon DynamoDB Streams
- Amazon DynamoDB 테이블

Amazon Simple Storage Service 버킷

IAM Access Analyzer는 Amazon S3 버킷을 분석할 때 버킷에 적용된 S3 버킷 정책, ACL 또는 액세스 포인트(다중 리전 액세스 포인트 포함)가 외부 엔터티에 액세스 권한을 부여할 때 결과를 생성합니다. 외부 엔터티는 신뢰 영역 내에 없는 필터를 생성하는 데 사용할 수 있는 보안 주체 또는 기타 엔터티입니다. 예를 들어 버킷 정책이 다른 계정에 액세스 권한을 부여하거나 퍼블릭 액세스를 허용하는 경우, IAM Access Analyzer에서 결과를 생성합니다. 그러나 버킷에서 Block Public Access(퍼블릭 액세스 차단)를 활성화하면 계정 수준 또는 버킷 수준에서 액세스를 차단할 수 있습니다.

> **Note**
>
> 액세스 포인트와 해당 정책이 분석기 계정 외부에 있기 때문에 IAM Access Analyzer는 크로스 계정 액세스 포인트에 연결된 액세스 포인트 정책을 분석하지 않습니다. IAM Access Analyzer는 버킷이 크로스 계정 액세스 포인트에 대한 액세스를 위임하고 버킷 또는 계정에서 퍼블릭 액세스 차단이 활성화되지 않은 경우 공개 결과를 생성합니다. 퍼블릭 액세스 차단을 활성화하면 공개 결과가 해결되고 IAM Access Analyzer가 크로스 계정 액세스 포인트에 대한 크로스 계정 결과를 생성합니다.

Amazon S3 Block Public Access(퍼블릭 액세스 차단) 설정은 버킷에 적용되는 버킷 정책을 재정의합니다. 이 설정은 버킷의 액세스 포인트에 적용되는 액세스 포인트 정책도 재정의합니다. IAM Access Analyzer는 정책이 변경될 때마다 버킷 수준에서 Block Public Access(퍼블릭 액세스 차단) 설정을 분석합니다. 하지만 6시간마다 한 번씩만 계정 수준에서 Block Public Access(퍼블릭 액세스 차단) 설정을 평가합니다. 즉, IAM Access Analyzer에서 버킷에 대한 퍼블릭 액세스의 결과를 최대 6시간 동안 생성 또는 확인하지 못할 수 있습니다. 예를 들어 퍼블릭 액세스를 허용하는 버킷 정책이 있는 경우, IAM Access Analyzer에서 해당 액세스에 대한 결과를 생성합니다. Block Public Access(퍼블릭 액세스 차

단)를 활성화하여 계정 수준에서 버킷에 대한 모든 퍼블릭 액세스를 차단하면 버킷에 대한 모든 퍼블릭 액세스가 차단되더라도 IAM Access Analyzer에서 최대 6시간 동안 버킷 정책에 대한 결과를 확인하지 못합니다. 계정 수준에서 퍼블릭 액세스 차단을 활성화하면 크로스 계정 액세스 포인트에 대한 공개 결과를 확인하는 데 최대 6시간이 걸릴 수 있습니다.

다중 리전 액세스 포인트의 경우 IAM Access Analyzer는 정해진 정책을 사용하여 결과를 생성합니다. IAM Access Analyzer는 6시간마다 한 번씩 다중 리전 액세스 포인트의 변경 사항을 평가합니다. 즉, 다중 리전 액세스 포인트를 생성 또는 삭제하거나 관련 정책을 업데이트하는 경우에도 IAM Access Analyzer에서 최대 6시간 동안 결과를 생성하거나 확인하지 않습니다.

Amazon Simple Storage Service 디렉터리 버킷

Amazon S3 디렉터리 버킷은 Amazon S3 Express One 스토리지 클래스를 사용하며, 이는 성능이 중요한 워크로드 또는 애플리케이션에 권장됩니다. Amazon S3 디렉터리 버킷의 경우 IAM Access Analyzer는 외부 엔터티가 디렉터리 버킷에 액세스할 수 있도록 허용하는 정책의 조건문을 비롯하여 디렉터리 버킷 정책을 분석합니다. Amazon S3 디렉터리 버킷에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 디렉터리 버킷을 참조하세요.

AWS Identity and Access Management 역할

IAM 역할의 경우 IAM Access Analyzer에서는 신뢰 정책을 분석합니다. 역할 신뢰 정책에서 역할을 수임하기 위해 신뢰하는 보안 주체를 정의합니다. 역할 신뢰 정책은 IAM의 역할에 연결된 필수 리소스 기반 정책입니다. IAM Access Analyzer는 신뢰 영역 밖의 외부 엔터티가 액세스할 수 있는 신뢰 영역 내의 역할에 결과를 생성합니다.

> **Note**
>
> IAM 역할은 글로벌 리소스입니다. 역할 신뢰 정책에서 외부 엔터티에 액세스 권한을 부여하는 경우, IAM Access Analyzer가 활성화된 각 리전에서 결과를 생성합니다.

AWS Key Management Service 키

AWS KMS keys의 경우 IAM Access Analyzer는 키에 적용되는 키 정책 및 권한 부여를 분석합니다. IAM Access Analyzer는 키 정책 또는 권한 부여로 외부 엔터티가 키에 액세스할 수 있는 경우 결과를 생성합니다. 예를 들어 정책 문에서 kms:CallerAccount 조건 키를 사용하여 특정 AWS 계정의 모든 사용자에게 액세스를 허용하고 현재 계정(현재 분석기의 신뢰 영역)이 아닌 다른 계정을 지정하면 IAM Access Analyzer에서 결과를 생성합니다. IAM 정책 문의 AWS KMS 조건 키에 대한 자세한 내용을 알아보려면 AWS KMS 조건 키를 참조하세요.

IAM Access Analyzer는 KMS 키를 분석할 때 키 정책 및 권한 부여 목록과 같은 주요 메타데이터를 읽습니다. 키 정책에서 키 메타데이터를 읽을 수 없는 IAM Access Analyzer 역할을 허용하지 않는 경우 액세스 거부 오류 결과가 생성됩니다. 예를 들어 다음과 같은 예제 정책 문이 키에 적용되는 유일한 정책인 경우, IAM Access Analyzer에서 액세스 거부 오류 결과가 생성됩니다.

```
{
    "Sid": "Allow access for Key Administrators",
    "Effect": "Allow",
    "Principal": {
       "AWS": "arn:aws:iam::111122223333:role/Admin"
    },
    "Action": "kms:*",
    "Resource": "*"
}
```

이 문은 AWS 계정 111122223333의 Admin이라는 역할만 키에 액세스할 수 있도록 허용하므로 IAM Access Analyzer에서 키를 완전히 분석할 수 없으므로 액세스 거부 오류 결과가 생성됩니다. 오류 결과는 Findings(결과) 테이블에 빨간색 텍스트로 표시됩니다. 결과는 다음과 비슷합니다.

```
{
    "error": "ACCESS_DENIED",
    "id": "12345678-1234-abcd-dcba-111122223333",
    "analyzedAt": "2019-09-16T14:24:33.352Z",
    "resource": "arn:aws:kms:us-
west-2:1234567890:key/1a2b3c4d-5e6f-7a8b-9c0d-1a2b3c4d5e6f7g8a",
    "resourceType": "AWS::KMS::Key",
    "status": "ACTIVE",
    "updatedAt": "2019-09-16T14:24:33.352Z"
}
```

KMS 키를 생성할 때 키에 액세스할 수 있도록 부여된 권한은 키를 생성하는 방법에 따라 다릅니다. 키 리소스에 대해 액세스 거부 오류 결과를 수신하는 경우, IAM Access Analyzer에 키에 액세스할 수 있는 권한을 부여하기 위해 키 리소스에 다음과 같은 정책 문을 적용합니다.

```
{
    "Sid": "Allow IAM Access Analyzer access to key metadata",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/aws-service-role/access-
analyzer.amazonaws.com/AWSServiceRoleForAccessAnalyzer"
        },
```

```
    "Action": [
        "kms:DescribeKey",
        "kms:GetKeyPolicy",
        "kms:List*"
    ],
    "Resource": "*"
},
```

KMS 키 리소스에 대한 액세스 거부 결과를 수신한 다음 키 정책을 업데이트하여 결과를 확인하면 결과가 확인 완료 상태로 업데이트됩니다. 외부 엔터티에 키에 대한 권한을 부여하는 정책 설명 또는 키 부여가 있는 경우, 키 리소스에 대한 추가 결과가 표시될 수 있습니다.

AWS Lambda 함수 및 계층

AWS Lambda 함수의 경우 IAM Access Analyzer는 외부 엔터티에 함수에 대한 액세스 권한을 부여하는 정책의 조건 문을 비롯해 정책을 분석합니다. Lambda를 사용하면 고유한 리소스 기반 정책을 함수, 버전, 별칭 및 계층에 연결할 수 있습니다. IAM Access Analyzer는 함수와 계층에 연결된 리소스 기반 정책을 기반으로 외부 액세스를 보고합니다. IAM Access Analyzer는 검증된 ARN을 사용하여 간접적으로 호출된 별칭 및 특정 버전에 연결된 리소스 기반 정책을 기반으로 외부 액세스를 보고하지 않습니다.

자세한 내용은 AWS Lambda 개발자 안내서의 Using resource-based policies for Lambda와 Using versions를 참조하세요.

Amazon Simple Queue Service 대기열

Amazon SQS 대기열의 경우 IAM Access Analyzer는 외부 엔터티가 대기열에 액세스할 수 있도록 허용하는 정책의 조건 문을 비롯해 정책을 분석합니다.

AWS Secrets Manager 보안 암호

AWS Secrets Manager 암호의 경우, IAM Access Analyzer는 외부 엔터티가 암호에 액세스할 수 있도록 허용하는 정책을 조건 문을 비롯해 정책을 분석합니다.

Amazon Simple Notification Service(Amazon SNS) 주제

IAM Access Analyzer는 Amazon SNS 주제와 연결된 리소스 기반 정책으로, 주제에 대한 외부 액세스를 허용하는 정책에서의 조건문을 포함합니다. 외부 계정이 Amazon SNS 작업(예: 리소스 기반 정책을 통해 주제 구독 및 게시)을 수행하도록 허용할 수 있습니다. An Amazon SNS 주제는 신뢰 영역 외부의 계정에 있는 보안 주체가 주제에 대한 작업을 수행할 수 있을 경우 외부에서 액세스가 가능합니다. Amazon SNS 주제를 생성할 때 정책에서 `Everyone`을 선택할 경우, 공개적으로 주제에 액세스할 수

있습니다. `AddPermission`은 외부 액세스를 허용하는 Amazon SNS 주제에 대해 리소스 기반 정책을 추가하는 또 다른 수단입니다.

Amazon Elastic Block Store 볼륨 스냅샷

Amazon Elastic Block Store 볼륨 스냅샷은 리소스 기반 정책이 없습니다. 스냅샷은 Amazon EBS 공유 권한을 통해 공유됩니다. Amazon EBS 볼륨 스냅샷의 경우, IAM Access Analyzer가 외부 엔터티의 스냅샷 액세스를 허용하는 액세스 제어 목록을 분석합니다. Amazon EBS 볼륨 스냅샷은 암호화될 경우 외부 계정과 공유할 수 있습니다. 암호화되지 않은 볼륨 스냅샷을 외부 계정과 공유하고 공개 액세스 권한을 부여할 수 있습니다. 공유 설정은 스냅샷의 `CreateVolumePermissions` 속성에 있습니다. 고객이 Amazon EBS 스냅샷의 외부 액세스를 미리 확인할 경우, 스냅샷이 암호화되었다는 지표로 암호화 키를 지정할 수 있습니다. 이는 IAM Access Analyzer 미리 보기가 Secrets Manager 보안 암호를 처리하는 방식과 유사합니다.

Amazon Relational Database Service DB 스냅샷

Amazon RDS DB 스냅샷은 리소스 기반 정책이 없습니다. DB 스냅샷은 Amazon RDS 데이터베이스 권한을 통해 공유되고, 수동 DB 스냅샷만 공유할 수 있습니다. Amazon RDS DB 스냅샷의 경우, IAM Access Analyzer가 외부 엔터티의 스냅샷 액세스를 허용하는 액세스 제어 목록을 분석합니다. 암호화되지 않은 DB 스냅샷은 전체적으로 공개할 수 있습니다. 암호화된 DB 스냅샷은 공개적으로 공유할 수 없지만 최대 20개의 다른 계정과 공유할 수 있습니다. 자세한 내용은 DB 스냅샷 생성을 참조하십시오. IAM Access Analyzer는 데이터베이스 수동 스냅샷을 (Amazon S3 버킷 등에 대해) 신뢰할 수 있는 액세스 권한으로 내보내는 기능을 고려합니다.

> **Note**
>
> IAM Access Analyzer는 데이터베이스 자체에서 직접 구성된 공개 또는 계정 간 액세스는 식별하지 않습니다. IAM Access Analyzer는 Amazon RDS DB 스냅샷에서 구성된 공개 또는 계정 간 액세스에 대한 결과만 식별합니다.

Amazon Relational Database Service DB 클러스터 스냅샷

Amazon RDS DB 클러스터 스냅샷은 리소스 기반 정책이 없습니다. 스냅샷은 Amazon RDS DB 클러스터 권한을 통해 공유됩니다. Amazon RDS DB 클러스터 스냅샷의 경우, IAM Access Analyzer가 외부 엔터티의 스냅샷 액세스를 허용하는 액세스 제어 목록을 분석합니다. 암호화되지 않은 클러스터 스냅샷은 전체적으로 공개할 수 있습니다. 암호화된 클러스터 스냅샷은 공개적으로 공유할 수 없습니다. 암호화되지 않은 클러스터 스냅샷과 암호화된 클러스터 스냅샷은 최대 20개의 다른 계정과 공유할 수

있습니다. 자세한 내용은 DB 클러스터 스냅샷 생성을 참조하세요. IAM Access Analyzer는 DB 클러스터 스냅샷을 (Amazon S3 버킷 등에 대해) 신뢰할 수 있는 액세스 권한으로 내보내는 기능을 고려합니다.

> **Note**
>
> IAM Access Analyzer는 AWS Resource Access Manager을 사용하는 다른 AWS 계정 또는 조직과의 Amazon DB 클러스터 및 클론의 공유를 모니터링 결과를 포함하지 않습니다. IAM Access Analyzer는 Amazon RDS DB 클러스터 스냅샷에서 구성된 공개 또는 계정 간 액세스에 대한 결과만 식별합니다.

Amazon Elastic 컨테이너 레지스트리 리포지토리

Amazon ECR 리포지토리의 경우, IAM Access Analyzer는 외부 엔터티의 리포지토리 액세스를 허용하는 리소스 기반 정책을 분석합니다(정책의 조건문 포함). (이는 Amazon SNS 주제 및 Amazon EFS 파일 시스템 등의 다른 리소스와 유사합니다). Amazon ECR 리포지토리의 경우 보안 주체가 외부에서 사용할 수 있으려면 아이덴티티 기반 정책을 통해 `ecr:GetAuthorizationToken`에 대한 권한을 부여받아야 합니다.

Amazon Elastic File System 파일 시스템

Amazon EFS 파일 시스템의 경우 IAM Access Analyzer는 외부 엔터티가 파일 시스템에 액세스할 수 있도록 허용하는 정책의 조건 문을 비롯해 정책을 분석합니다. Amazon EFS 파일 시스템은 린쇠 영역 외부의 계정에서 보안 주체가 해당 파일 시스템에 작업을 수행할 수 있을 경우 외부에서 액세스가 가능합니다. 액세스는 IAM을 사용하는 파일 정책 시스템, 그리고 파일 시스템을 마운트하는 방식에 따라 정의됩니다. 예를 들어 다른 계정에 Amazon EFS 파일 시스템을 마운팅하는 것은 외부에서 액세스하는 것으로 간주됩니다. 단, 계정이 조직에 속해 있고 조직을 신뢰 영역으로 정의했어야 합니다. 퍼블릭 서브넷이 있는 Virtual Private Cloud에서 파일 시스템을 마운팅하는 경우, 파일 시스템을 외부에서 액세스할 수 있습니다. Amazon EFS를 AWS Transfer Family과 함께 사용할 경우, 파일 시스템이 아닌 다른 계정에서 소유한 Transfer Family 서버에서 받은 시스템 액세스 요청은 파일 시스템에서 공개 액세스를 허용할 경우 차단됩니다.

Amazon DynamoDB Streams

DynamoDB 정책에서 외부 엔터티가 DynamoDB 스트림에 액세스할 수 있는 크로스 계정 작업을 하나 이상 허용하는 경우 IAM Access Analyzer는 결과를 생성합니다. DynamoDB에서 지원되는 크로스 계정 작업에 대한 자세한 내용은 Amazon DynamoDB 개발자 가이드의 리소스 기반 정책에서 지원되는 IAM 작업을 참조하세요.

Amazon DynamoDB 테이블

DynamoDB 정책에서 외부 엔터티가 DynamoDB 테이블 또는 인덱스에 액세스할 수 있는 크로스 계정 작업을 하나 이상 허용하는 경우 IAM Access Analyzer는 DynamoDB 테이블에 대한 결과를 생성합니다. DynamoDB에서 지원되는 크로스 계정 작업에 대한 자세한 내용은 Amazon DynamoDB 개발자 가이드의 리소스 기반 정책에서 지원되는 IAM 작업을 참조하세요.

IAM Access Analyzer 설정

AWS Organizations 관리 계정에서 AWS Identity and Access Management Access Analyzer를 구성하는 경우 조직의 멤버 계정을 위임된 관리자로 추가하여 조직의 IAM Access Analyzer를 관리할 수 있습니다. 위임된 관리자에게는 조직 내에서 분석기를 만들고 관리할 수 있는 권한이 있습니다. 관리 계정만 위임된 관리자를 추가할 수 있습니다.

IAM Access Analyzer의 위임된 관리자

IAM Access Analyzer에 대한 위임된 관리자는 조직 내의 구성원 계정이며 전체 조직을 대상으로 분석기를 생성하고 관리할 수 있는 권한이 있는 계정입니다. 관리 계정만 위임된 관리자를 추가, 제거 또는 변경할 수 있습니다.

위임된 관리자를 추가하는 경우 이후에 위임된 관리자의 다른 계정으로 변경할 수 있습니다. 이렇게 하면 이전에 위임된 관리자 계정은 전체 조직을 대상으로 분석하기 위해 해당 계정을 사용하여 생성했던 모든 분석기에 대한 권한을 잃게 됩니다. 이러한 분석기는 비활성화된 상태로 이동하고 더 이상 새 결과를 생성하거나 기존 결과를 업데이트하지 않습니다. 이러한 분석기의 기존 결과에도 더 이상 액세스할 수 없습니다. 계정을 위임된 관리자로 구성하여 향후에 다시 이들을 액세스할 수 있습니다. 위임된 관리자와 동일한 계정을 사용하지 않을 경우 위임된 관리자를 변경하기 전에 분석기를 삭제하는 것이 좋습니다. 이렇게 하면 생성된 모든 결과가 삭제됩니다. 새로 위임된 관리자가 새 분석기를 생성하면 동일한 결과에 대해 새 인스턴스가 생성됩니다. 결과가 손실되지 않으며, 다른 계정의 새 분석기를 위해 인스턴스가 생성됩니다. 또한 관리자 권한도 가지고 있는 조직 관리 계정을 사용하여 조직에서 결과에 계속 액세스할 수 있습니다. 새로 위임된 관리자는 이 조직에서 리소스 모니터링을 시작할 수 있도록 IAM Access Analyzer에서 새 분석기를 생성해야 합니다.

위임된 관리자가 AWS 조직을 퇴사하면 위임된 관리 권한이 계정에서 제거됩니다. 조직을 신뢰 영역으로 하는 계정의 모든 분석기는 비활성화된 상태로 이동합니다. 이러한 분석기의 기존 결과에도 더 이상 액세스할 수 없습니다.

관리 계정에서 분석기를 처음 구성할 때 IAM Access Analyzer 콘솔의 분석기 설정 페이지에서 위임된 관리자 추가를 선택할 수 있습니다.

> **Note**
>
> IAM Access Analyzer는 매월 분석기별로 분석된 IAM 역할 및 사용자 수를 기준으로 미사용 액세스 분석기에 대한 요금을 부과합니다. 관리 계정과 위임된 관리자 계정에서 미사용 액세스 분석기를 생성하면 미사용 액세스 분석기 모두에 대한 요금이 부과됩니다. 요금에 대한 자세한 내용은 IAM Access Analyzer 요금을 참조하세요.

콘솔을 사용하여 위임된 관리자를 추가하려면

1. 조직의 관리 계정을 사용하여 AWS 콘솔에 로그인합니다.
2. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
3. 분석기 액세스에서 분석기 설정을 선택합니다.
4. Add delegated administrator(위임된 관리자 추가)를 선택합니다.
5. 위임된 관리자 필드에서 위임된 관리자를 만들려면 조직 구성원 계정의 AWS 계정 번호를 입력합니다.

   계정은 조직의 구성원이어야 합니다.
6. Save changes(변경 사항 저장)를 선택합니다.

AWS CLI 또는 AWS SDK를 사용하여 위임된 관리자를 추가하려면

AWS CLI, AWS API(AWS SDK 사용) 또는 AWS CloudFormation을 사용하여 위임된 관리자 계정에 조직 전체를 대상으로 액세스할 수 있는 분석기를 생성하는 경우, AWS Organizations API를 사용하여 IAM Access Analyzer에 대한 서비스 액세스를 활성화하고 구성원 계정을 위임된 관리자로 등록해야 합니다.

1. AWS Organizations에서 IAM Access Analyzer에 대해 신뢰할 수 있는 서비스 액세스를 활성화합니다. AWS Organizations 사용 설명서의 신뢰할 수 있는 액세스를 활성화 또는 비활성화하는 방법을 참조하십시오.
2. AWS Organizations `RegisterDelegatedAdministrator` API 작업 또는 `register-delegated-administrator` AWS CLI 명령을 사용하여 AWS 조직의 유효한 멤버 계정을 위임된 관리자로 등록합니다.

위임된 관리자를 변경하고 나면 새 관리자가 조직의 리소스에 대한 액세스 권한 모니터링을 시작하기 위해 분석기를 생성해야 합니다.

분석기 삭제

분석기 설정 페이지에서 기존 외부 및 미사용 액세스 분석기를 삭제할 수 있습니다. 분석기를 삭제하면 분석기에 지정된 리소스가 더 이상 모니터링되지 않으며 새 결과도 생성되지 않습니다. 분석기에서 생성된 모든 결과가 삭제됩니다.

결과를 생성한 분석기가 삭제되었기 때문에 삭제된 결과의 경우, 분석기가 삭제된 후 2일 내에 EventBridge로 이벤트가 전송됩니다. 분석기를 삭제한 후 Security Hub 결과를 삭제하려면 최대 90일이 걸릴 수 있습니다.

분석기를 삭제하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 분석기 액세스에서 분석기 설정을 선택합니다.
3. 삭제할 분석기를 선택하고 삭제를 선택합니다.
4. 확인 텍스트 상자에 **`delete`**를 입력한 다음 삭제를 선택합니다.

아카이브 규칙

아카이브 규칙은 규칙을 생성할 때 정의한 기준을 충족하는 새 결과를 자동으로 아카이브합니다. 아카이브 규칙 기준을 충족하는 기존 결과를 아카이브하기 위해 아카이브 규칙을 소급해서 적용할 수도 있습니다. 예를 들어, 정기적으로 액세스 권한을 부여한 특정 Amazon S3 버킷에 대한 결과를 자동으로 아카이브하는 아카이브 규칙을 생성할 수 있습니다. 또는 특정 보안 주체에게 여러 리소스에 대한 액세스 권한을 부여하는 경우 해당 보안 주체에게 부여된 액세스 권한에 대해 생성된 새로운 결과를 자동으로 아카이브하는 규칙을 생성할 수 있습니다. 이렇게 하면 보안 위험성이 있는 활성 상태의 결과에만 집중할 수 있습니다.

아카이브 규칙을 생성하면 규칙 기준과 일치하는 새로운 결과만 자동으로 보관됩니다. 기존 결과는 자동으로 아카이브되지 않습니다. 규칙을 만들 때 규칙의 기준당 최대 20개의 값을 포함할 수 있습니다. 아카이브 규칙을 생성하거나 업데이트하는 데 사용할 수 있는 필터 키 목록은 IAM Access Analyzer 필터 키 단원을 참조하세요.

> **Note**
>
> 아카이브 규칙을 생성하거나 편집할 때 IAM Access Analyzer는 규칙에 대한 필터에 포함되는 값의 유효성을 검사하지 않습니다. 예를 들어, AWS 계정과 일치하는 규칙을 추가하는 경우 IAM Access Analyzer는 유효한 AWS 계정 번호가 아니더라도 필드의 모든 값을 허용합니다.

아카이브 규칙을 생성하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 분석기 액세스를 선택한 다음 분석기 설정을 선택합니다.
3. 분석기 섹션에서 보관 규칙을 만들고자 하는 분석기를 선택합니다.
4. 아카이빙 규칙 탭에서 아카이빙 규칙 생성을 선택합니다.
5. 기본 이름을 변경하려면 규칙 이름을 입력합니다.
6. 규칙 섹션의 Criteria(조건)에서 규칙에 대해 일치시킬 속성을 선택합니다.
7. 포함, 해당, 같지 않음 등의 속성 값의 조건을 선택합니다.

   사용 가능한 연산자는 선택한 속성에 따라 다릅니다.
8. 선택에 따라 속성에 대해 값을 추가하거나 규칙에 대한 기준을 추가합니다. 외부 액세스 결과의 경우, 규칙이 퍼블릭 액세스에 대한 새 결과를 아카이브하지 않도록 퍼블릭 액세스 기준을 포함하고 false로 설정할 수도 있습니다.

   기준에 다른 값을 추가하려면 Add another value(다른 값 추가)를 선택합니다. 규칙에 대해 다른 기준을 추가하려면 기준 추가를 선택합니다.
9. 기준 및 값 추가가 완료되면 규칙 생성을 선택하여 새 결과에만 규칙을 적용합니다. 규칙 기준에 따라 새 결과와 기존 결과를 아카이브하려면 활성 결과 생성 및 아카이브를 선택합니다. 결과 섹션에서 아카이브 규칙을 적용할 활성 결과 목록을 검토할 수 있습니다.

예를 들어 Amazon S3 버킷에 대한 결과를 자동으로 아카이브하는 규칙을 생성하려면 리소스 유형을 선택한 다음 조건으로 해당을 선택합니다. 그런 다음 값 목록에서 S3 버킷을 선택합니다.

특정 계정에 대한 결과를 자동으로 보관하는 미사용 액세스 결과에 대한 규칙을 만드려면 리소스 소유자 계정을 선택한 후 조건에 대해 같음을 선택합니다. 값 텍스트 상자에 AWS 계정 ID를 입력합니다.

조건을 계속 정의하여 사용자 환경에 적합한 규칙을 사용자 지정한 다음 규칙 생성을 선택합니다.

새 규칙을 생성하고 여러 조건을 추가하는 경우 Remove this criterion(이 조건 제거)를 선택하여 규칙에서 단일 기준을 제거할 수 있습니다. Remove value(값 제거)를 선택하여 기준에 대해 추가된 값을 제거할 수 있습니다.

아카이브 규칙을 편집하려면

1. 이름 열에서 편집할 규칙의 이름을 선택합니다.

   한 번에 하나의 아카이브 규칙만 편집할 수 있습니다.

2. 새 기준을 추가하거나 각 기준에 대한 기존 기준 및 값을 제거합니다.
3. 새 결과에만 규칙을 적용하려면 변경 사항 저장을 선택합니다. 규칙 기준에 따라 새 결과와 기존 결과를 아카이브하려면 활성 결과 저장 및 아카이브를 선택합니다.

아카이브 규칙을 삭제하려면

1. 삭제하려는 규칙 확인란을 선택합니다.
2. 삭제를 선택합니다.
3. Delete archive rule(아카이브 규칙 삭제) 확인 대화 상자에 **`delete`**을 입력한 다음, 삭제를 선택합니다.

규칙은 현재 리전의 분석기에서만 삭제됩니다. 다른 리전에서 생성한 각 분석기에 대해 아카이브 규칙을 별도로 삭제해야 합니다.

Amazon EventBridge로 AWS Identity and Access Management Access Analyzer 모니터링

이 주제의 정보를 이용해 Amazon EventBridge로 IAM Access Analyzer 결과와 액세스 미리 보기를 모니터링하는 방법을 알아봅니다. EventBridge는 Amazon CloudWatch Events의 새 버전입니다.

결과 이벤트

IAM Access Analyzer는 기존 결과의 상태 변경 시, 그리고 검색 결과가 삭제될 때 생성된 각 결과에 대해 Eventbridge에 이벤트를 전송합니다. 결과와 결과에 대한 알림을 받으려면 Amazon EventBridge에서 이벤트 규칙을 생성해야 합니다. 이벤트 규칙을 생성할 때 규칙에 따라 트리거할 대상 동작을 지정할 수도 있습니다. 예를 들어 IAM Access Analyzer에서 새 결과에 대한 이벤트를 수신할 때 Amazon SNS 주제를 트리거하는 이벤트 규칙을 생성할 수 있습니다.

액세스 미리 보기 이벤트

IAM Access Analyzer는 각 액세스 미리 보기 및 해당 상태 변경에 대해 EventBridge에 이벤트를 전송합니다. 여기에는 액세스 미리 보기가 처음 생성될 때(Creating 상태), 액세스 미리 보기가 완료될 때(Completed 상태) 또는 액세스 미리 보기 생성이 실패한 때(Failed 상태)의 이벤트가 포함됩니다. 액세스 미리 보기에 대한 알림을 받으려면 Amazon EventBridge에서 이벤트 규칙을 생성해야 합니다. 이벤트 규칙을 생성할 때 규칙에 따라 트리거할 대상 동작을 지정할 수 있습니다. 예를 들어 IAM Access Analyzer에서 완료된 액세스 미리 보기에 대한 이벤트를 수신할 때 Amazon SNS 주제를 트리거하는 이벤트 규칙을 생성할 수 있습니다.

이벤트 알림 빈도

IAM Access Analyzer는 계정에서 이벤트가 발생한 시점부터 약 1시간 내에 새 결과와 상태가 업데이트된 결과에 대한 이벤트를 EventBridge에 전송합니다. IAM Access Analyzer는 또한 보존 기간이 만료되어 해결된 결과가 삭제되는 경우에도 EventBridge에 이벤트를 전송합니다. 결과를 생성한 분석기가 삭제되었기 때문에 삭제가 된 결과의 경우, 분석기가 삭제된 후 약 24시간 내에 EventBridge로 이벤트가 전송됩니다. 결과가 삭제될 때 검색 상태는 변경되지 않습니다. 대신 `isDeleted` 속성이 `true`로 설정됩니다. 또한 IAM Access Analyzer는 새로 생성된 액세스 미리 보기 및 액세스 미리 보기 상태 변경에 대한 이벤트를 EventBridge로 보냅니다.

외부 액세스 조사 결과 이벤트 사례

다음은 EventBridge로 전송되는 IAM Access Analyzer 외부 액세스 분석 이벤트의 예시입니다. 나열된 `id`는 Eventbridge의 이벤트 ID입니다. 자세한 내용은 EventBridge의 이벤트 및 이벤트 패턴을 참조하세요.

`detail` 객체에서 `accountId` 및 `region` 속성의 값은 결과에 보고된 계정 및 리전을 나타냅니다. `isDeleted` 속성은 이벤트가 삭제 중인 결과에서 발생했는지 여부를 나타냅니다. `id`은 결과 ID입니다. `resources` 배열은 결과를 생성한 분석기의 ARN이 있는 singleton입니다.

```
{
    "account": "111122223333",
    "detail": {
        "accountId": "111122223333",
        "action": [
            "s3:GetObject"
        ],
        "analyzedAt": "2019-11-21T01:22:22Z",
        "condition": {},
        "createdAt": "2019-11-20T04:58:50Z",
        "id": "22222222-dcba-4444-dcba-333333333333",
        "isDeleted": false,
        "isPublic": false,
        "principal": {
            "AWS": "999988887777"
        },
        "region": "us-west-2",
        "resource": "arn:aws:s3:::my-bucket",
        "resourceType": "AWS::S3::Bucket",
        "status": "ACTIVE",
```

```
        "updatedAt": "2019-11-21T01:14:07Z",
        "version": "1.0"
    },
    "detail-type": "Access Analyzer Finding",
    "id": "11111111-2222-4444-aaaa-333333333333",
    "region": "us-west-2",
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2019-11-21T01:22:33Z",
    "version": "0"
}
```

IAM Access Analyzer는 오류 결과에 대한 이벤트도 EventBridge에 전송합니다. 오류 결과는 IAM Access Analyzer에서 리소스를 분석할 수 없을 때 생성되는 결과입니다. 오류 결과를 위한 이벤트에는 다음 예제에서와 같이 `error` 속성이 포함됩니다.

```
{
    "account": "111122223333",
    "detail": {
        "accountId": "111122223333",
        "analyzedAt": "2019-11-21T01:22:22Z",
        "createdAt": "2019-11-20T04:58:50Z",
        "error": "ACCESS_DENIED",
        "id": "22222222-dcba-4444-dcba-333333333333",
        "isDeleted": false,
        "region": "us-west-2",
        "resource": "arn:aws:s3:::my-bucket",
        "resourceType": "AWS::S3::Bucket",
        "status": "ACTIVE",
        "updatedAt": "2019-11-21T01:14:07Z",
        "version": "1.0"
    },
    "detail-type": "Access Analyzer Finding",
    "id": "11111111-2222-4444-aaaa-333333333333",
    "region": "us-west-2",
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2019-11-21T01:22:33Z",
    "version": "0"
```

```
}
```

미사용 액세스 조사 결과 관련 이벤트의 예

다음은 EventBridge로 전송되는 IAM Access Analyzer 미사용 액세스 분석 이벤트의 예시입니다. 나열된 `id`는 Eventbridge의 이벤트 ID입니다. 자세한 내용은 EventBridge의 이벤트 및 이벤트 패턴을 참조하세요.

`detail` 객체에서 `accountId` 및 `region` 속성의 값은 결과에 보고된 계정 및 리전을 나타냅니다. `isDeleted` 속성은 이벤트가 삭제 중인 결과에서 발생했는지 여부를 나타냅니다. `id`은 결과 ID입니다.

```
{
    "version": "0",
    "id": "dc7ce3ee-114b-3243-e249-7f10f9054b21",
    "detail-type": "Unused Access Finding for IAM entities",
    "source": "aws.access-analyzer",
    "account": "123456789012",
    "time": "2023-09-29T17:31:40Z",
    "region": "us-west-2",
    "resources": [
       "arn:aws:access-analyzer:us-west-2:123456789012:analyzer/
integTestLongLivingAnalyzer-DO-NOT-DELETE"
       ],
    "detail": {
        "findingId": "b8ae0460-5d29-4922-b92a-ba956c986277",
        "resource": "arn:aws:iam::111122223333:role/FindingIntegTestFakeRole",
        "resourceType": "AWS::IAM::Role",
        "accountId": "111122223333",
        "createdAt": "2023-09-29T17:29:18.758Z",
        "updatedAt": "2023-09-29T17:29:18.758Z",
        "analyzedAt": "2023-09-29T17:29:18.758Z",
        "previousStatus": "",
        "status": "ACTIVE",
        "version": "62160bda-8e94-46d6-ac97-9670930d8ffb",
        "isDeleted": false,
        "findingType": "UnusedPermission",
        "numberOfUnusedServices": 0,
        "numberOfUnusedActions": 1
        }
    }
```

IAM Access Analyzer는 오류 결과에 대한 이벤트도 EventBridge에 전송합니다. 오류 결과는 IAM Access Analyzer에서 리소스를 분석할 수 없을 때 생성되는 결과입니다. 오류 결과를 위한 이벤트에는 다음 예제에서와 같이 `error` 속성이 포함됩니다.

```
{
    "version": "0",
    "id": "c2e7aa1a-4df7-7652-f33e-64113b8997d4",
    "detail-type": "Unused Access Finding for IAM entities",
    "source": "aws.access-analyzer",
    "account": "111122223333",
    "time": "2023-10-31T20:26:12Z",
    "region": "us-west-2",
    "resources": [
      "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/ba811f91-
de99-41a4-97c0-7481898b53f2"
      ],
    "detail": {
        "findingId": "b01a34f2-e118-46c9-aef8-0d8526b495c7",
        "resource": "arn:aws:iam::123456789012:role/TestRole",
        "resourceType": "AWS::IAM::Role",
        "accountId": "444455556666",
        "createdAt": "2023-10-31T20:26:08.647Z",
        "updatedAt": "2023-10-31T20:26:09.245Z",
        "analyzedAt": "2023-10-31T20:26:08.525Z",
        "previousStatus": "",
        "status": "ACTIVE",
        "version": "7c7a72a2-7963-4c59-ac71-f0be597010f7",
        "isDeleted": false,
        "findingType": "UnusedIAMRole",
        "error": "INTERNAL_ERROR"
        }
  }
```

액세스 미리 보기 이벤트 예

다음 예제에서는 액세스 미리 보기를 생성할 때 EventBridge로 전송되는 첫 번째 이벤트의 데이터를 보여줍니다. `resources` 배열은 액세스 미리 보기가 연결된 분석기의 ARN이 있는 singleton입니다. `detail` 객체에서 `id`는 액세스 미리 보기 ID를 나타내고 `configuredResources`는 액세스 미리 보기가 생성된 리소스를 나타냅니다. `status`는 `Creating`이며 액세스 미리 보기 상태를 나타냅니다. 액세스 미리 보기가 방금 생성되었으므로 `previousStatus`는 지정되어 있지 않습니다.

```
{
```

```
    "account": "111122223333",
    "detail": {
        "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
        "configuredResources": [
            "arn:aws:s3:::example-bucket"
        ],
        "createdAt": "2020-02-20T00:00:00.00Z",
        "region": "us-west-2",
        "status": "CREATING",
        "version": "1.0"
    },
    "detail-type": "Access Preview State Change",
    "id": "aaaabbbb-2222-3333-4444-555566667777",
    "region": "us-west-2",
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2020-02-20T00:00:00.00Z",
    "version": "0"
}
```

다음 예제에서는 상태가 `Creating`에서 `Completed`로 변경된 액세스 미리 보기에 대해 EventBridge로 전송된 이벤트 데이터를 보여줍니다. 세부 정보 객체에서 `id`는 액세스 미리 보기 ID를 나타냅니다. `status` 및 `previousStatus`는 액세스 미리 보기 상태를 나타내며, 여기서는 이전 상태가 `Creating`, 현재 상태가 `Completed`입니다.

```
{
    "account": "111122223333",
    "detail": {
        "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
        "configuredResources": [
            "arn:aws:s3:::example-bucket"
        ],
        "createdAt": "2020-02-20T00:00:00.000Z",
        "previousStatus": "CREATING",
        "region": "us-west-2",
        "status": "COMPLETED",
        "version": "1.0"
    },
    "detail-type": "Access Preview State Change",
    "id": "11112222-3333-4444-5555-666677778888",
    "region": "us-west-2",
```

```
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2020-02-20T00:00:00.00Z",
    "version": "0"
}
```

다음 예제에서는 상태가 `Creating`에서 `Failed`로 변경된 액세스 미리 보기에 대해 EventBridge로 전송된 이벤트 데이터를 보여줍니다. `detail` 객체에서 `id`는 액세스 미리 보기 ID를 나타냅니다. `status` 및 `previousStatus`는 액세스 미리 보기 상태를 나타내며, 여기서는 이전 상태가 `Creating`, 현재 상태가 `Failed`입니다. `statusReason` 필드에는 액세스 미리 보기가 잘못된 리소스 구성으로 인해 실패했음을 나타내는 사유 코드가 제공됩니다.

```
{
    "account": "111122223333",
    "detail": {
        "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
        "configuredResources": [
            "arn:aws:s3:::example-bucket"
        ],
        "createdAt": "2020-02-20T00:00:00.00Z",
        "previousStatus": "CREATING",
        "region": "us-west-2",
        "status": "FAILED",
        "statusReason": {
            "code": "INVALID_CONFIGURATION"
        },
        "version": "1.0"
    },
    "detail-type": "Access Preview State Change",
    "id": "99998888-7777-6666-5555-444433332222",
    "region": "us-west-2",
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2020-02-20T00:00:00.00Z",
    "version": "0"
}
```

콘솔을 사용하여 이벤트 규칙 생성

다음 절차에서는 콘솔을 사용하여 이벤트를 생성하는 방법을 설명합니다.

1. Amazon EventBridge 콘솔(https://console.aws.amazon.com/events/)을 엽니다.
2. 다음 값을 사용하여 찾기 이벤트 또는 액세스 미리 보기 이벤트를 모니터링하는 EventBridge 규칙을 생성합니다.

   - 규칙 유형에서 이벤트 패턴이 있는 규칙을 선택합니다.
   - 이벤트 소스(Event source)에서 기타(Other)를 선택합니다.
   - 이벤트 패턴(Event pattern)에서 사용자 지정 패턴(JSON 편집기)(Custom patterns (JSON editor))을 선택하고 다음 이벤트 패턴 예제 중 하나를 텍스트 영역에 붙여 넣습니다.
     - 외부 액세스 또는 미사용 액세스 조사 결과 이벤트를 기반으로 규칙을 생성하려면 다음 패턴 예제를 사용합니다.

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Access Analyzer Finding"
  ]
}
```

     - 미사용 액세스 조사 결과 이벤트만을 기반으로 규칙을 생성하려면 다음 패턴 예제를 사용합니다.

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Unused Access Finding for IAM entities"
  ]
}
```

> **Note**
>
> 외부 액세스 조사 결과 이벤트만을 기반으로 규칙을 생성할 수는 없습니다.

- 액세스 미리 보기 이벤트를 기반으로 규칙을 생성하려면 다음 패턴 예제를 사용합니다.

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Access Preview State Change"
  ]
}
```

- 대상 유형(Target types)에서 AWS 서비스를 선택하고 대상 선택(Select a target)에서 Amazon SNS 주제 또는 AWS Lambda 함수와 같은 대상을 선택합니다. 규칙에 정의된 이벤트 패턴과 일치하는 이벤트를 수신할 때 대상이 트리거됩니다.

규칙 생성에 대한 자세한 내용을 알아보려면 Amazon EventBridge User Guide(Amazon EventBridge 사용 설명서)의 Creating Amazon EventBridge rules that react to events(이벤트에 대응하는 Amazon EventBridge 규칙 생성)를 참조하세요.

CLI를 사용하여 이벤트 규칙 생성

1. 다음에 따라 AWS CLI을 사용하여 Amazon EventBridge에 대한 규칙을 생성합니다. 규칙 이름 *TestRule*을 자신의 규칙 이름으로 바꿉니다.

```
aws events put-rule --name TestRule --event-pattern "{\"source\":[\"aws.access-
analyzer\"]}"
```

2. 특정 속성을 가진 결과와 같이 생성된 결과의 하위 집합에 대해서만 대상 작업을 트리거하도록 규칙을 사용자 정의할 수 있습니다. 다음 예제에서는 활성 상태인 결과에 대해서만 대상 작업을 트리거하는 규칙을 생성하는 방법을 보여 줍니다.

```
aws events put-rule --name TestRule --event-pattern "{\"source\":[\"aws.access-
analyzer\"],\"detail-type\":[\"Access Analyzer Finding\"],\"detail\":{\"status\":
[\"ACTIVE\"]}}"
```

다음 예제에서는 상태가 `Creating`에서 `Completed`로 변경된 액세스 미리 보기에 대해서만 대상 작업을 트리거하는 규칙을 생성하는 방법을 보여줍니다.

```
aws events put-rule --name TestRule --event-pattern "{\"source\":[\"aws.access-
analyzer\"],\"detail-type\":[\"Access Preview State Change\"],\"detail\":{\"status
\":[\"COMPLETED\"]}}"
```

3. Lambda 함수를 생성한 규칙의 대상으로 정의하려면 다음 예제 명령을 사용합니다. ARN의 리전 및 함수 이름을 사용자 환경에 맞게 바꿉니다.

```
aws events put-targets --rule TestRule --targets Id=1,Arn=arn:aws:lambda:us-
east-1:111122223333:function:MyFunction
```

4. 규칙 대상을 호출하는 데 필요한 권한을 추가합니다. 다음 예제에서는 앞의 예에 따라 Lambda 함수에 권한을 부여하는 방법을 보여줍니다.

```
aws lambda add-permission --function-name MyFunction --statement-id 1 --action
 'lambda:InvokeFunction' --principal events.amazonaws.com
```

AWS Security Hub와 IAM Access Analyzer의 통합

AWS Security Hub에서는 AWS 전반의 보안 상태에 대한 포괄적인 보기가 제공됩니다. 이를 통해 고객 환경에서 보안 업계 표준 및 모범 사례를 준수하는지 확인할 수도 있습니다. Security Hub는 AWS 계정, 서비스 및 지원되는 서드 파티 파트너 제품에서 보안 데이터를 수집합니다. 또한 보안 추세를 분석하고 우선 순위가 가장 높은 보안 문제를 식별합니다.

IAM Access Analyzer를 Security Hub와 통합하면 IAM Access Analyzer의 조사 결과를 Security Hub로 보낼 수 있습니다. 그러면 Security Hub의 전반적인 보안 태세 분석에 이러한 조사 결과가 포함됩니다.

목차

- 결과 전송을 중지하는 방법

IAM Access Analyzer가 조사 결과를 Security Hub로 보내는 방법

Security Hub의 경우 보안 문제를 조사 결과와 같이 추적합니다. 일부 결과는 다른 AWS 서비스 또는 서드 파티에서 감지한 문제에서 비롯됩니다. Security Hub에는 보안 문제를 감지하고 조사 결과를 생성하는 데 사용하는 규칙 집합도 있습니다.

Security Hub는 이러한 모든 출처를 총망라하여 조사 결과를 관리할 도구를 제공합니다. 사용자는 조사 결과 목록을 조회하고 필터링할 수 있으며 각 조사 결과의 세부 정보를 조회할 수도 있습니다. 자세한 내용은 AWS Security Hub User Guide의 Viewing findings를 참조하세요. 또한 조사 결과에 대한 조사 상태를 추적할 수도 있습니다. 자세한 내용은 AWS Security Hub User Guide의 Taking action on findings를 참조하세요.

Security Hub의 모든 결과는 표준 JSON 형식을 사용합니다. 이를 AWS Security Finding Format(ASFF)이라고 합니다. ASFF에는 문제의 출처, 영향을 받은 리소스와 결과의 현재 상태 등에 관한 세부 정보가 포함됩니다. 자세한 내용은 AWS Security Hub User Guide의 AWS Security Finding Format (ASFF)을 참조하세요.

AWS Identity and Access Management Access Analyzer는 Security Hub에 조사 결과를 전송하는 AWS 서비스 중 하나입니다. 미사용 액세스의 경우 IAM Access Analyser는 IAM 사용자 또는 역할에 부여된 미사용 액세스를 감지하고 각각에 대한 조사 결과를 생성합니다. 그런 다음 조사 결과를 Security Hub로 전송합니다.

외부 액세스의 경우 IAM Access Analyser는 조직 또는 계정에서 지원되는 리소스에 대한 외부 보안 주체의 퍼블릭 액세스 또는 크로스 계정 액세스를 허용하는 정책문을 감지합니다. IAM Access Analyser는 퍼블릭 액세스에 대한 조사 결과를 생성하여 Security Hub로 보냅니다. 크로스 계정 액세스의 경우 IAM Access Analyzer는 한 번에 하나의 외부 보안 주체에 대한 단일 조사 결과를 Security Hub로 보냅니다. IAM Access Analyzer에 여러 크로스 계정 조사 결과가 있는 경우 IAM Access Analyzer가 다음 크로스 계정 조사 결과를 제공하기 전에 단일 외부 보안 주체에 대한 Security Hub 조사 결과를 해결해야 합니다. 분석기에 대한 신뢰 영역 외부에서 크로스 계정 액세스 권한이 있는 외부 보안 주체의 전체 목록을 보려면 IAM Access Analyser에서 조사 결과를 확인해야 합니다.

IAM Access Analyzer가 보내는 조사 결과의 유형

IAM Access Analyzer는 AWS Security 분석 결과 형식(ASFF)을 사용하여 해당 조사 결과를 Security Hub로 보냅니다. ASFF의 경우, `Types` 필드가 결과 유형을 제공합니다. IAM Access Analyzer의 조사 결과는 `Types`의 값이 다음과 같을 수 있습니다.

- 외부 액세스 조사 결과 - 효과, 데이터 노출, 외부 액세스 허용
- 외부 액세스 조사 결과 – 소프트웨어 및 구성 검사/AWS 보안 모범 사례/외부 액세스 허용
- 미사용 액세스 조사 결과 - 소프트웨어 및 구성 검사/AWS 보안 모범 사례/미사용 권한
- 미사용 액세스 조사 결과 - 소프트웨어 및 구성 검사/AWS 보안 모범 사례/미사용 IAM 역할
- 미사용 액세스 조사 결과 - 소프트웨어 및 구성 검사/AWS 보안 모범 사례/미사용 IAM 사용자 암호
- 미사용 액세스 조사 결과 - 소프트웨어 및 구성 검사/AWS 보안 모범 사례/미사용 IAM 사용자 액세스 키

조사 결과 전송 지연 시간

IAM Access Analyzer가 새 조사 결과를 생성하면 일반적으로 30분 안에 Security Hub로 전송됩니다. 하지만 IAM Access Analyzer에 정책 변경에 대한 알림이 전송되지 않는 경우가 드물게 있습니다. 예:

- Amazon S3 계정 수준 블록 퍼블릭 액세스 설정 변경 사항이 적용되는 데 최대 12시간이 걸릴 수 있습니다.
- AWS CloudTrail 로그 전달에 전송 문제가 있는 경우 정책을 변경해도 조사 결과에 보고된 리소스의 다시 검색을 트리거하지 않을 수 이습니다.

이런 경우 IAM Access Analyzer는 다음 정기 검색 중에 새 정책 또는 업데이트된 정책을 분석합니다.

Security Hub를 사용할 수 없을 때 다시 시도

Security Hub를 사용할 수 없는 경우, IAM Access Analyzer가 정기적으로 조사 결과 전송을 다시 시도합니다.

Security Hub에서 기존 조사 결과 업데이트

IAM Access Analyzer는 조사 결과를 Security Hub로 보낸 다음 업데이트를 계속 전송하여 Security Hub에 대한 결과 활동의 추가적인 관찰 결과를 반영합니다. 이러한 업데이트는 같은 조사 결과 내에서 반영됩니다.

외부 액세스 조사 결과의 경우 IAM Access Analyzer는 이를 리소스별로 그룹화합니다. IAM Access Analyzer에서 리소스에 대한 조사 결과가 적어도 하나 이상 활성 상태인 경우, Security Hub에서도 해당 리소스에 대한 조사 결과가 활성 상태를 유지합니다. 주어진 리소스에 대한 IAM Access Analyzer의 모든 조사 결과가 보관되거나 확인된 경우, Security Hub 조사 결과도 보관됩니다. Security Hub 조사 결과는 퍼블릭 액세스 및 크로스 계정 액세스 간 정책이 변경되면 업데이트됩니다. 이 업데이트에 결과의 유형, 제목, 설명 및 심각도 변경 사항을 포함할 수 있습니다.

미사용 액세스 조사 결과의 경우 IAM Access Analyzer는 이를 리소스별로 그룹화하지 않습니다. 대신, IAM Access Analyzer에서 미사용 액세스 조사 결과가 해결되면 해당 Security Hub 조사 결과도 해결됩니다. 미사용 액세스 조사 결과를 생성한 IAM 사용자 또는 역할을 업데이트하면 Security Hub 조사 결과가 업데이트됩니다.

Security Hub에서 IAM Access Analyzer 조사 결과 보기

Security Hub의 IAM Access Analyzer 조사 결과를 조회하려면 요약 페이지의 AWS: IAM Access Analyzer 섹션에서 조사 결과 보기를 선택합니다. 아니면 탐색 창에서 [Findings]를 선택해도 됩니다. 그런 다음 결과를 필터링하여 값이 **`IAM Access Analyzer`**인 Product name: 필드를 선택하면 AWS Identity and Access Management Access Analyzer 결과만 표시하도록 할 수 있습니다.

Security Hub에서 IAM Access Analyzer 조사 결과 이름 해석

AWS Identity and Access Management Access Analyzer는 AWS Security Finding Format(ASFF)을 사용하여 조사 결과를 Security Hub로 보냅니다. ASFF의 경우, Types 필드가 결과 유형을 제공합니다. ASFF 유형은 AWS Identity and Access Management Access Analyzer과(와)는 다른 명명 체계를 사용합니다. 다음 테이블에 Security Hub에 표시되는 대로 AWS Identity and Access Management Access Analyzer 조사 결과와 연관된 모든 ASFF 유형에 관한 세부 정보를 포함하였습니다.

ASFF 결과 유형	Security Hub	설명
효과/데이터 노출/외부 액세스 허용	<resource ARN>이 퍼블릭 액세스 허용	리소스에 연결된 리소스 기반 정책이 모든 외부 보안 주체에 해당 리소스에 대한 퍼블릭 액세스를 허용합니다.
소프트웨어 및 구성 점검/AWS 보안 모범 사례/외부 액세스 허용	<resource ARN>이 교차 계정 액세스 허용	리소스에 연결된 리소스 기반 정책이 해당 분석기의 신뢰 영역에서 벗어나는 외부 보안 주체에 교차 계정 액세스를 허용합니다.
소프트웨어 및 구성 검사/AWS 보안 모범 사례/미사용 권한	<resource ARN>미사용 권한 포함	사용자 또는 역할에 미사용 서비스 및 작업 권한이 포함되어 있습니다.

ASFF 결과 유형	Security Hub	설명
소프트웨어 및 구성 검사/AWS 보안 모범 사례/미사용 IAM 역할	<resource ARN>미사용 IAM 역할 포함	사용자 또는 역할에 미사용 IAM 역할이 포함되어 있습니다.
소프트웨어 및 구성 점검/AWS 보안 모범 사례/미사용 IAM 사용자 암호	<resource ARN>미사용 IAM 사용자 암호 포함	사용자 또는 역할에 미사용 IAM 사용자 암호가 포함되어 있습니다.
소프트웨어 및 구성 검사/AWS 보안 모범 사례/미사용 IAM 사용자 액세스 키	<resource ARN>미사용 IAM 사용자 액세스 키 포함	사용자 또는 역할에 미사용 IAM 사용자 액세스 키가 포함되어 있습니다.

IAM Access Analyzer의 일반적인 조사 결과

IAM Access Analyzer는 AWS Security 분석 결과 형식(ASFF)을 사용하여 조사 결과를 Security Hub로 보냅니다.

다음은 외부 액세스 조사 결과에 대한 IAM Access Analyzer의 일반적인 조사 결과를 예시로 나타낸 것입니다.


```
{
    "SchemaVersion": "2018-10-08",
    "Id": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/my-analyzer/
arn:aws:s3:::my-bucket",
    "ProductArn": "arn:aws:securityhub:us-west-2::product/aws/access-analyzer",
    "GeneratorId": "aws/access-analyzer",
    "AwsAccountId": "111122223333",
    "Types": ["Software and Configuration Checks/AWS Security Best Practices/External
 Access Granted"],
    "CreatedAt": "2020-11-10T16:17:47Z",
    "UpdatedAt": "2020-11-10T16:43:49Z",
    "Severity": {
        "Product": 1,
        "Label": "LOW",
        "Normalized": 1
    },
    "Title": "AwsS3Bucket/arn:aws:s3:::my-bucket/ allows cross-account access",
```

```
    "Description": "AWS::S3::Bucket/arn:aws:s3:::my-bucket/ allows cross-account access
 from AWS 444455556666",
    "Remediation": {
        "Recommendation": {"Text": "If the access isn't intended, it indicates a
 potential security risk. Use the console for the resource to modify or remove the
 policy that grants the unintended access. You can use the Rescan button on the Finding
 details page in the Access Analyzer console to confirm whether the change removed the
 access. If the access is removed, the status changes to Resolved."}
    },
    "SourceUrl": "https://console.aws.amazon.com/access-analyzer/home?region=us-
west-2#/findings/details/dad90d5d-63b4-6575-b0fa-ef9c556ge798",
    "Resources": [
        {
            "Type": "AwsS3Bucket",
            "Id": "arn:aws:s3:::my-bucket",
            "Details": {
                "Other": {
                    "External Principal Type": "AWS",
                    "Condition": "none",
                    "Action Granted": "s3:GetObject,s3:GetObjectVersion",
                    "External Principal": "444455556666"
                }
            }
        }
    ],
    "WorkflowState": "NEW",
    "Workflow": {"Status": "NEW"},
    "RecordState": "ACTIVE"
}
```

다음은 미사용 액세스 조사 결과에 대한 IAM Access Analyzer의 일반적인 조사 결과를 예시로 나타낸 것입니다.

```
{
    "Findings": [
    {
      "SchemaVersion": "2018-10-08",
      "Id": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/integTestAnalyzer-
DO-NOT-DELETE/arn:aws:iam::111122223333:role/TestRole/UnusedPermissions",
      "ProductArn": "arn:aws:securityhub:us-west-2::product/aws/access-analyzer",
      "ProductName": "IAM Access Analyzer",
      "CompanyName": "AWS",
      "Region": "us-west-2",
```

```
      "GeneratorId": "aws/access-analyzer",
      "AwsAccountId": "111122223333",
      "Types": [
        "Software and Configuration Checks/AWS Security Best Practices/Unused
 Permission"
      ],
      "CreatedAt": "2023-09-18T16:29:09.657Z",
      "UpdatedAt": "2023-09-21T20:39:16.651Z",
      "Severity": {
        "Product": 1,
        "Label": "LOW",
        "Normalized": 1
      },
      "Title": "AwsIamRole/arn:aws:iam::111122223333:role/IsengardRole-DO-NOT-DELETE/
 contains unused permissions",
      "Description": "AWS::IAM::Role/arn:aws:iam::111122223333:role/IsengardRole-DO-
NOT-DELETE/ contains unused service and action-level permissions",
      "Remediation": {
        "Recommendation": {
          "Text":"If the unused permissions aren’t required, delete the permissions to
 refine access to your account. Use the IAM console to modify or remove the policy that
 grants the unused permissions. If all the unused permissions are removed, the status
 of the finding changes to Resolved."
        }
      },
      "SourceUrl": "https://us-west-2.console.aws.amazon.com/access-analyzer/
home?region=us-west-2#/unused-access-findings?resource=arn%3Aaws%3Aiam%3A
%3A903798373645%3Arole%2FTestRole",
      "ProductFields": {
      "numberOfUnusedActions": "256",
      "numberOfUnusedServices": "15",
      "resourceOwnerAccount": "111122223333",
      "findingId": "DEMO24d8d-0d3f-4d3d-99f4-299fc8a62ee7",
      "findingType": "UnusedPermission",
      "aws/securityhub/FindingId": "arn:aws:securityhub:us-west-2::product/aws/access-
analyzer/arn:aws:access-analyzer:us-west-2:111122223333:analyzer/integTestAnalyzer-DO-
NOT-DELETE/arn:aws:iam::111122223333:role/TestRole/UnusedPermissions",
      "aws/securityhub/ProductName": "AM Access Analyzer",
      "aws/securityhub/CompanyName": "AWS"
    },
    "Resources": [
    {
      "Type": "AwsIamRole",
      "Id": "arn:aws:iam::111122223333:role/TestRole"
```

```
    }
  ],
  "WorkflowState": "NEW",
  "Workflow": {
    "Status": "NEW"
    },
  "RecordState": "ARCHIVED",
  "FindingProviderFields": {
    "Severity": {
      "Label": "LOW"
    },
    "Types": [
    "Software and Configuration Checks/AWS Security Best Practices/Unused Permission"
    ]
  }
  }
]
}
```

통합 활성화 및 구성

Security Hub와의 통합을 사용하려면 Security Hub를 활성화해야 합니다. Security Hub를 활성화하는 방법에 대한 자세한 내용은 AWS Security Hub 사용 설명서의 Security Hub 설정을 참조하세요.

IAM Access Analyzer와 Security Hub를 둘 다 활성화하면 통합이 자동으로 활성화됩니다. IAM Access Analyzer는 즉시 Security Hub로 조사 결과를 전송하기 시작합니다.

결과 전송을 중지하는 방법

Security Hub로 결과를 전송하는 작업을 중지하려면 Security Hub 콘솔 또는 API를 사용하면 됩니다.

AWS Security Hub 사용 설명서에서 통합에서 결과 흐름 활성화 및 비활성화(콘솔) 또는 통합에서 결과 흐름 비활성화(Security Hub API, AWS CLI)를 참조하세요.

AWS CloudTrail을 사용하여 IAM Access Analyzer API 호출 로깅

IAM Access Analyzer는 IAM Access Analyzer에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 IAM Access Analyzer에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 IAM Access Analyzer 콘솔에서 수행한 호출과 IAM Access Analyzer API 작업에 대한 코드 호출이 포함됩니다.

추적을 생성하면 IAM Access Analyzer 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록(Event history)에서 최신 이벤트를 볼 수 있습니다.

CloudTrail에서 수집한 정보를 사용하여 IAM Access Analyzer에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 IAM Access Analyzer 정보

CloudTrail은 계정 생성 시 AWS 계정에서 사용되도록 설정됩니다. IAM Access Analyzer에서 활동이 발생하면 해당 활동이 이벤트 기록(Event history)의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 정보는 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

IAM Access Analyzer에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 또는 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

모든 IAM Access Analyzer 작업은 CloudTrail에서 로깅되고 IAM Access Analyzer API 참조에 기록됩니다. 예를 들어 `CreateAnalyzer`, `CreateArchiveRule`, `ListFindings` 작업을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 자격 증명 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 AWS Identity and Access Management(IAM) 사용자 자격 증명으로 했는지.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.

- 다른 AWS 서비스에서 요청했는지 여부.

자세한 정보는 CloudTrail userIdentity 요소를 참조하세요.

IAM Access Analyzer 로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함합니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음 예제에는 “2021년 6월 14일”에 `Alice-tempcreds`(이)라는 이름의 수임된 역할 세션에 의해 수행된 `CreateAnalyzer` 작업을 보여주는 CloudTrail 로그 항목이 나와 있습니다. 역할 세션은 `admin-tempcreds`(이)라는 이름의 역할에 의해 실행되었습니다.

```
{
  "eventVersion": "1.05",
  "userIdentity":   {
    "type": "AssumedRole",
    "principalId": "AROAIBKEVSQ6C2EXAMPLE:Alice-tempcreds",
    "arn": "arn:aws:sts::111122223333:assumed-role/admin-tempcreds/Alice-tempcreds",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "true",
        "creationDate": "2021-06-14T22:54:20Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/admin-tempcreds",
        "accountId": "111122223333",
        "userName": "admin-tempcreds"
      },
     "webIdFederationData": {},
    }
  },
  "eventTime": "2021-06-14T22:57:36Z",
  "eventSource": "access-analyzer.amazonaws.com",
  "eventName": "CreateAnalyzer",
  "awsRegion": "us-west-2",
```

```
  "sourceIPAddress": "198.51.100.179",
  "userAgent": "aws-sdk-java/1.12.79 Linux/5.4.141-78.230 OpenJDK_64-
Bit_Server_VM/25.302-b08 java/1.8.0_302 vendor/Oracle_Corporation cfg/retry-mode/
standard",
  "requestParameters": {
    "analyzerName": "test",
    "type": "ACCOUNT",
    "clientToken": "11111111-abcd-2222-abcd-222222222222",
        "tags": {
            "tagkey1": "tagvalue1"
        }
  },
  "responseElements": {
    "arn": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/test"
  },
  "requestID": "22222222-dcba-4444-dcba-333333333333",
  "eventID": "33333333-bcde-5555-bcde-444444444444",
  "readOnly": false,
  "eventType": "AwsApiCall",,
  "managementEvent": true,
  "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

IAM Access Analyzer 필터 키

아래의 필터 키를 사용하여 아카이브 규칙(CreateArchiveRule)을 정의하거나, 아카이브 규칙(UpdateArchiveRule)을 업데이트하거나, 조사 결과(ListFindings 및 ListFindingsV2) 목록을 검색하거나, 리소스에 대한 액세스 미리 보기 조사 결과(ListAccessPreviewFindings) 목록을 검색할 수 있습니다. 아카이브 규칙 구성에 있어 IAM API 및 AWS CloudFormation 사이에는 차이가 없습니다.

Criterion	설명	Type	아카이브 규칙	결과 나열	액세스 미리 보기 결과 나열
리소스	외부 보안 주체가 액세스할 수 있는 리소스를 고유하게 식별하는 ARN입니다. 자세한 내용은	String	예	예	예

Criterion	설명	Type	아카이브 규칙	결과 나열	액세스 미리 보기 결과 나열
	Amazon 리소스 이름(ARN)을 참조하세요.				

Criterion	설명	Type	아카이브 규칙	결과 나열	액세스 미리 보기 결과 나열
resourceType `AWS::IAM::Role` | `AWS::KMS::Key` | `AWS::Lambda::Function` | `AWS::Lambda::LayerVersion` | `AWS::S3::Bucket` | `AWS::S3Express::DirectoryBucket` | `AWS::SQS::Queue` | `AWS::SecretsManager::Secret` | `AWS::EFS::FileSystem` | `AWS::EC2::Snapshot` | `AWS::ECR:`	외부 보안 주체가 액세스할 수 있는 리소스의 유형입니다.	String	 예	 예	 예

Criterion	설명	Type	아카이브 규칙	결과 나열	액세스 미리 보기 결과 나열
`:Repository | AWS::RDS::DBSnapshot | AWS::RDS::DBClusterSnapshot | AWS::SNS::Topic | AWS::DynamoDB::Stream | AWS::DynamoDB::Table`					
resourceOwnerAccount	리소스를 소유한 12자리 AWS 계정 ID입니다. 자세한 내용은 AWS 계정 식별자를 참조하세요.	String	 예	 예	 예
isPublic	결과에 퍼블릭 액세스를 허용하는 정책이 있는 리소스가 보고되는지 여부를 나타냅니다.	불	 예	 예	 예

Criterion	설명	Type	아카이브 규칙	결과 나열	액세스 미리 보기 결과 나열
findingType `UnusedIAMRole | UnusedIAMUserAccessKey | UnusedIAMUserPassword | UnusedPermission`	결과의 유형입니다. 사용되지 않은 액세스 조사 결과에 대해서는 조사 결과 유형별로만 필터링할 수 있습니다.	String	 예	 예	 예
상태 `ACTIVE | ARCHIVED | RESOLVED`	결과의 현재 상태입니다.	String	 아니요	 예	 예
오류	결과에 대해 보고된 오류를 나타냅니다.	String	 예	 예	 예

Criterion	설명	Type	아카이브 규칙	결과 나열	액세스 미리 보기 결과 나열
principal.AWS	결과의 `Principal` 필드에 있는 리소스에 대한 액세스 권한이 부여된 계정입니다. 12자리의 AWS 계정 ID나 외부 AWS 사용자 또는 역할의 ARN을 입력합니다. 자세한 내용은 AWS 계정 식별자를 참조하세요.	String	 예	 예	 예
principal.Federated	결과에 있는 리소스에 액세스할 수 있는 연동 자격 증명의 ARN입니다. 자세한 내용은 ID 공급자 및 연동을 참조하세요.	String	 예	 예	 예
condition.aws:PrincipalArn	리소스 액세스 조건으로 지정된 보안 주체(IAM 사용자, 역할 또는 그룹)의 ARN입니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.	String	 예	 예	 예
condition.aws:PrincipalOrgID	리소스 액세스 조건으로 지정된 보안 주체의 조직 식별자입니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.	String	 예	 예	 예

Criterion	설명	Type	아카이브 규칙	결과 나열	액세스 미리 보기 결과 나열
condition .aws:PrincipalOrgPaths	리소스 액세스 조건으로 지정된 조직 또는 OU(조직 단위) ID입니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.	String	 예	 예	 예
condition .aws:SourceIp	지정된 IP 주소를 사용할 때 보안 주체가 리소스에 액세스할 수 있도록 허용하는 IP 주소입니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.	IP 주소	 예	 예	 예
condition .aws:SourceVpc	지정된 VPC를 사용할 때 보안 주체가 리소스에 액세스할 수 있도록 허용하는 VPC ID입니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.	String	 예	 예	 예
condition .aws:UserId	리소스 액세스 조건으로 지정된 외부 계정 IAM 사용자의 사용자 ID입니다. 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.	String	 예	 예	 예

Criterion	설명	Type	아카이브 규칙	결과 나열	액세스 미리 보기 결과 나열
condition .cognito-identity. amazonaws .com:aud	결과에서 IAM 역할 액세스 조건으로 지정된 Amazon Cognito 자격 증명 풀 ID입니다. 자세한 내용은 IAM 및 AWS STS 조건 컨텍스트 키를 참조하세요.	String	 예	 예	 예
condition .graph.fa cebook.co m:app_id	결과에서 IAM 역할에 대한 Facebook으로 로그인 연동 액세스를 허용하기 위한 조건으로 지정된 Facebook 애플리케이션 ID(또는 사이트 ID)입니다. 자세한 내용은 IAM 및 AWS STS 조건 컨텍스트 키를 참조하세요.	String	 예	 예	 예
condition .accounts .google.c om:aud	IAM 역할에 대한 액세스 조건으로 지정된 Google 애플리케이션 ID입니다. 자세한 내용은 IAM 및 AWS STS 조건 컨텍스트 키를 참조하세요.	String	 예	 예	 예

Criterion	설명	Type	아카이브 규칙	결과 나열	액세스 미리 보기 결과 나열
condition .kms:CallerAccount	AWS KMS를 호출하는 서비스에서 사용되는 호출 엔터티(IAM 사용자, 역할 또는 계정 루트 사용자)를 소유한 AWS 계정 ID입니다. 자세한 내용은 AWS Key Management Service에 사용되는 조건 키를 참조하세요.	String	 예	 예	 예
condition .www.amazon.com:app_id	역할에 대한 Login with Amazon 연동 액세스를 허용하도록 조건이 지정된 Amazon 애플리케이션 ID(또는 사이트 ID)입니다. 자세한 내용은 다음을 참조하세요.	String	 예	 예	 예
id	결과의 ID입니다.	String	 아니요	 예	 예
changeType	액세스 미리 보기 검색 결과가 IAM Access Analyzer에서 식별된 기존 액세스와 어떻게 비교되는지에 대한 컨텍스트를 제공합니다.	String	 아니요	 아니요	 예

Criterion	설명	Type	아카이브 규칙	결과 나열	액세스 미리 보기 결과 나열
existingFindingId	IAM Access Analyzer 검색 결과의 기존 ID로 액세스 미리 보기의 기존 검색 결과에 대해서만 제공됩니다.	String	 아니요	 아니요	 예
existingFindingStatus	검색 결과의 기존 상태로 액세스 미리 보기의 기존 검색 결과에 대해서만 제공됩니다.	String	 아니요	 아니요	 예

AWS Identity and Access Management Access Analyzer에 서비스 연결 역할 사용

AWS Identity and Access Management Access Analyzer은 IAM 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 IAM Access Analyzer에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 IAM Access Analyzer에서 사전 정의되며. 사용자를 대신해 다른 AWS 서비스를 호출하기 위해 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할은 IAM Access Analyzer를 더 쉽게 설정할 수 있습니다. IAM Access Analyzer에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않은 한, IAM Access Analyzer만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며, 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 Service-Linked Role(서비스 연결 역할) 열에 Yes(예)가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

AWS Identity and Access Management Access Analyzer에 대한 서비스 연결 역할 권한

AWS Identity and Access Management Access Analyzer는 AWSServiceRoleForAccessAnalyzer라는 서비스 연결 역할을 사용합니다 - Access Analyzer가 외부 액세스에 대한 리소스 메타데이터를 분석하고 미사용 액세스를 식별할 수 있도록 활동을 분석하는 것을 허용합니다.

AWSServiceRoleForAccessAnalyzer 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `access-analyzer.amazonaws.com`

AccessAnalyzerServiceRolePolicy라는 역할 권한 정책을 사용하면 IAM Access Analyzer가 특정 리소스에 대한 작업을 완료할 수 있습니다.

IAM 엔터티(예: 사용자, 그룹, 역할)가 서비스 연결 역할을 생성, 편집 또는 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한 단원을 참조하세요.

IAM Access Analyzer의 서비스 연결 역할 생성

서비스 연결 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console 또는 AWS API에서 Access Analyzer를 활성화하면 IAM Access Analyzer에서 서비스 연결 역할을 생성합니다. IAM Access Analyzer를 활성화한 모든 리전에서 동일한 서비스 연결 역할이 사용됩니다. 외부 액세스와 미사용 액세스 결과 모두 동일한 서비스 연결 역할을 사용합니다.

> **Note**
>
> IAM Access Analyzer는 리전별로 적용됩니다. 각 리전에서 독립적으로 IAM Access Analyzer를 활성화해야 합니다.

이 서비스 연결 역할을 삭제하면 다음에 분석기를 생성할 때 IAM Access Analyzer에서 역할이 다시 생성됩니다.

IAM 콘솔을 사용해 Access Analyzer 사용 사례로 서비스 연결 역할을 생성할 수도 있습니다. AWS CLI 또는 AWS API에서 `access-analyzer.amazonaws.com` 서비스 이름의 서비스 연결 역할을 생성합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 생성 단원을 참조하세요. 이 서비스 연결 역할을 삭제한 후에는 동일한 프로세스를 사용하여 역할을 다시 생성할 수 있습니다.

IAM Access Analyzer의 서비스 연결 역할 편집

IAM Access Analyzer에서는 AWSServiceRoleForAccessAnalyzer 서비스 연결 역할을 편집할 수 없습니다. 서비스 연결 역할을 생성한 후에는 다양한 엔터티가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집 섹션을 참조하세요.

IAM Access Analyzer의 서비스 연결 역할 삭제

서비스 연결 역할을 요구하는 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 이렇게 하면 적극적으로 모니터링하거나 유지 관리하지 않는 미사용 엔터티가 없게 됩니다. 단, 서비스 연결 역할에 대한 리소스를 먼저 정리해야 수동으로 삭제할 수 있습니다.

> **Note**
>
> 리소스를 삭제할 때 IAM Access Analyzer가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

AWSServiceRoleForAccessAnalyzer에서 사용하는 IAM Access Analyzer 리소스를 삭제하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. Access reports(보고서 액세스) 섹션의 Access analyzer(분석기 액세스)에서 Analyzers(분석기)를 선택합니다.
3. 분석기 테이블의 분석기 목록 위의 왼쪽 위에 있는 확인란을 선택하여 모든 분석기를 선택합니다.
4. 삭제를 선택합니다.
5. 분석기를 삭제할 것인지 확인하려면 **`delete`**을 입력한 다음, 삭제를 선택합니다.

IAM을 사용하여 수동으로 서비스 연결 역할 삭제

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 AWSServiceRoleForAccessAnalyzer 서비스 연결 역할을 삭제합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제 단원을 참조하세요.

IAM Access Analyzer 서비스 연결 역할이 지원되는 리전

IAM Access Analyzer에서는 서비스를 사용할 수 있는 모든 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 내용은 AWS 리전 및 엔드포인트 섹션을 참조하세요.

액세스 미리 보기

외부 엔터티와 공유되는 리소스를 식별하는 데 도움이 될 뿐만 아니라, 리소스 권한을 배포하기 전에 AWS IAM Access Analyzer 결과의 미리 보기를 제공하므로 정책 변경으로 인해 의도한 공용 및 크로스 계정 액세스 권한만 리소스에 부여되는지 확인할 수 있습니다. 이렇게 하면 리소스에 대한 외부 액세스를 의도한 대로 시작할 수 있습니다.

Amazon S3 콘솔에서 Amazon S3 버킷에 대한 퍼블릭 및 크로스 계정 액세스를 미리 보고 검증할 수 있습니다. 또한 IAM Access Analyzer API를 통해서도 리소스에 대한 제안된 권한을 제공하여 Amazon S3 버킷, AWS KMS 키, IAM 역할, Amazon SQS 대기열 및 Secrets Manager 암호에 대한 퍼블릭 및 크로스 계정 액세스를 미리 볼 수 있습니다.

주제

- Amazon S3 콘솔에서 액세스 미리 보기
- IAM Access Analyzer API를 사용하여 액세스 미리 보기

Amazon S3 콘솔에서 액세스 미리 보기

Amazon S3 콘솔에서 버킷 정책을 완료하면 Amazon S3 버킷에 대한 퍼블릭 및 크로스 계정 액세스를 미리 볼 수 있는 옵션이 있습니다. 정책 변경으로 의도한 외부 액세스 권한만 부여되는지 검증한 후에 변경 사항 저장(Save changes)을 선택할 수 있습니다. 이 선택적 단계를 사용하여 버킷의 AWS Identity and Access Management Access Analyzer 결과를 미리 볼 수 있습니다. 정책 변경으로 인해 외부 액세스에 대한 새로운 결과가 초래되는지 또는 기존 결과가 해결되는지 검증할 수 있습니다. 언제든 이 검증 단계를 건너뛰고 Amazon S3 버킷 정책을 저장할 수 있습니다.

버킷에 대한 외부 액세스를 미리 보려면 버킷 리전에 계정이 신뢰 영역인 활성 계정 분석기가 있어야 합니다. 또한 IAM Access Analyzer 및 액세스 미리 보기를 사용하는 데 필요한 권한이 있어야 합니다. 필요한 IAM Access Analyzer 및 권한을 활성화하는 방법에 대한 자세한 내용은 IAM Access Analyzer 활성화 섹션을 참조하세요.

버킷 정책을 생성하거나 편집할 때 Amazon S3 버킷에 대한 액세스를 미리 보려면

1. 버킷 정책 생성 또는 편집을 마치면 정책이 유효한 Amazon S3 버킷 정책인지 확인합니다. 정책 ARN이 버킷 ARN과 일치해야 하며 정책 요소가 유효해야 합니다.
2. 정책 아래의 외부 액세스 미리 보기(Preview external access)에서 활성 계정 분석기를 선택한 다음 미리 보기(Preview)를 선택합니다. 버킷에 대한 IAM Access Analyzer 결과의 미리 보기가 생성됩니다. 미리 보기에서는 표시된 Amazon S3 버킷 정책을 기존 버킷 권한과 함께 분석합니다. 여기에는 버킷 및 계정 BPA 설정, 버킷 ACL, 버킷에 연결된 Amazon S3 액세스 포인트 및 다중 리전 액세스 포인트, 해당하는 정책 및 BPA 설정이 포함됩니다.
3. 액세스 미리 보기가 완료되면 IAM Access Analyzer 결과의 미리 보기가 표시됩니다. 각 결과에서는 정책을 저장한 후 버킷에 액세스할 수 있는 계정 외부의 보안 주체 인스턴스를 보고합니다. 각 결과를 검토하여 버킷에 대한 액세스를 검증할 수 있습니다. 결과 헤더에는 액세스에 대한 요약이 제공되며 결과를 확장하여 결과 세부 정보를 검토할 수 있습니다. 결과 배지에는 버킷 정책 저장에 따라 버킷에 대한 액세스가 어떻게 변경되는지에 대한 컨텍스트가 제공됩니다. 예를 들어 정책 변

경으로 인해 외부 액세스에 대한 새로운 결과가 초래되는지 또는 기존 결과가 해결되는지 검증할 수 있습니다.

a. 신규(New) - 정책에서 도입하는 새로운 외부 액세스에 대한 결과를 나타냅니다.

b. 해결됨(Resolved) - 정책에서 제거하는 기존 외부 액세스에 대한 결과를 나타냅니다.

c. 보관됨(Archived) - 결과를 의도된 것으로 표시하는 경우를 정의하는 분석기의 아카이브 규칙에 따라 자동으로 보관되는 새 외부 액세스에 대한 결과를 나타냅니다.

d. 기존(Existing) - 변경되지 않고 유지되는 외부 액세스에 대한 기존 결과를 나타냅니다.

e. 퍼블릭(Public) - 리소스에 대한 퍼블릭 액세스에 관한 결과인 경우 위 배지 중 하나와 더불어 퍼블릭(Public) 배지도 표시됩니다.

4. 도입하거나 제거하지 않으려는 외부 액세스를 식별한 경우 의도하는 외부 액세스가 달성될 때까지 정책을 수정한 다음 미리 보기(Preview)를 다시 선택할 수 있습니다. 결과에 퍼블릭(Public) 레이블이 지정된 경우 변경 사항 저장(Save changes)을 선택하기 전에 정책을 수정하여 퍼블릭 액세스를 제거하는 것이 좋습니다. 액세스 미리 보기는 선택적 단계이며 언제든 변경 사항 저장(Save changes)을 선택할 수 있습니다.

IAM Access Analyzer API를 사용하여 액세스 미리 보기

IAM Access Analyzer API를 사용하여 Amazon S3 버킷, AWS KMS 키, IAM 역할, Amazon SQS 대기열 및 Secrets Manager 암호에 대한 퍼블릭 및 크로스 계정 액세스를 미리 볼 수 있습니다. 소유한 기존 리소스나 배포할 새 리소스에 대해 제안된 권한을 제공하여 액세스를 미리 볼 수 있습니다.

리소스에 대한 외부 액세스를 미리 보려면 리소스의 계정 및 리전에 대한 활성 계정 분석기가 있어야 합니다. 또한 IAM Access Analyzer 및 액세스 미리 보기를 사용하는 데 필요한 권한이 있어야 합니다. 필요한 IAM Access Analyzer 및 권한을 활성화하는 방법에 대한 자세한 내용은 IAM Access Analyzer 활성화 섹션을 참조하세요.

리소스에 대한 액세스를 미리 보려면 `CreateAccessPreview` 작업을 사용하고 분석기 ARN과 리소스에 대한 액세스 제어 구성을 제공할 수 있습니다. 이 서비스는 액세스 미리 보기를 위한 고유 ID를 반환하며, `GetAccessPreview` 작업에서 이 ID를 사용하여 액세스 미리 보기의 상태를 확인할 수 있습니다. 상태가 `Completed`이면 `ListAccessPreviewFindings` 작업을 사용하여 액세스 미리 보기에 대해 생성된 결과를 검색할 수 있습니다. `GetAccessPreview` 및 `ListAccessPreviewFindings` 작업은 약 24시간 내에 생성된 액세스 미리 보기 및 결과를 검색합니다.

검색된 각 결과에는 액세스를 설명하는 결과 세부 정보가 포함됩니다. 결과의 미리 보기 상태는 권한 배포 후 결과가 `Active`, `Archived` 또는 `Resolved`인지 여부와 `changeType`을 설명합니다.

changeType은 액세스 미리 보기 검색 결과가 IAM Access Analyzer에서 식별된 기존 액세스와 어떻게 비교되는지에 대한 컨텍스트를 제공합니다.

- New - 새로 도입된 액세스에 관한 결과입니다.
- Unchanged - 미리 보기 결과가 변경되지 않고 유지되는 기존 결과입니다.
- Changed - 미리 보기 결과가 상태가 변경된 기존 결과입니다.

status 및 changeType을 사용하면 리소스 구성으로 기존 리소스 액세스가 어떻게 변경되는지 이해할 수 있습니다. changeType이 Unchanged 또는 Changed이면 IAM Access Analyzer 결과의 기존 ID 및 상태도 결과에 포함됩니다. 예를 들어 미리 보기 상태 Resolved이고 기존 상태가 Active인 Changed 결과는 리소스에 대한 기존 Active 결과가 제안된 권한 변경의 결과로 Resolved가 된다는 것을 나타냅니다.

ListAccessPreviews 작업을 사용하여 지정된 분석기의 액세스 미리 보기 목록을 검색할 수 있습니다. 이 작업은 약 1시간 내에 생성된 액세스 미리 보기에 대한 정보를 검색합니다.

일반적으로 액세스 미리 보기가 기존 리소스에 관한 것이고 구성 옵션을 지정하지 않은 상태로 두면 액세스 미리 보기에서는 기본적으로 기존 리소스 구성을 사용합니다. 액세스 미리 보기가 새 리소스에 관한 것이고 구성 옵션을 지정하지 않은 상태로 두면 액세스 미리 보기에서는 리소스 유형에 대한 기본값을 사용합니다. 각 리소스 유형의 구성 사례는 아래를 참조하세요.

Amazon S3 버킷에 대한 액세스 미리 보기

새 Amazon S3 버킷 또는 소유한 기존 Amazon S3 버킷에 대한 액세스 미리 보기를 생성하려면 버킷에 연결된 Amazon S3 버킷 정책, 버킷 ACL, 버킷 BPA 설정 및 Amazon S3 액세스 포인트(다중 리전 액세스 포인트 포함)를 지정하여 버킷 구성을 제안할 수 있습니다.

> **Note**
>
> 새 버킷에 대한 액세스 미리 보기를 생성하기 전에 Amazon S3 HeadBucket 작업을 호출하여 명명된 버킷이 이미 있는지 확인하는 것이 좋습니다. 이 작업은 버킷이 있고 버킷에 액세스할 수 있는 권한이 있는지 확인하는 데 유용합니다.

버킷 정책 - 구성이 기존 Amazon S3 버킷에 관한 구성이고 Amazon S3 버킷 정책을 지정하지 않는 경우 액세스 미리 보기에서는 버킷에 연결된 기존 정책을 사용합니다. 액세스 미리 보기가 새 리소스에 대한 미리 보기이고 Amazon S3 버킷 정책을 지정하지 않는 경우 액세스 미리 보기에서는 정책이 없는

버킷을 가정합니다. 기존 버킷 정책의 삭제를 제안하려면 빈 문자열을 지정할 수 있습니다. 지원되는 버킷 정책 제한에 대한 자세한 내용은 버킷 정책 예를 참조하세요.

버킷 ACL 권한 - 버킷당 최대 100개의 ACL 권한을 제안할 수 있습니다. 제안된 권한 구성이 기존 버킷에 대한 것이면 액세스 미리 보기에서는 기존 권한 대신 제안된 권한 구성 목록을 사용합니다. 그렇지 않으면 액세스 미리 보기에서는 버킷의 기존 권한을 사용합니다.

버킷 액세스 포인트 - 분석에서는 버킷당 최대 100개의 액세스 포인트(다중 리전 액세스 포인트 포함)를 지원합니다. 여기에는 버킷당 제안할 수 있는 최대 10개의 새 액세스 포인트가 포함됩니다. 제안된 Amazon S3 구성이 기존 버킷에 대한 것이면 액세스 미리 보기에서는 기존 액세스 포인트 대신 제안된 액세스 포인트 구성을 사용합니다. 정책 없이 액세스 포인트를 제안하려면 빈 문자열을 액세스 포인트 정책으로 제공할 수 있습니다. 액세스 포인트 정책 제한에 대한 자세한 내용은 액세스 포인트 규제 및 제한을 참조하세요.

퍼블릭 액세스 차단 구성 - 제안된 구성이 기존 Amazon S3 버킷에 대한 것이고 구성을 지정하지 않는 경우 액세스 미리 보기에서는 기존 설정을 사용합니다. 제안된 구성이 새 버킷에 대한 구성이고 버킷 BPA 구성을 지정하지 않는 경우 액세스 미리 보기에서는 `false`를 사용합니다. 제안된 구성이 새 액세스 포인트 또는 다중 리전 액세스 포인트에 대한 구성이고 액세스 포인트 BPA 구성을 지정하지 않는 경우 액세스 미리 보기에서는 `true`를 사용합니다.

AWS KMS 키에 대한 액세스 미리 보기

새 AWS KMS 키 또는 소유하는 기존 AWS KMS 키에 대한 액세스 미리 보기를 생성하려면 키 정책과 AWS KMS 권한 구성을 지정하여 AWS KMS 키 구성을 제안할 수 있습니다.

AWS KMS 키 정책 - 구성이 기존 키에 대한 구성이고 키 정책을 지정하지 않는 경우 액세스 미리 보기에서는 키의 기존 정책을 사용합니다. 액세스 미리 보기가 새 리소스에 대한 미리 보기이고 키 정책을 지정하지 않는 경우 액세스 미리 보기에서는 기본 키 정책을 사용합니다. 제안된 키 정책은 빈 문자열이어서는 안 됩니다.

AWS KMS 권한 - 분석에서는 구성당 최대 100개의 KMS 권한을 지원합니다*.* 제안된 권한 구성이 기존 키에 대한 구성이면 액세스 미리 보기에서는 기존 권한 대신 제안된 권한 구성 목록을 사용합니다. 그렇지 않으면 액세스 미리 보기에서는 키의 기존 권한을 사용합니다.

IAM 역할에 대한 액세스 미리 보기

새 IAM 역할 또는 소유하는 기존 IAM 역할에 대한 액세스 미리 보기를 생성하려면 신뢰 정책을 지정하여 IAM 역할 구성을 제안할 수 있습니다.

역할 신뢰 정책 - 구성이 새 IAM 역할에 대한 구성인 경우 신뢰 정책을 지정해야 합니다. 구성이 소유한 기존 IAM 역할에 대한 구성이고 신뢰 정책을 제안하지 않는 경우 액세스 미리 보기에서는 역할의 기존 신뢰 정책을 사용합니다. 제안된 신뢰 정책은 빈 문자열이어서는 안 됩니다.

Amazon SQS 대기열에 대한 액세스 미리 보기

새 Amazon SQS 대기열 또는 소유한 기존 Amazon SQS 대기열에 대한 액세스 미리 보기를 생성하려면 대기열의 Amazon SQS 정책을 지정하여 Amazon SQS 대기열 구성을 제안할 수 있습니다.

Amazon SQS 대기열 정책 - 구성이 기존 Amazon SQS 대기열에 대한 구성이고 Amazon SQS 정책을 지정하지 않는 경우 액세스 미리 보기에서는 대기열의 기존 Amazon SQS 정책을 사용합니다. 액세스 미리 보기가 새 리소스에 대한 미리 보기이고 정책을 지정하지 않는 경우 액세스 미리 보기에서는 정책이 없는 Amazon SQS 대기열을 가정합니다. 기존 Amazon SQS 대기열 정책의 삭제를 제안하려면 Amazon SQS 정책에 빈 문자열을 지정할 수 있습니다.

Secrets Manager 암호에 대한 액세스 미리 보기

새 Secrets Manager 암호 또는 소유한 기존 Secrets Manager 암호에 대한 액세스 미리 보기를 생성하려면 암호 정책 및 AWS KMS 암호화 키(선택 사항)를 지정하여 Secrets Manager 암호 구성을 제안할 수 있습니다.

암호 정책 - 구성이 기존 암호에 대한 구성이고 암호 정책을 지정하지 않는 경우 액세스 미리 보기에서는 암호의 기존 정책을 사용합니다. 액세스 미리 보기가 새 리소스에 대한 미리 보기이고 정책을 지정하지 않는 경우 액세스 미리 보기에서는 정책이 없는 암호를 가정합니다. 기존 정책의 삭제를 제안하려면 빈 문자열을 지정할 수 있습니다.

AWS KMS 암호화 키 - 제안된 구성이 새 암호에 대한 구성이고 AWS KMS 키 ID를 지정하지 않는 경우 액세스 미리 보기에서는 AWS 계정의 기본 KSM 키를 사용합니다. AWS KMS 키 ID에 빈 문자열을 지정하는 경우 액세스 미리 보기에서는 AWS 계정의 기본 KMS 키를 사용합니다.

정책 검증 검사

IAM Access Analyzer는 IAM 정책을 엔터티에 연결하기 전에 검증하는 데 도움이 되는 정책 확인을 제공합니다. 여기에는 정책 문법 및 AWS 모범 사례에 따라 정책을 검증하기 위해 정책 검증을 통해 제공되는 기본 정책 확인이 포함됩니다. 보안 경고, 오류, 일반 경고 및 정책에 대한 제안 사항이 포함된 정책 유효성 검사 결과를 볼 수 있습니다.

사용자 지정 정책 확인을 사용하여 보안 표준에 따라 새 액세스를 확인할 수 있습니다. 새 액세스를 확인할 때마다 요금이 부과됩니다. 요금에 대한 자세한 내용은 IAM Access Analyzer 요금을 참조하세요.

주제

IAM Access Analyzer 정책 검증

AWS Identity and Access Management Access Analyzer 정책 검증을 사용하여 정책을 검증할 수 있습니다. AWS CLI, AWS API 또는 IAM 콘솔의 JSON 정책 편집기를 사용하여 정책을 생성 또는 편집할 수 있습니다. IAM Access Analyzer가 IAM 정책 문법 및 AWS 모범 사례에 대해 정책을 검증할 수 있습니다. 보안 경고, 오류, 일반 경고 및 정책에 대한 제안 사항이 포함된 정책 유효성 검사 결과를 볼 수 있습니다. 이러한 검색 결과는 보안 모범 사례 준수 기능을 갖춘 정책을 작성하는 데 도움이 되는 실행 가능한 권장 사항을 제공합니다. IAM Access Analyzer에서 실행되는 기본 정책 확인 목록을 보려면 Access Analyzer 정책 검사 참조 섹션을 참조하세요.

IAM에서 정책 검증(콘솔)

IAM 콘솔에서 관리형 정책을 생성하거나 편집할 때 IAM Access Analyzer 정책 확인에서 생성되는 조사 결과를 볼 수 있습니다. 또한 인라인 사용자 또는 역할 정책에 대해서도 이러한 결과를 볼 수 있습니다. IAM Access Analyzer는 인라인 그룹 정책의 경우에는 이러한 조사 결과를 생성하지 않습니다.

IAM JSON 정책에 대한 정책 검사에서 생성되는 결과를 보려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 다음 방법 중 하나를 사용하여 정책 생성이나 편집을 시작합니다.
    a. 새 관리형 정책을 생성하려면 정책 페이지로 이동하여 새 정책을 생성합니다. 자세한 내용은 JSON 편집기를 사용하여 정책 생성 섹션을 참조하세요.
    b. 기존 고객 관리형 정책에 대한 정책 검사를 보려면 정책 페이지로 이동하여 정책 이름을 선택한 다음 편집을 선택합니다. 자세한 내용은 고객 관리형 정책 편집(콘솔) 섹션을 참조하세요.
    c. 사용자 또는 역할의 인라인 정책에 대한 정책 검사를 보려면 사용자 또는 역할 페이지로 이동하여 사용자 또는 역할의 이름을 선택하고, 권한 탭에서 정책의 이름을 선택한 다음 편집을 선택합니다. 자세한 내용은 고객 관리형 정책 편집(콘솔) 섹션을 참조하세요.
3. 정책 편집기에서 JSON 탭을 선택합니다.
4. 정책 아래의 정책 검증 창에서 다음 탭 중 하나 이상을 선택합니다. 탭 이름에는 정책에 대한 각 결과 유형의 수도 표시됩니다.

- 보안(Security) - 액세스가 지나치게 허용적이어서 AWS에서 보안 위험으로 간주하는 액세스를 정책에서 허용하는 경우 경고가 표시됩니다.
- 오류(Errors) - 정책이 작동하지 않게 하는 행이 정책에 포함된 경우 오류가 표시됩니다.
- 경고 - 정책이 모범 사례를 준수하지 않지만 보안 위험은 아닌 문제인 경우 경고가 표시됩니다.
- 제안(Suggestions) - AWS에서 정책의 권한에 영향을 주지 않는 개선 사항을 권장하는 경우 제안 사항이 표시됩니다.

5. IAM Access Analyzer 정책 검사에서 제공되는 결과 세부 정보를 검토합니다. 각 결과에는 보고된 문제의 위치가 표시됩니다. 문제의 원인과 해결 방법에 대해 자세히 알아보려면 결과 옆의 자세히 알아보기(Learn more) 링크를 선택합니다. Access Analyzer 정책 검사 참조 페이지에서 각 결과와 연결된 정책 검사를 검색할 수도 있습니다.
6. 선택 사항으로, 기존 정책을 편집하는 경우 사용자 지정 정책 확인을 실행하여 업데이트된 정책이 기존 버전과 비교하여 새 액세스를 허용하는지 확인할 수 있습니다. 정책 아래의 정책 검증 창에서 새 액세스 확인 탭을 선택한 다음 정책 확인을 선택합니다. 수정된 권한으로 새 액세스 권한이 부여되는 경우 정책 검증 창에서 해당 설명이 강조 표시됩니다. 새 액세스 권한을 부여하지 않으려면 새 액세스가 감지되지 않을 때까지 정책 설명을 업데이트하고 정책 확인을 선택합니다. 자세한 내용은 IAM Access Analyzer 정책 확인 섹션을 참조하세요.

> **Note**
>
> 새 액세스를 확인할 때마다 요금이 부과됩니다. 요금에 대한 자세한 내용은 IAM Access Analyzer 요금을 참조하세요.

7. 정책을 업데이트하여 결과를 해결합니다.

> **Important**
>
> 새 정책이나 편집된 정책은 철저한 테스트를 거친 후에 프로덕션 워크플로에서 구현합니다.

8. 마쳤으면 [Next]를 선택합니다. 정책 검사기는 IAM Access Analyzer에서 보고하지 않는 구문 오류를 보고합니다.

> **Note**
>
> 언제든지 Visual 탭과 JSON 탭 간 전환이 가능합니다. 그러나 변경을 적용하거나 시각적 탭에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화되도록 정책을 재구성할 수 있습니다. 자세한 내용은 정책 재구성 섹션을 참조하세요.

9. 새 정책의 경우, 검토 및 생성 페이지에서 생성하는 정책에 대한 정책 이름과 설명(선택 사항)을 입력합니다. 이 정책에 정의된 권한을 검토하여 정책이 부여한 권한을 확인합니다. 그런 다음 정책 생성을 선택하여 작업을 저장합니다.

   기존의 정책의 경우 검토 및 저장 페이지에서 이 정책에 정의된 권한을 검토하여 정책이 부여한 권한을 확인합니다. 새 버전을 기본값으로 설정합니다. 확인란을 선택하여 업데이트된 버전을 정책의 기본 버전으로 저장합니다. 그런 다음 변경 사항 저장을 선택하여 변경 사항을 저장합니다.

IAM Access Analyzer를 사용하여 정책 검증(AWS CLI 또는 AWS API)

IAM Access Analyzer 정책 검증에서 생성된 조사 결과를 AWS Command Line Interface(AWS CLI)에서 볼 수 있습니다.

IAM Access Analyzer 정책 검증에서 생성된 조사 결과를 보려면(AWS CLI 또는 AWS API)

다음 중 하나를 사용하세요.

- AWS CLI: aws accessanalyzer validate-policy
- AWS API: ValidatePolicy

Access Analyzer 정책 검사 참조

AWS Identity and Access Management Access Analyzer 정책 검증을 사용하여 정책을 검증할 수 있습니다. AWS CLI, AWS API 또는 IAM 콘솔의 JSON 정책 편집기를 사용하여 정책을 생성 또는 편집할 수 있습니다. IAM Access Analyzer가 IAM 정책 문법 및 AWS 모범 사례에 대해 정책을 검증할 수 있습니다. 보안 경고, 오류, 일반 경고 및 정책에 대한 제안 사항이 포함된 정책 유효성 검사 결과를 볼 수 있습니다. 이러한 검색 결과는 보안 모범 사례 준수 기능을 갖춘 정책을 작성하는 데 도움이 되는 실행 가능한 권장 사항을 제공합니다. IAM Access Analyzer가 제공하는 기본 정책 확인 목록은 다음과 같습니다. 정책 검증 검사 실행과 관련된 추가 요금은 없습니다. 정책 검증을 사용하여 정책을 검증하는 방법에 대한 자세한 내용은 IAM Access Analyzer 정책 검증 섹션을 참조하세요.

오류 - ARN 계정이 허용되지 않음

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
ARN account not allowed: The service {{service}} does not support specifying an account
 ID in the resource ARN.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The service {{service}} does not support specifying an account ID in
 the resource ARN."
```

오류 해결

리소스 ARN에서 계정 ID를 제거합니다. 일부 AWS 서비스의 리소스 ARN에서는 계정 ID 지정을 지원하지 않습니다.

예를 들어 Amazon S3에서는 버킷 ARN에 계정 ID를 네임스페이스로 지정할 수 없습니다. Amazon S3 버킷 이름은 전역적으로 고유하며 네임스페이스는 모든 AWS 계정에서 공유됩니다. Amazon S3에서 사용할 수 있는 모든 리소스 유형을 보려면 서비스 승인 참조의 Amazon S3에서 정의한 리소스 유형을 참조하세요.

관련 용어

- 정책 리소스
- 계정 식별자
- 리소스 ARN
- ARN 형식의 AWS 서비스 리소스

오류 - ARN 리전이 허용되지 않음

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
ARN Region not allowed: The service {{service}} does not support specifying a Region in
 the resource ARN.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The service {{service}} does not support specifying a Region in the
 resource ARN."
```

오류 해결

리소스 ARN에서 리전을 제거합니다. 일부 AWS 서비스의 리소스 ARN에서는 리전 지정을 지원하지 않습니다.

예를 들어 IAM은 글로벌 서비스입니다. IAM 리소스 ARN의 리전 부분은 항상 비워 둡니다. IAM 리소스는 현재의 AWS 계정처럼 글로벌 리소스입니다. 예를 들어 IAM 사용자로 로그인하면 어느 지리적 리전에서나 AWS 서비스에 액세스할 수 있습니다.

- 정책 리소스
- 리소스 ARN
- ARN 형식의 AWS 서비스 리소스

오류 - 데이터 형식 불일치

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Data type mismatch: The text does not match the expected JSON data type {{data_type}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The text does not match the expected JSON data type {{data_type}}."
```

오류 해결

지원되는 데이터 형식을 사용하도록 텍스트를 업데이트합니다.

예를 들어 `Version` 전역 조건 키에는 `String` 데이터 형식이 필요합니다. 날짜 또는 정수를 지정하면 데이터 형식이 일치하지 않습니다.

관련 용어

- 전역 조건 키
- IAM JSON 정책 요소: 조건 연산자

오류 - 대/소문자가 다른 중복 키

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Duplicate keys with different case: The condition key {{key}} appears more than once
 with different capitalization in the same condition block. Remove the duplicate
 condition keys.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The condition key {{key}} appears more than once with different
 capitalization in the same condition block. Remove the duplicate condition keys."
```

오류 해결

동일한 조건 블록 내에서 유사한 조건 키를 검토하고 모든 인스턴스에 동일한 대/소문자를 사용합니다.

조건 블록은 정책 문 `Condition` 요소 내의 텍스트입니다. 조건 키 이름은 대/소문자를 구분하지 않습니다. 조건 키 값의 대/소문자 구분은 사용하는 조건 연산자에 따라 다릅니다. 조건 키에서 대/소문자 구분에 대한 자세한 내용은 IAM JSON 정책 요소: Condition 섹션을 참조하세요.

관련 용어

- 조건
- 조건 블록
- 전역 조건 키
- AWS 서비스 조건 키

오류 - 잘못된 작업

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid action: The action {{action}} does not exist. Did you mean {{valid_action}}?
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The action {{action}} does not exist. Did you mean
 {{valid_action}}?"
```

오류 해결

지정한 작업이 유효하지 않습니다. 서비스 접두사 또는 작업 이름을 잘못 입력하면 발생할 수 있습니다. 몇 가지 일반적인 문제의 경우 정책 검사에서 추천 작업을 반환합니다.

관련 용어

- 정책 작업
- AWS 서비스 작업

이 오류가 있는 AWS 관리형 정책

AWS 관리형 정책을 사용하면 일반 AWS 사용 사례에 따라 권한을 할당하여 AWS를 시작할 수 있습니다.

다음 AWS 관리형 정책은 잘못된 작업을 해당 정책 문에 포함합니다. 잘못된 작업은 정책에 따라 부여되는 권한에 영향을 미치지 않습니다. AWS 관리형 정책을 참조하여 관리형 정책을 생성하는 경우 AWS는 정책에서 잘못된 작업을 제거할 것을 권장합니다.

- AmazonEMRFullAccessPolicy_v2
- CloudWatchSyntheticsFullAccess

오류 - 잘못된 ARN 계정

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid ARN account: The resource ARN account ID {{account}} is not valid. Provide a
 12-digit account ID.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The resource ARN account ID {{account}} is not valid. Provide a 12-
digit account ID."
```

오류 해결

리소스 ARN에서 계정 ID를 업데이트합니다. 계정 ID는 12자리 정수입니다. 계정 ID를 확인하는 방법은 AWS 계정 ID 찾기를 참조하세요.

관련 용어

- 정책 리소스
- 계정 식별자
- 리소스 ARN
- ARN 형식의 AWS 서비스 리소스

오류 - 잘못된 ARN 접두사

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid ARN prefix: Add the required prefix (arn) to the resource ARN.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Add the required prefix (arn) to the resource ARN."
```

오류 해결

AWS 리소스 ARN에는 필수 `arn:` 접두사를 포함해야 합니다.

관련 용어

- 정책 리소스
- 리소스 ARN
- ARN 형식의 AWS 서비스 리소스

오류 - 잘못된 ARN 리전

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid ARN Region: The Region {{region}} is not valid for this resource. Update the
 resource ARN to include a supported Region.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The Region {{region}} is not valid for this resource. Update the
 resource ARN to include a supported Region."
```

오류 해결

지정된 리전에서 리소스 유형이 지원되지 않습니다. 각 리전에서 지원되는 AWS 서비스를 정리한 표는 리전 표를 참조하세요.

관련 용어

- 정책 리소스
- 리소스 ARN
- 리전 이름 및 코드

오류 - 잘못된 ARN 리소스

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid ARN resource: Resource ARN does not match the expected ARN format. Update the
 resource portion of the ARN.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Resource ARN does not match the expected ARN format. Update the
 resource portion of the ARN."
```

오류 해결

리소스 ARN은 알려진 리소스 유형의 사양과 일치해야 합니다. 서비스의 예상 ARN 형식을 확인하려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요. 서비스 이름을 선택하여 해당 리소스 유형 및 ARN 형식을 확인합니다.

관련 용어

- 정책 리소스

- 리소스 ARN
- ARN 형식의 AWS 서비스 리소스

오류 - 잘못된 ARN 서비스 사례

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid ARN service case: Update the service name ${service} in the resource ARN to use
 all lowercase letters.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Update the service name ${service} in the resource ARN to use all
 lowercase letters."
```

오류 해결

리소스 ARN의 서비스는 서비스 접두사의 사양(대/소문자 포함)과 일치해야 합니다. 서비스의 접두사를 확인하려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요. 서비스의 이름을 선택하고 첫 번째 문장에서 해당 접두사를 찾습니다.

관련 용어

- 정책 리소스
- 리소스 ARN
- ARN 형식의 AWS 서비스 리소스

오류 - 잘못된 조건 데이터 형식

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid condition data type: The condition value data types do not match. Use condition
 values of the same JSON data type.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The condition value data types do not match. Use condition values of
 the same JSON data type."
```

오류 해결

조건 키 값 페어의 값은 조건 키 및 조건 연산자의 데이터 형식과 일치해야 합니다. 서비스의 조건 키 데이터 형식을 확인하려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요. 서비스의 이름을 선택하여 해당 서비스의 조건 키를 확인합니다.

예를 들어 `CurrentTime` 전역 조건 키는 `Date` 조건 연산자를 지원합니다. 조건 블록에서 값에 문자열이나 정수를 지정하면 데이터 형식이 일치하지 않습니다.

관련 용어

- 조건
- 조건 블록
- IAM JSON 정책 요소: 조건 연산자
- 전역 조건 키
- AWS 서비스 조건 키

오류 - 잘못된 조건 키 형식

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid condition key format: The condition key format is not valid. Use the format
 service:keyname.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The condition key format is not valid. Use the format
 service:keyname."
```

오류 해결

조건 키 값 페어의 키는 서비스의 사양과 일치해야 합니다. 서비스의 조건 키를 확인하려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요. 서비스의 이름을 선택하여 해당 서비스의 조건 키를 확인합니다.

관련 용어

- 조건
- 전역 조건 키
- AWS 서비스 조건 키

오류 - 잘못된 조건 다중 부울

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid condition multiple Boolean: The condition key does not support multiple Boolean
 values. Use a single Boolean value.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The condition key does not support multiple Boolean values. Use a
 single Boolean value."
```

오류 해결

조건 키 값 페어의 키에는 단일 부울 값이 필요합니다. 다중 부울 값을 지정하면 조건 일치에서 예상하는 결과가 반환되지 않을 수 있습니다.

서비스의 조건 키를 확인하려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요. 서비스의 이름을 선택하여 해당 서비스의 조건 키를 확인합니다.

- 조건
- 전역 조건 키
- AWS 서비스 조건 키

오류 - 잘못된 조건 연산자

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid condition operator: The condition operator {{operator}} is not valid. Use a
 valid condition operator.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The condition operator {{operator}} is not valid. Use a valid
 condition operator."
```

오류 해결

지원되는 조건 연산자를 사용하도록 조건을 업데이트합니다.

관련 용어

- IAM JSON 정책 요소: 조건 연산자
- 조건 요소
- JSON 정책 개요

오류 - 잘못된 효과

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid effect: The effect {{effect}} is not valid. Use Allow or Deny.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The effect {{effect}} is not valid. Use Allow or Deny."
```

오류 해결

유효한 효과를 사용하도록 `Effect` 요소를 업데이트합니다. `Effect` 유효값은 **`Allow`** 및 **`Deny`**입니다.

관련 용어

- Effect 요소
- JSON 정책 개요

오류 - 잘못된 전역 조건 키

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid global condition key: The condition key {{key}} does not exist. Use a valid
 condition key.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The condition key {{key}} does not exist. Use a valid condition
 key."
```

오류 해결

지원되는 전역 조건 키를 사용하도록 조건 키 값 페어의 조건 키를 업데이트합니다.

전역 조건 키는 `aws:` 접두사가 있는 조건 키입니다. AWS 서비스는 전역 조건 키를 지원하거나 서비스 접두사를 포함하는 서비스별 키를 제공할 수 있습니다. 예를 들어 IAM 조건 키에는 `iam:` 접두사가 포함됩니다. 자세한 내용은 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하고 키를 보려는 서비스를 선택합니다.

관련 용어

- 전역 조건 키

오류 - 잘못된 파티션

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid partition: The resource ARN for the service {{service}} does not support the
 partition {{partition}}. Use the supported values: {{partitions}}
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The resource ARN for the service {{service}} does not support the
 partition {{partition}}. Use the supported values: {{partitions}}"
```

오류 해결

지원되는 파티션을 포함하도록 리소스 ARN을 업데이트합니다. 지원되는 파티션을 포함한 경우에는 포함한 파티션을 서비스 또는 리소스에서 지원하지 않을 수 있습니다.

파티션은 AWS 리전의 그룹입니다. 각 AWS 계정은 하나의 파티션으로 범위가 지정됩니다. Classic 리전에서는 aws 파티션을 사용합니다. 중국 리전에서는 aws-cn을 사용합니다.

관련 용어

- Amazon 리소스 이름(ARN) - 파티션

오류 - 잘못된 정책 요소

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid policy element: The policy element {{element}} is not valid.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The policy element {{element}} is not valid."
```

오류 해결

지원되는 JSON 정책 요소만 포함하도록 정책을 업데이트합니다.

관련 용어

- JSON 정책 요소

오류 - 잘못된 보안 주체 형식

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid principal format: The Principal element contents are not valid. Specify a key-
value pair in the Principal element.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The Principal element contents are not valid. Specify a key-value
 pair in the Principal element."
```

오류 해결

지원되는 키 값 페어 형식을 사용하도록 보안 주체를 업데이트합니다.

보안 주체는 리소스 기반 정책에서 지정할 수 있고 자격 증명 기반 정책에서는 지정할 수 없습니다.

예를 들어 모든 AWS 계정의 모든 사용자에 대한 액세스 권한을 정의하려면 정책에 다음 보안 주체를 사용합니다.

```
"Principal": { "AWS": "123456789012" }
```

관련 용어

- JSON 정책 요소: Principal
- 자격 증명 기반 정책 및 리소스 기반 정책

오류 - 잘못된 보안 주체 키

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid principal key: The principal key {{principal-key}} is not valid.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The principal key {{principal-key}} is not valid."
```

오류 해결

지원되는 보안 주체 키를 사용하도록 보안 주체 키 값 페어의 키를 업데이트합니다. 지원되는 보안 주체 키는 다음과 같습니다.

- AWS
- CanonicalUser
- 연동됨
- Service

관련 용어

- Principal 요소

오류 - 잘못된 리전

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid Region: The Region {{region}} is not valid. Update the condition value to a
 suported Region.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The Region {{region}} is not valid. Update the condition value to a
 suported Region."
```

오류 해결

지원되는 리전을 포함하도록 조건 키 값 페어의 값을 업데이트합니다. 각 리전에서 지원하는 AWS 서비스를 정리한 표는 리전 표를 참조하세요.

관련 용어

- 정책 리소스
- 리소스 ARN
- 리전 이름 및 코드

오류 - 잘못된 서비스

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid service: The service {{service}} does not exist. Use a valid service name.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The service {{service}} does not exist. Use a valid service name."
```

오류 해결

작업 또는 조건 키의 서비스 접두사는 서비스 접두사의 사양(대/소문자 포함)과 일치해야 합니다. 서비스의 접두사를 확인하려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요. 서비스의 이름을 선택하고 첫 번째 문장에서 해당 접두사를 찾습니다.

관련 용어

- 알려진 서비스와 해당 작업, 리소스 및 조건 키

오류 - 잘못된 서비스 조건 키

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid service condition key: The condition key {{key}} does not exist in the service
 {{service}}. Use a valid condition key.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The condition key {{key}} does not exist in the service {{service}}.
 Use a valid condition key."
```

오류 해결

서비스의 알려진 조건 키를 사용하도록 조건 키 값 페어의 키를 업데이트합니다. 전역 조건 키 이름은 aws 접두사로 시작합니다. AWS 서비스에서는 서비스 접두사를 포함하는 서비스별 키를 제공할 수 있습니다. 서비스의 접두사를 확인하려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

관련 용어

- 전역 조건 키
- 알려진 서비스와 해당 작업, 리소스 및 조건 키

오류 - 작업의 잘못된 서비스

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid service in action: The service {{service}} specified in the action does not
 exist. Did you mean {{service2}}?
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The service {{service}} specified in the action does not exist. Did
 you mean {{service2}}?"
```

오류 해결

작업의 서비스 접두사는 서비스 접두사의 사양(대/소문자 포함)과 일치해야 합니다. 서비스의 접두사를 확인하려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요. 서비스의 이름을 선택하고 첫 번째 문장에서 해당 접두사를 찾습니다.

관련 용어

- Action 요소
- 알려진 서비스 및 해당 작업

오류 - 연산자의 잘못된 변수

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid variable for operator: Policy variables can only be used with String and ARN
 operators.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Policy variables can only be used with String and ARN operators."
```

오류 해결

정책 변수는 `Resource` 요소를 비롯해 `Condition` 요소의 문자열 비교에 사용할 수 있습니다. 문자열 연산자 또는 ARN 연산자를 사용하는 경우 조건에서 변수를 지원합니다. 문자열 연산자에는 `StringEquals`, `StringLike` 및 `StringNotLike`가 포함됩니다. ARN 연산자에는 `ArnEquals` 및 `ArnLike`가 포함됩니다. 숫자, 날짜, 부울, 이진, IP 주소 또는 Null 연산자 등과 같은 다른 연산자에는 정책 변수를 사용할 수 없습니다.

관련 용어

- Condition 요소에서 정책 변수 사용
- 조건 요소

오류 - 잘못된 버전

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid version: The version ${version} is not valid. Use one of the following
 versions: ${versions}
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The version ${version} is not valid. Use one of the following
 versions: ${versions}"
```

오류 해결

`Version` 정책 요소는 AWS에서 정책을 처리하는 데 사용할 언어 구문 규칙을 지정합니다. 사용 가능한 모든 정책 기능을 사용하려면 모든 정책에서 `Statement` 요소 앞에 최신 `Version` 요소를 포함합니다.

```
"Version": "2012-10-17"
```

관련 용어

- 버전 요소

오류 - JSON 구문 오류

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Json syntax error: Fix the JSON syntax error at index {{index}} line {{line}} column
 {{column}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Fix the JSON syntax error at index {{index}} line {{line}} column
 {{column}}."
```

오류 해결

정책에 구문 오류가 있습니다. JSON 구문을 확인하세요.

관련 용어

- JSON 검사기
- IAM JSON 정책 요소 참조
- JSON 정책 개요

오류 - JSON 구문 오류

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Json syntax error: Fix the JSON syntax error.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Fix the JSON syntax error."
```

오류 해결

정책에 구문 오류가 있습니다. JSON 구문을 확인하세요.

관련 용어

- JSON 검사기
- IAM JSON 정책 요소 참조
- JSON 정책 개요

오류 - 작업 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing action: Add an Action or NotAction element to the policy statement.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Add an Action or NotAction element to the policy statement."
```

오류 해결

AWS JSON 정책에는 `Action` 또는 `NotAction` 요소가 있어야 합니다.

관련 용어

- Action 요소
- NotAction 요소
- JSON 정책 개요

오류 - ARN 필드 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing ARN field: Resource ARNs must include at least {{fields}} fields in the
 following structure: arn:partition:service:region:account:resource
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Resource ARNs must include at least {{fields}} fields in the
 following structure: arn:partition:service:region:account:resource"
```

오류 해결

리소스 ARN의 모든 필드는 알려진 리소스 유형의 사양과 일치해야 합니다. 서비스의 예상 ARN 형식을 확인하려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요. 서비스 이름을 선택하여 해당 리소스 유형 및 ARN 형식을 확인합니다.

관련 용어

- 정책 리소스

- 리소스 ARN
- ARN 형식의 AWS 서비스 리소스

오류 - ARN 리전 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing ARN Region: Add a Region to the {{service}} resource ARN.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Add a Region to the {{service}} resource ARN."
```

오류 해결

대부분 AWS 서비스의 리소스 ARN에서는 리전을 지정해야 합니다. 각 리전에서 지원하는 AWS 서비스를 정리한 표는 리전 표를 참조하세요.

관련 용어

- 정책 리소스
- 리소스 ARN
- 리전 이름 및 코드

오류 - 효과 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing effect: Add an Effect element to the policy statement with a value of Allow or
 Deny.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Add an Effect element to the policy statement with a value of Allow
 or Deny."
```

오류 해결

AWS JSON 정책에는 값이 **`Allow`** 및 **`Deny`**인 `Effect` 요소가 있어야 합니다.

관련 용어

- Effect 요소
- JSON 정책 개요

오류 - 보안 주체 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing principal: Add a Principal element to the policy statement.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Add a Principal element to the policy statement."
```

오류 해결

리소스 기반 정책에는 `Principal` 요소가 있어야 합니다.

예를 들어 모든 AWS 계정의 모든 사용자에 대한 액세스 권한을 정의하려면 정책에 다음 보안 주체를 사용합니다.

```
"Principal": { "AWS": "123456789012" }
```

관련 용어

- Principal 요소
- 자격 증명 기반 정책 및 리소스 기반 정책

오류 - 한정자 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing qualifier: The request context key ${key} has multiple values. Use the
 ForAllValues or ForAnyValue condition key qualifiers in your policy.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The request context key ${key} has multiple values. Use the
 ForAllValues or ForAnyValue condition key qualifiers in your policy."
```

오류 해결

Condition 요소에서 같음, 보다 작음 등의 조건 연산자를 사용하여 정책의 조건을 요청 컨텍스트의 키 및 값과 비교하는 표현식을 작성합니다. 단일 조건 키에 다수의 값이 포함된 요청일 경우에는 배열("Key2":["Value2A", "Value2B"])처럼 조건을 대괄호로 묶어야 합니다. 또한 ForAllValues 또는 ForAnyValue 집합 연산자는 StringLike 조건 연산자와 함께 사용해야 합니다. 이러한 한정자는 조건 연산자에 설정 작업 기능을 추가하므로 여러 요청 값을 여러 조건 값에 대해 테스트할 수 있습니다.

관련 용어

- 다중 값 컨텍스트 키
- 조건 요소

이 오류가 있는 AWS 관리형 정책

AWS 관리형 정책을 사용하면 일반 AWS 사용 사례에 따라 권한을 할당하여 AWS를 시작할 수 있습니다.

다음 AWS 관리형 정책은 해당 정책 문에 조건 키의 한정자가 누락되어 있습니다. AWS 관리형 정책을 참조로 사용하여 고객 관리형 정책을 생성하는 경우 AWS에서는 Condition 요소에 ForAllValues 또는 ForAnyValue 조건 키를 추가하는 것을 권장합니다.

- AWSGlueConsoleSageMakerNotebookFullAccess

오류 - 리소스 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing resource: Add a Resource or NotResource element to the policy statement.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Add a Resource or NotResource element to the policy statement."
```

오류 해결

역할 신뢰 정책을 제외한 모든 정책에는 `Resource` 또는 `NotResource` 요소가 포함되어야 합니다.

관련 용어

- 리소스 요소
- NotResource 요소
- 자격 증명 기반 정책 및 리소스 기반 정책
- JSON 정책 개요

오류 - 누락된 설명

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing statement: Add a statement to the policy
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Add a statement to the policy"
```

오류 해결

JSON 정책에는 설명이 포함되어야 합니다.

관련 용어

- JSON 정책 요소

오류 - if exists의 Null

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Null with if exists: The Null condition operator cannot be used with the IfExists
 suffix. Update the operator or the suffix.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The Null condition operator cannot be used with the IfExists suffix.
 Update the operator or the suffix."
```

오류 해결

`Null` 조건 연산자를 제외하고 조건 연산자 이름 끝에 `IfExists`를 추가할 수 있습니다. `Null` 조건 연산자를 사용하여 권한을 부여하는 시점에 조건 키의 유무를 검사할 수 있습니다. `...ifExists`를 사용하는 경우 "요청 컨텍스트에 정책 키가 있으면 정책에 지정된 대로 키를 처리하고, 키가 없으면 조건 요소를 true로 평가합니다."

관련 용어

- ...IfExists 조건 연산자
- Null 조건 연산자
- 조건 요소

오류 - SCP 구문 오류 작업 와일드카드

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
SCP syntax error action wildcard: SCP actions can include wildcards (*) only at the end
 of a string. Update {{action}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "SCP actions can include wildcards (*) only at the end of a string.
 Update {{action}}."
```

오류 해결

AWS Organizations 서비스 제어 정책(SCP)에서는 `Action` 또는 `NotAction` 요소의 값을 지정할 수 있습니다. 그러나 이러한 값에서는 문자열 끝에만 와일드카드(*)를 포함할 수 있습니다. 즉, `iam:Get*`는 지정할 수 있지만 `iam:*role`은 지정할 수 없습니다.

여러 작업을 지정하려면 AWS에서는 개별적으로 나열할 것을 권장합니다.

관련 용어

- SCP Action 및 NotAction 요소
- SCP 평가
- AWS Organizations 서비스 제어 정책
- IAM JSON 정책 요소: Action

오류 - SCP 구문 오류 조건 허용

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
SCP syntax error allow condition: SCPs do not support the Condition element with effect
 Allow. Update the element Condition or the effect.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "SCPs do not support the Condition element with effect Allow. Update
 the element Condition or the effect."
```

오류 해결

AWS Organizations 서비스 제어 정책(SCP)에서는 `"Effect": "Deny"`를 사용하는 경우에만 `Condition` 요소의 값을 지정할 수 있습니다.

단일 작업만 허용하려면 `...NotEquals` 버전의 조건 연산자를 사용하여 지정하는 조건을 제외한 모든 항목에 대한 액세스를 거부할 수 있습니다. 그러면 연산자를 통해 수행된 비교를 부정합니다.

관련 용어

- SCP Condition 요소
- SCP 평가
- AWS Organizations 서비스 제어 정책
- 예제 정책: 요청된 리전에 따라 AWS에 대한 액세스 거부
- IAM JSON 정책 요소: 조건 연산자

- IAM JSON 정책 요소: Condition

오류 - SCP 구문 오류 NotAction 허용

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
SCP syntax error allow NotAction: SCPs do not support NotAction with effect Allow.
 Update the element NotAction or the effect.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "SCPs do not support NotAction with effect Allow. Update the element
 NotAction or the effect."
```

오류 해결

AWS Organizations 서비스 제어 정책(SCP)에서는 `"Effect": "Allow"`와 함께 `NotAction` 요소를 사용할 수 없습니다.

작업 목록을 허용하거나 나열되지 않은 모든 작업을 거부하도록 로직을 다시 작성해야 합니다.

관련 용어

- SCP Action 및 NotAction 요소
- SCP 평가
- AWS Organizations 서비스 제어 정책
- IAM JSON 정책 요소: Action

오류 - SCP 구문 오류 리소스 허용

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
SCP syntax error allow resource: SCPs do not support Resource with effect Allow. Update
 the element Resource or the effect.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "SCPs do not support Resource with effect Allow. Update the element
 Resource or the effect."
```

오류 해결

AWS Organizations 서비스 제어 정책(SCP)에서는 `"Effect": "Deny"`를 사용하는 경우에만 `Resource` 요소의 값을 지정할 수 있습니다.

모든 리소스를 허용하거나 나열된 모든 리소스를 거부하도록 로직을 다시 작성해야 합니다.

관련 용어

- SCP Resource 요소
- SCP 평가
- AWS Organizations 서비스 제어 정책
- IAM JSON 정책 요소: Resource

오류 - SCP 구문 오류 NotResource

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
SCP syntax error NotResource: SCPs do not support the NotResource element. Update the
 policy to use Resource instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "SCPs do not support the NotResource element. Update the policy to
 use Resource instead."
```

오류 해결

AWS Organizations 서비스 제어 정책(SCP)에서는 `NotResource` 요소를 지원하지 않습니다.

모든 리소스를 허용하거나 나열된 모든 리소스를 거부하도록 로직을 다시 작성해야 합니다.

관련 용어

- SCP Resource 요소
- SCP 평가
- AWS Organizations 서비스 제어 정책
- IAM JSON 정책 요소: Resource

오류 - SCP 구문 오류 보안 주체

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
SCP syntax error principal: SCPs do not support specifying principals. Remove the
 Principal or NotPrincipal element.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "SCPs do not support specifying principals. Remove the Principal or
 NotPrincipal element."
```

오류 해결

AWS Organizations 서비스 제어 정책(SCP)에서는 `Principal` 또는 `NotPrincipal` 요소를 지원하지 않습니다.

`Condition` 요소에 `aws:PrincipalArn` 전역 조건 키를 사용하여 Amazon 리소스 이름(ARN)을 지정할 수 있습니다.

관련 용어

- SCP 구문
- 보안 주체의 전역 조건 키

오류 - 고유 Sid 필요

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unique Sids required: Duplicate statement IDs are not supported for this policy type.
 Update the Sid value.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Duplicate statement IDs are not supported for this policy type.
 Update the Sid value."
```

오류 해결

일부 정책 유형의 경우 문 ID는 고유해야 합니다. `Sid`(문 ID) 요소를 사용하면 정책 문에 제공하는 선택적 식별자를 입력할 수 있습니다. `SID` 요소를 사용하여 문 배열의 각 문에 문 ID 값을 할당할 수 있습니다. SQS나 SNS처럼 ID 요소를 지정할 수 있는 서비스에서는 `Sid` 값이 정책 문서 ID의 하위 ID나 마찬가지입니다. 예를 들어 IAM에서 `Sid` 값은 JSON 정책 내에서 고유해야 합니다.

관련 용어

- IAM JSON 정책 요소: Sid

오류 – 정책에서 지원되지 않는 작업

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unsupported action in policy: The action {{action}} is not supported for the resource-
based policy attached to the resource type {{resourceType}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The action {{action}} is not supported for the resource-based policy
 attached to the resource type {{resourceType}}."
```

오류 해결

일부 작업은 다른 리소스 유형에 연결된 리소스 기반 정책의 `Action` 요소에서 지원되지 않습니다. 예를 들어 Amazon S3 버킷 정책에서는 AWS Key Management Service 작업이 지원되지 않습니다. 리소스 기반 정책에 연결된 리소스 유형에서 지원하는 작업을 지정합니다.

관련 용어

- JSON 정책 요소: Action(작업)

오류 - 지원되지 않는 요소 조합

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unsupported element combination: The policy elements ${element1} and ${element2} can
 not be used in the same statement. Remove one of these elements.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The policy elements ${element1} and ${element2} can not be used in
 the same statement. Remove one of these elements."
```

오류 해결

JSON 정책 요소의 일부 조합은 함께 사용할 수 없습니다. 예를 들어 동일한 정책 문에서 `Action`과 `NotAction` 둘 다 사용할 수 없습니다. 함께 사용할 수 없는 다른 페어에는 `Principal/NotPrincipal` 및 `Resource/NotResource`가 있습니다.

관련 용어

- IAM JSON 정책 요소 참조
- JSON 정책 개요

오류 - 지원되지 않는 전역 조건 키

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unsupported global condition key: The condition key aws:ARN is not supported. Use
 aws:PrincipalArn or aws:SourceArn instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The condition key aws:ARN is not supported. Use aws:PrincipalArn or
 aws:SourceArn instead."
```

오류 해결

AWS에서는 지정된 전역 조건 키 사용을 지원하지 않습니다. 사용 사례에 따라 `aws:PrincipalArn` 또는 `aws:SourceArn` 전역 조건 키를 사용할 수 있습니다. 예를 들어 `aws:ARN` 대신 `aws:PrincipalArn`을 사용하여 요청을 한 보안 주체의 Amazon 리소스 이름(ARN)을 정책에서 지정한 ARN과 비교합니다. 또는 `aws:SourceArn` 전역 조건 키를 사용하여 서비스 대 서비스 요청을 하는 리소스의 Amazon 리소스 이름(ARN)을 정책에서 지정한 ARN과 비교합니다.

관련 용어

- AWS 전역 조건 컨텍스트 키

오류 - 지원되지 않는 보안 주체

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unsupported principal: The policy type ${policy_type} does not support the Principal
 element. Remove the Principal element.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The policy type ${policy_type} does not support the Principal
 element. Remove the Principal element."
```

오류 해결

`Principal` 요소는 리소스에 대한 액세스가 허용되거나 거부되는 보안 주체를 지정합니다. IAM 자격 증명 기반 정책에서는 `Principal` 요소를 사용할 수 없습니다. IAM 역할을 위한 신뢰 정책 및 리소스 기반 정책에서는 사용할 수 있습니다. 리소스 기반 정책은 리소스에 직접 삽입할 수 있는 정책입니다. 예를 들어 Amazon S3 버킷 또는 AWS KMS 키에 정책을 포함할 수 있습니다.

관련 용어

- AWS JSON 정책 요소: Principal
- IAM의 크로스 계정 리소스 액세스

오류 – 정책에서 지원되지 않는 리소스 ARN

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unsupported resource ARN in policy: The resource ARN is not supported for the resource-
based policy attached to the resource type {{resourceType}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The resource ARN is not supported for the resource-based policy
 attached to the resource type {{resourceType}}."
```

오류 해결

일부 리소스 ARN은 정책이 다른 리소스 유형에 연결된 경우 리소스 기반 정책의 `Resource` 요소에서 지원되지 않습니다. 예를 들어 Amazon S3 버킷 정책에 대한 `Resource` 요소에서는 AWS KMS ARN이 지원되지 않습니다. 리소스 기반 정책에 연결된 리소스 유형에서 지원하는 리소스 ARN을 지정합니다.

관련 용어

- JSON 정책 요소: Action(작업)

오류 - 지원되지 않는 Sid

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unsupported Sid: Update the characters in the Sid element to use one of the following
 character types: [a-z, A-Z, 0-9]
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Update the characters in the Sid element to use one of the following
 character types: [a-z, A-Z, 0-9]"
```

오류 해결

`Sid` 요소는 대문자, 소문자 및 숫자를 지원합니다.

관련 용어

- IAM JSON 정책 요소: Sid

오류 - 보안 주체에서 지원되지 않는 와일드카드

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unsupported wildcard in principal: Wildcards (*, ?) are not supported with the
 principal key {{principal_key}}. Replace the wildcard with a valid principal value.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Wildcards (*, ?) are not supported with the principal key
 {{principal_key}}. Replace the wildcard with a valid principal value."
```

오류 해결

Principal 요소 구조는 키 값 페어 사용을 지원합니다. 정책에 지정된 보안 주체 값에는 와일드카드(*)가 포함됩니다. 지정한 보안 주체 키에는 와일드카드를 포함할 수 없습니다. 예를 들어 Principal 요소로 사용자를 지정할 때는 "모든 사용자"의 의미로 와일드카드를 사용할 수 없습니다. 특정 사용자 또는 사용자들의 이름을 지정해야 합니다. 마찬가지로 수임된 역할 세션을 지정할 때 “모든 세션”을 의미하는 와일드카드(*)를 사용할 수 없습니다. 특정 세션의 이름을 지정해야 합니다. 또한 이름이나 ARN의 일부와 일치하도록 하기 위해 와일드카드를 사용할 수 없습니다.

이 결과를 해결하려면 와일드카드를 제거하고 더 구체적인 보안 주체를 지정합니다.

관련 용어

- AWS JSON 정책 요소: Principal

오류 - 변수에 중괄호 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing brace in variable: The policy variable is missing a closing curly brace. Add }
 after the variable text.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The policy variable is missing a closing curly brace. Add } after
 the variable text."
```

오류 해결

정책 변수 구조에서는 $ 접두사와 그 뒤에 중괄호 페어({ })를 사용할 수 있습니다. ${ } 문자 안에는 정책에서 사용할 요청 값의 이름을 포함합니다.

이 결과를 해결하려면 열고 닫는 전체 중괄호 세트가 있도록 누락된 중괄호를 추가합니다.

관련 용어

- IAM 정책 요소: 변수

오류 - 변수에 따옴표 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing quote in variable: The policy variable default value must begin and end with a
 single quote. Add the missing quote.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The policy variable default value must begin and end with a single
 quote. Add the missing quote."
```

오류 해결

정책에 변수를 추가할 때 변수의 기본값을 지정할 수 있습니다. 변수가 없는 경우 AWS은(는) 제공한 기본 텍스트를 사용합니다.

변수에 기본값을 추가하려면 기본값을 작은따옴표(' ')를 사용하고 변수 텍스트와 기본값을 쉼표와 공백(,)으로 묶습니다.

예를 들어, 보안 주체가 team=yellow(으)로 태그가 지정된 경우, amzn-s3-demo-bucket-yellow(이)라는 이름의 amzn-s3-demo-bucket Amazon S3 버킷에 액세스할 수 있습니다. 이 리소스를 사용하는 정책을 사용하면 팀 구성원이 자신의 리소스에 액세스할 수 있지만 다른 팀의 리소스에는 액세스할 수 없습니다. 팀 태그가 없는 사용자의 경우 company-wide의 기본값을 설정할 수 있

습니다. 이러한 사용자는 팀 참여에 대한 지침과 같은 광범위한 정보를 볼 수 있는 amzn-s3-demo-bucket-company-wide 버킷에만 액세스할 수 있습니다.

```
"Resource":"arn:aws:s3:::amzn-s3-demo-bucket-${aws:PrincipalTag/team, 'company-wide'}"
```

관련 용어

- IAM 정책 요소: 변수

오류 - 변수에 지원되지 않는 공백

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unsupported space in variable: A space is not supported within the policy variable
 text. Remove the space.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "A space is not supported within the policy variable text. Remove the
 space."
```

오류 해결

정책 변수 구조에서는 $ 접두사와 그 뒤에 중괄호 페어({ })를 사용할 수 있습니다. ${ } 문자 안에는 정책에서 사용할 요청 값의 이름을 포함합니다. 기본 변수를 지정할 때 공백을 포함할 수 있지만 변수 이름에는 공백을 포함할 수는 없습니다.

관련 용어

- IAM 정책 요소: 변수

오류 - 빈 변수

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Empty variable: Empty policy variable. Remove the ${ } variable structure or provide a
 variable within the structure.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Empty policy variable. Remove the ${ } variable structure or provide
 a variable within the structure."
```

오류 해결

정책 변수 구조에서는 $ 접두사와 그 뒤에 중괄호 페어({ })를 사용할 수 있습니다. ${ } 문자 안에는 정책에서 사용할 요청 값의 이름을 포함합니다.

관련 용어

- IAM 정책 요소: 변수

오류 - 요소에서 지원되지 않는 변수

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Variable unsupported in element: Policy variables are supported in the Resource and
 Condition elements. Remove the policy variable {{variable}} from this element.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Policy variables are supported in the Resource and Condition
 elements. Remove the policy variable {{variable}} from this element."
```

오류 해결

정책 변수는 Resource 요소를 비롯해 Condition 요소의 문자열 비교에 사용할 수 있습니다.

관련 용어

- IAM 정책 요소: 변수

오류 - 버전에서 지원되지 않는 변수

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Variable unsupported in version: To include variables in your policy, use the policy
 version 2012-10-17 or later.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "To include variables in your policy, use the policy version
 2012-10-17 or later."
```

오류 해결

정책 변수를 사용하려면 `Version` 요소를 포함하고 해당 요소를 정책 변수를 지원하는 버전으로 설정해야 합니다. 변수는 버전 `2012-10-17`에서 도입되었습니다. 정책 언어의 조기 버전은 정책 변수를 지원하지 않기 때문입니다. `Version`을 `2012-10-17` 이상으로 설정하지 않으면 `${aws:username}`과 같은 변수가 정책에서 리터럴 문자열로 처리됩니다.

`Version` 정책 요소는 정책 버전과 다릅니다. `Version` 정책 요소는 정책 내에서 사용되며 정책 언어의 버전을 정의합니다. 정책 버전은 IAM에서 고객 관리형 정책을 변경할 때 생성됩니다. 변경된 정책은 기존 정책을 덮어쓰지 않습니다. 대신 IAM에서 관리형 정책의 새 버전을 생성합니다.

관련 용어

- IAM 정책 요소: 변수
- IAM JSON 정책 요소: Version

오류 - 프라이빗 IP 주소

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Private IP address: aws:SourceIp works only for public IP address ranges. The values
 for condition key aws:SourceIp include only private IP addresses and will not have the
 desired effect. Update the value to include only public IP addresses.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "aws:SourceIp works only for public IP address ranges. The values
 for condition key aws:SourceIp include only private IP addresses and will not have the
 desired effect. Update the value to include only public IP addresses."
```

오류 해결

전역 조건 키 aws:SourceIp는 퍼블릭 IP 주소 범위에만 적용됩니다. 정책에서 프라이빗 IP 주소만 허용하면 이 오류가 발생합니다. 이 경우 조건이 일치하지 않습니다.

- aws:SourceIp 전역 조건 키
- IAM JSON 정책 요소: Condition

오류 - 프라이빗 NotIpAddress

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Private NotIpAddress: The values for condition key aws:SourceIp include only private
 IP addresses and has no effect. aws:SourceIp works only for public IP address ranges.
 Update the value to include only public IP addresses.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The values for condition key aws:SourceIp include only private IP
 addresses and has no effect. aws:SourceIp works only for public IP address ranges.
 Update the value to include only public IP addresses."
```

오류 해결

전역 조건 키 aws:SourceIp는 퍼블릭 IP 주소 범위에만 적용됩니다. NotIpAddress 조건 연산자를 사용하고 프라이빗 IP 주소만 나열할 경우 이 오류가 발생합니다. 이 경우 조건이 항상 일치하고 적용되지 않습니다.

- aws:SourceIp 전역 조건 키
- IAM JSON 정책 요소: Condition

오류 - 정책 크기가 SCP 할당량 초과

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Policy size exceeds SCP quota: The {{policySize}} characters in the service control
 policy (SCP) exceed the {{policySizeQuota}} character maximum for SCPs. We recommend
 that you use multiple granular policies.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The {{policySize}} characters in the service control policy (SCP)
 exceed the {{policySizeQuota}} character maximum for SCPs. We recommend that you use
 multiple granular policies."
```

오류 해결

AWS Organizations 서비스 제어 정책(SCP)에서는 `Action` 또는 `NotAction` 요소의 값을 지정할 수 있습니다. 그러나 이러한 값에서는 문자열 끝에만 와일드카드(*)를 포함할 수 있습니다. 즉, `iam:Get*`는 지정할 수 있지만 `iam:*role`은 지정할 수 없습니다.

여러 작업을 지정하려면 AWS에서는 개별적으로 나열할 것을 권장합니다.

관련 용어

- AWS Organizations에 대한 할당량
- AWS Organizations 서비스 제어 정책

오류 - 잘못된 서비스 보안 주체 형식

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid service principal format: The service principal does not match the expected
 format. Use the format {{expectedFormat}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The service principal does not match the expected format. Use the
 format {{expectedFormat}}."
```

오류 해결

조건 키 값 페어의 값은 정의된 서비스 보안 주체 형식과 일치해야 합니다.

서비스 보안 주체는 서비스에 권한을 부여하는 데 사용되는 식별자입니다. `Principal` 요소에서 서비스 보안 주체를 지정하거나 일부 전역 조건 키와 서비스별 키의 값에 대한 값으로 지정할 수 있습니다. 서비스 보안 주체는 각 서비스에서 정의합니다.

서비스 보안 주체의 식별자에는 서비스 이름이 포함되어 있으며 일반적으로 모두 소문자로 된 다음 형식을 갖습니다.

service-name.amazonaws.com

일부 서비스별 키에서는 서비스 보안 주체에 다른 형식을 사용할 수 있습니다. 예를 들어 kms:ViaService 조건 키는 서비스 보안 주체에 대해 모두 소문자로 된 다음 형식이 필요합니다.

service-name.AWS_region.amazonaws.com

관련 용어

- 서비스 보안 주체
- AWS 전역 조건 키
- kms:ViaService 조건 키

오류 - 조건에 태그 키 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing tag key in condition: The condition key {{conditionKeyName}} must include a tag
 key to control access based on tags. Use the format {{conditionKeyName}}tag-key and
 specify a key name for tag-key.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The condition key {{conditionKeyName}} must include a tag key to
 control access based on tags. Use the format {{conditionKeyName}}tag-key and specify a
 key name for tag-key."
```

오류 해결

태그를 기반으로 액세스를 제어하려면 정책의 조건 요소에 태그 정보를 제공하세요.

예를 들어 AWS 리소스에 대한 액세스 제어하려면 aws:ResourceTag 조건 키를 포함합니다. 이 키에는 aws:ResourceTag/*tag-key* 형식이 필요합니다. 조건에서 태그 키 owner 및 태그 값 JaneDoe을 지정하려면 다음 형식을 사용합니다.

```
"Condition": {
```

```
    "StringEquals": {"aws:ResourceTag/owner": "JaneDoe"}
}
```

관련 용어

- 태그를 사용한 액세스 제어
- 조건
- 전역 조건 키
- AWS 서비스 조건 키

오류 – 잘못된 vpc 형식

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid vpc format: The VPC identifier in the condition key value is not valid. Use the
 prefix 'vpc-' followed by 8 or 17 alphanumeric characters.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The VPC identifier in the condition key value is not valid. Use the
 prefix 'vpc-' followed by 8 or 17 alphanumeric characters."
```

오류 해결

이 aws:SourceVpc 조건 키는 접두사 vpc- 뒤에 8자 또는 17자의 영숫자 문자를 사용해야 합니다 (예: vpc-11223344556677889 또는 vpc-12345678).

관련 용어

- AWS 전역 조건 키: aws:SourceVpc

오류 – 잘못된 vpce 형식

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid vpce format: The VPCE identifier in the condition key value is not valid.  Use
 the prefix 'vpce-' followed by 8 or 17 alphanumeric characters.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The VPCE identifier in the condition key value is not valid.  Use
 the prefix 'vpce-' followed by 8 or 17 alphanumeric characters."
```

오류 해결

이 aws:SourceVpce 조건 키는 접두사 vpce- 뒤에 8자 또는 17자의 영숫자 문자를 사용해야 합니다(예: vpce-11223344556677889 또는 vpce-12345678).

관련 용어

- AWS 전역 조건 키: aws:SourceVpce

오류 – 페더레이션 보안 주체가 지원되지 않음

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Federated principal not supported: The policy type does not support a federated
 identity provider in the principal element. Use a supported principal.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The policy type does not support a federated identity provider in
 the principal element. Use a supported principal."
```

오류 해결

Principal 요소는 IAM 역할에 연결된 신뢰 정책에 페더레이션 보안 주체를 사용하여 ID 페더레이션을 통한 액세스를 제공합니다. 자격 증명 정책 및 기타 리소스 기반 정책은 Principal 요소에서 페더레이션 자격 증명 공급자를 지원하지 않습니다. 예를 들어 Amazon S3 버킷 정책에서는 SAML 보안 주체를 사용할 수 없습니다. Principal 요소를 지원되는 보안 주체 유형으로 변경합니다.

관련 용어

- ID 페더레이션의 역할 만들기
- JSON 정책 요소: Principal

오류 – 조건 키에 대해 지원되지 않는 작업

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unsupported action for condition key: The following actions: {{actions}} are not
 supported by the condition key {{key}}. The condition will not be evaluated for these
 actions. We recommend that you move these actions to a different statement without
 this condition key.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The following actions: {{actions}} are not supported by the
 condition key {{key}}. The condition will not be evaluated for these actions. We
 recommend that you move these actions to a different statement without this condition
 key."
```

오류 해결

정책 설명문의 `Condition` 요소에 있는 조건 키가 `Action` 요소의 모든 작업에 적용되는지 확인해야 합니다. 지정한 작업이 정책에 의해 효과적으로 허용되거나 거부되도록 하려면 지원되지 않는 작업을 조건 키 없는 다른 설명문으로 이동해야 합니다.

> **Note**
>
> `Action` 요소에 와일드카드가 있는 경우 IAM Access Analyzer는 이 오류에 대해 이러한 작업을 평가하지 않습니다.

관련 용어

- JSON 정책 요소: Action(작업)

오류 – 정책에서 지원되지 않는 작업

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unsupported action in policy: The action {{action}} is not supported for the resource-
based policy attached to the resource type {{resourceType}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The action {{action}} is not supported for the resource-based policy
 attached to the resource type {{resourceType}}."
```

오류 해결

일부 작업은 다른 리소스 유형에 연결된 리소스 기반 정책의 `Action` 요소에서 지원되지 않습니다. 예를 들어 Amazon S3 버킷 정책에서는 AWS Key Management Service 작업이 지원되지 않습니다. 리소스 기반 정책에 연결된 리소스 유형에서 지원하는 작업을 지정합니다.

관련 용어

- JSON 정책 요소: Action(작업)

오류 – 정책에서 지원되지 않는 리소스 ARN

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unsupported resource ARN in policy: The resource ARN is not supported for the resource-
based policy attached to the resource type {{resourceType}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The resource ARN is not supported for the resource-based policy
 attached to the resource type {{resourceType}}."
```

오류 해결

일부 리소스 ARN은 정책이 다른 리소스 유형에 연결된 경우 리소스 기반 정책의 `Resource` 요소에서 지원되지 않습니다. 예를 들어 Amazon S3 버킷 정책에 대한 `Resource` 요소에서는 AWS KMS ARN이 지원되지 않습니다. 리소스 기반 정책에 연결된 리소스 유형에서 지원하는 리소스 ARN을 지정합니다.

관련 용어

- JSON 정책 요소: Action(작업)

오류 – 서비스 보안 주체에 대해 지원되지 않는 조건 키

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unsupported condition key for service principal: The following condition keys are not
 supported when used with the service principal: {{conditionKeys}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The following condition keys are not supported when used with the
 service principal: {{conditionKeys}}."
```

오류 해결

서비스 보안 주체를 사용하는 리소스 기반 정책의 `Principal` 요소에서 AWS 서비스, 즉 서비스에 대한 식별자를 지정할 수 있습니다. 특정 서비스 보안 주체에는 일부 조건 키를 사용할 수 없습니다. 예를 들어, 서비스 보안 주체 `cloudfront.amazonaws.com`이(가) 있는 `aws:PrincipalOrgID` 조건 키를 사용할 수 없습니다. `Principal` 요소에서 서비스 보안 주체에 적용되지 않는 조건 키를 제거해야 합니다.

관련 용어

- 서비스 보안 주체
- JSON 정책 요소: Principal

오류 – 주체가 아닌 역할 신뢰 정책 구문 오류

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Role trust policy syntax error notprincipal: Role trust policies do not support
 NotPrincipal. Update the policy to use a Principal element instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Role trust policies do not support NotPrincipal. Update the policy
 to use a Principal element instead."
```

오류 해결

역할 신뢰 정책은 IAM 역할에 연결된 리소스 기반 정책입니다. 신뢰 정책은 역할을 수임할 수 있는 보안 주체 엔터티(계정, 사용자, 역할 및 페더레이션 사용자)를 정의합니다. 역할 신뢰 정책은 `NotPrincipal`을 지원하지 않습니다. 대신 `Principal` 요소를 사용하도록 정책을 업데이트하세요.

관련 용어

- JSON 정책 요소: Principal
- JSON 정책 요소: NotPrincipal

오류 – 주체에서 지원되지 않은 역할 신뢰 정책

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Role trust policy unsupported wildcard in principal: "Principal:" "*" is not supported
 in the principal element of a role trust policy. Replace the wildcard with a valid
 principal value.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": ""Principal:" "*" is not supported in the principal element of a role
 trust policy. Replace the wildcard with a valid principal value."
```

오류 해결

역할 신뢰 정책은 IAM 역할에 연결된 리소스 기반 정책입니다. 신뢰 정책은 역할을 수임할 수 있는 보안 주체 엔터티(계정, 사용자, 역할 및 페더레이션 사용자)를 정의합니다. `"Principal:" "*"`은 역할 신뢰 정책의 `Principal` 요소에서 지원되지 않습니다. 유용한 보안 주체 값으로 와일드카드를 바꿉니다.

관련 용어

- JSON 정책 요소: Principal

오류 – 역할 신뢰 정책 구문 오류 리소스

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Role trust policy syntax error resource: Role trust policies apply to the role
 that they are attached to. You cannot specify a resource. Remove the Resource or
 NotResource element.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Role trust policies apply to the role that they are attached to. You
 cannot specify a resource. Remove the Resource or NotResource element."
```

오류 해결

역할 신뢰 정책은 IAM 역할에 연결된 리소스 기반 정책입니다. 신뢰 정책은 역할을 수임할 수 있는 보안 주체 엔터티(계정, 사용자, 역할 및 페더레이션 사용자)를 정의합니다. 역할 신뢰 정책은 자신이 연결된 역할에 적용됩니다. 역할 신뢰 정책에서 `Resource` 또는 `NotResource` 요소를 지정할 수 없습니다. `Resource` 또는 `NotResource` 요소를 제거합니다.

- JSON 정책 요소: Resource
- JSON 정책 요소: NotResource

오류 - 형식 불일치 IP 범위

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Type mismatch IP range: The condition operator {{operator}} is used with an invalid IP
 range value. Specify the IP range in standard CIDR format.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The condition operator {{operator}} is used with an invalid IP range
 value. Specify the IP range in standard CIDR format."
```

오류 해결

CIDR 형식의 IP 주소 조건 연산자 데이터 형식을 사용하도록 텍스트를 업데이트합니다.

관련 용어

- IP 주소 조건 연산자
- IAM JSON 정책 요소: 조건 연산자

오류 – 조건 키에 대한 작업 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing action for condition key: The {{actionName}} action must be in the action block
 to allow setting values for the condition key {{keyName}}. Add {{actionName}} to the
 action block.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The {{actionName}} action must be in the action block to allow
 setting values for the condition key {{keyName}}. Add {{actionName}} to the action
 block."
```

오류 해결

정책 문의 `Condition` 요소에 있는 조건 키는 지정된 작업이 `Action` 요소에 있지 않는 한 평가되지 않습니다. 지정한 조건 키가 정책에서 효과적으로 허용/거부되도록 하려면 작업을 `Action` 요소에 추가합니다.

관련 용어

- JSON 정책 요소: Action(작업)

오류 – 역할 신뢰 정책에 잘못된 연동 보안 주체 구문

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid federated principal syntax in role trust policy: The principal value specifies
 a federated principal that does not match the expected format. Update the federated
 principal to a domain name or a SAML metadata ARN.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The principal value specifies a federated principal that does not
 match the expected format. Update the federated principal to a domain name or a SAML
 metadata ARN."
```

오류 해결

보안 주체 값은 예상 형식과 일치하지 않는 연동된 보안 주체를 지정합니다. 연동된 보안 주체의 형식을 유효한 도메인 이름이나 SAML 메타데이터 ARL으로 업데이트합니다.

관련 용어

- 연동 사용자 및 역할

오류 – 보안 주체에 대한 작업이 일치하지 않음

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Mismatched action for principal: The {{actionName}} action is invalid with the
 following principal(s): {{principalNames}}. Use a SAML provider principal with
 the sts:AssumeRoleWithSAML action or use an OIDC provider principal with the
 sts:AssumeRoleWithWebIdentity action. Ensure the provider is Federated if you use
 either of the two options.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The {{actionName}} action is invalid with the following
 principal(s): {{principalNames}}. Use a SAML provider principal with the
 sts:AssumeRoleWithSAML action or use an OIDC provider principal with the
 sts:AssumeRoleWithWebIdentity action. Ensure the provider is Federated if you use
 either of the two options."
```

오류 해결

정책 문의 `Action` 요소에서 지정된 작업은 `Principal` 요소에 지정된 보안 주체에 유효하지 않습니다. 예를 들어 `sts:AssumeRoleWithWebIdentity` 작업과 SAML 공급자 보안 주체를 사용할 수 없습니다. `sts:AssumeRoleWithSAML` 작업과 함께 SAML 공급자 보안 주체를 사용하거나 `sts:AssumeRoleWithWebIdentity` 작업과 함께 OIDC 공급자 보안 주체를 사용해야 합니다.

관련 용어

- AssumeRoleWithSAML
- AssumeRoleWithWebIdentity

오류 – 신뢰 정책이 있는 역할에 대한 작업 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing action for roles anywhere trust policy: The rolesanywhere.amazonaws.com service
 principal requires the sts:AssumeRole, sts:SetSourceIdentity, and sts:TagSession
 permissions to assume a role. Add the missing permissions to the policy.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The rolesanywhere.amazonaws.com service principal requires the
 sts:AssumeRole, sts:SetSourceIdentity, and sts:TagSession permissions to assume a
 role. Add the missing permissions to the policy."
```

오류 해결

IAM Roles Anywhere가 역할을 수입하고 임시 AWS 자격 증명을 제공할 수 있으려면 역할에서 IAM Roles Anywhere 서비스 보안 주체를 신뢰해야 합니다. IAM Roles Anywhere 서비스 보안 주체는 역할을 수임하기 위한 `sts:AssumeRole`, `sts:SetSourceIdentity`, `sts:TagSession` 권한이 필요합니다. 이 권한 중 하나라도 누락되면 정책에 추가해야 합니다.

관련 용어

- AWS Identity and Access Management Roles Anywhere의 신뢰 모델

일반 경고 - NotResource를 사용하여 SLR 생성

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Create SLR with NotResource: Using the iam:CreateServiceLinkedRole action with
 NotResource can allow creation of unintended service-linked roles for multiple
 resources. We recommend that you specify resource ARNs instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using the iam:CreateServiceLinkedRole action with NotResource can
 allow creation of unintended service-linked roles for multiple resources. We recommend
 that you specify resource ARNs instead."
```

일반 경고 해결

`iam:CreateServiceLinkedRole` 작업은 AWS 서비스가 대신 작업을 수행할 수 있도록 허용하는 IAM 역할을 생성할 권한을 부여합니다. 정책에서 `iam:CreateServiceLinkedRole`을 `NotResource` 요소와 함께 사용하면 여러 리소스에 대해 의도하지 않은 서비스 연결 역할 생성을 허용할 수 있습니다. AWS에서는 대신 `Resource` 요소에 허용되는 ARN을 지정할 것을 권장합니다.

- CreateServiceLinkedRole 작업
- IAM JSON 정책 요소: NotResource
- IAM JSON 정책 요소: Resource

일반 경고 - Action에 별표와 NotResource를 사용하여 SLR 생성

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Create SLR with star in action and NotResource: Using an action with a wildcard(*) and
 NotResource can allow creation of unintended service-linked roles because it can allow
 iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you
 specify resource ARNs instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using an action with a wildcard(*) and NotResource can
 allow creation of unintended service-linked roles because it can allow
 iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you
 specify resource ARNs instead."
```

일반 경고 해결

`iam:CreateServiceLinkedRole` 작업은 AWS 서비스가 대신 작업을 수행할 수 있도록 허용하는 IAM 역할을 생성할 권한을 부여합니다. 정책에서 `Action`에 와일드카드(*)를 사용하고 `NotResource` 요소를 포함하면 여러 리소스에 대해 의도하지 않은 서비스 연결 역할 생성을 허용할 수 있습니다. AWS에서는 대신 `Resource` 요소에 허용되는 ARN을 지정할 것을 권장합니다.

- CreateServiceLinkedRole 작업
- IAM JSON 정책 요소: NotResource
- IAM JSON 정책 요소: Resource

일반 경고 - NotAction 및 NotResource를 사용하여 SLR 생성

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Create SLR with NotAction and NotResource: Using NotAction with NotResource
 can allow creation of unintended service-linked roles because it allows
 iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you
 specify resource ARNs instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using NotAction with NotResource can allow creation of unintended
 service-linked roles because it allows iam:CreateServiceLinkedRole permissions on
 multiple resources. We recommend that you specify resource ARNs instead."
```

일반 경고 해결

`iam:CreateServiceLinkedRole` 작업은 AWS 서비스가 대신 작업을 수행할 수 있도록 허용하는 IAM 역할을 생성할 권한을 부여합니다. `NotAction` 요소를 `NotResource` 요소와 함께 사용하면 여러 리소스에 대해 의도하지 않은 서비스 연결 역할 생성을 허용할 수 있습니다. AWS에서는 대신 `Resource` 요소에 제한된 ARN 목록에서 `iam:CreateServiceLinkedRole`을 허용하도록 정책을 다시 작성할 것을 권장합니다. `iam:CreateServiceLinkedRole`을 `NotAction` 요소에 추가할 수도 있습니다.

- CreateServiceLinkedRole 작업
- IAM JSON 정책 요소: NotAction
- IAM JSON 정책 요소: Action
- IAM JSON 정책 요소: NotResource
- IAM JSON 정책 요소: Resource

일반 경고 - Resource에 별표를 사용하여 SLR 생성

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Create SLR with star in resource: Using the iam:CreateServiceLinkedRole action with
 wildcards (*) in the resource can allow creation of unintended service-linked roles.
 We recommend that you specify resource ARNs instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using the iam:CreateServiceLinkedRole action with wildcards (*) in
 the resource can allow creation of unintended service-linked roles. We recommend that
 you specify resource ARNs instead."
```

일반 경고 해결

`iam:CreateServiceLinkedRole` 작업은 AWS 서비스가 대신 작업을 수행할 수 있도록 허용하는 IAM 역할을 생성할 권한을 부여합니다. 정책에서 `iam:CreateServiceLinkedRole`을 사용할 때 `Resource` 요소에 와일드카드(*)를 포함하면 여러 리소스에 대해 의도하지 않은 서비스 연결 역할 생성을 허용할 수 있습니다. AWS에서는 대신 `Resource` 요소에 허용되는 ARN을 지정할 것을 권장합니다.

- CreateServiceLinkedRole 작업
- IAM JSON 정책 요소: Resource

이 일반 경고가 포함된 AWS 관리형 정책

AWS 관리형 정책을 사용하면 일반 AWS 사용 사례에 따라 권한을 할당하여 AWS를 시작할 수 있습니다.

이러한 사용 사례 중 일부는 계정 내의 고급 사용자를 위한 것입니다. 다음 AWS 관리형 정책은 고급 사용자 액세스 권한을 제공하고 모든 AWS 서비스를 위한 서비스 연결 역할을 생성할 수 있는 권한을 부여합니다. AWS에서는 다음 AWS 관리형 정책을 고급 사용자로 간주하는 IAM 자격 증명에만 연결할 것을 권장합니다.

- PowerUserAccess
- AlexaForBusinessFullAccess
- AWSOrganizationsServiceTrustPolicy – 이 AWS 관리형 정책은 AWS Organizations 서비스 연결 역할에서 사용하기 위한 권한을 제공합니다. 이 역할을 사용하여 Organizations에서 AWS 조직의 다른 서비스를 위한 추가 서비스 연결 역할을 생성할 수 있습니다.

일반 경고 - Action 및 Resource에 별표를 사용하여 SLR 생성

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Create SLR with star in action and resource: Using wildcards (*) in the action and
 the resource can allow creation of unintended service-linked roles because it allows
 iam:CreateServiceLinkedRole permissions on all resources. We recommend that you
 specify resource ARNs instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using wildcards (*) in the action and the resource
 can allow creation of unintended service-linked roles because it allows
 iam:CreateServiceLinkedRole permissions on all resources. We recommend that you
 specify resource ARNs instead."
```

일반 경고 해결

`iam:CreateServiceLinkedRole` 작업은 AWS 서비스가 대신 작업을 수행할 수 있도록 허용하는 IAM 역할을 생성할 권한을 부여합니다. 정책에서 `Action` 및 `Resource`요소에 와일드카드(*)를 사용하면 여러 리소스에 대해 의도하지 않은 서비스 연결 생성을 허용할 수 있습니다. 이 경우 `"Action": "*"`, `"Action": "iam:*"` 또는 `"Action": "iam:Create*"`를 지정하여 서비스 연결 역할을 생성할 수 있습니다. AWS에서는 대신 `Resource` 요소에 허용되는 ARN을 지정할 것을 권장합니다.

- CreateServiceLinkedRole 작업
- IAM JSON 정책 요소: Action
- IAM JSON 정책 요소: Resource

이 일반 경고가 포함된 AWS 관리형 정책

AWS 관리형 정책을 사용하면 일반 AWS 사용 사례에 따라 권한을 할당하여 AWS를 시작할 수 있습니다.

이러한 사용 사례 중 일부는 계정 내의 관리자를 위한 것입니다. 다음 AWS 관리형 정책은 관리자 액세스 권한을 제공하고 모든 AWS 서비스를 위한 서비스 연결 역할을 생성할 수 있는 권한을 부여합니다. AWS에서는 다음 AWS 관리형 정책을 관리자로 간주하는 IAM 자격 증명에만 연결할 것을 권장합니다.

- AdministratorAccess
- IAMFullAccess

일반 경고 - Resource에 별표와 NotAction을 사용하여 SLR 생성

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Create SLR with star in resource and NotAction: Using a resource with wildcards (*)
 and NotAction can allow creation of unintended service-linked roles because it allows
 iam:CreateServiceLinkedRole permissions on all resources. We recommend that you
 specify resource ARNs instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using a resource with wildcards (*) and NotAction can allow creation
 of unintended service-linked roles because it allows iam:CreateServiceLinkedRole
 permissions on all resources. We recommend that you specify resource ARNs instead."
```

일반 경고 해결

`iam:CreateServiceLinkedRole` 작업은 AWS 서비스가 대신 작업을 수행할 수 있도록 허용하는 IAM 역할을 생성할 권한을 부여합니다. 정책에서 `NotAction` 요소를 사용할 때 `Resource` 요소에 와일드카드(*)를 포함하면 여러 리소스에 대해 의도하지 않은 서비스 연결 역할 생성을 허용할 수 있습니다. AWS에서는 대신 `Resource` 요소에 허용되는 ARN을 지정할 것을 권장합니다. `iam:CreateServiceLinkedRole`을 `NotAction` 요소에 추가할 수도 있습니다.

- CreateServiceLinkedRole 작업
- IAM JSON 정책 요소: NotAction
- IAM JSON 정책 요소: Action
- IAM JSON 정책 요소: Resource

일반 경고 - 사용되지 않는 전역 조건 키

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Deprecated global condition key: We recommend that you update aws:ARN to use the newer
 condition key aws:PrincipalArn.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "We recommend that you update aws:ARN to use the newer condition key
 aws:PrincipalArn."
```

일반 경고 해결

정책에 사용되지 않는 전역 조건 키가 포함되어 있습니다. 지원되는 전역 조건 키를 사용하도록 조건 키 값 페어의 조건 키를 업데이트합니다.

- 전역 조건 키

일반 경고 - 잘못된 날짜 값

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid date value: The date {{date}} might not resolve as expected. We recommend that
 you use the YYYY-MM-DD format.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The date {{date}} might not resolve as expected. We recommend that
 you use the YYYY-MM-DD format."
```

일반 경고 해결

Unix Epoch 시간은 1970년 1월 1일 이후 경과한 시점, 즉 음의 윤초를 나타냅니다. Epoch 시간이 예상하는 정확한 시간으로 확인되지 않을 수도 있습니다. AWS에서는 날짜 및 시간 형식으로 W3C 표준을 사용할 것을 권장합니다. 예를 들어 YYYY-MM-DD (1997-07-16)과 같이 전체 날짜를 지정하거나 YYYY-MM-DDThh:mm:ssTZD (1997-07-16T19:20:30+01:00)과 같이 초까지 시간을 추가할 수 있습니다.

- W3C 날짜 및 시간 형식
- IAM JSON 정책 요소: Version
- aws:CurrentTime 전역 조건 키

일반 경고 - 잘못된 역할 참조

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid role reference: The Principal element includes the IAM role ID {{roleid}}. We
 recommend that you use a role ARN instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The Principal element includes the IAM role ID {{roleid}}. We
 recommend that you use a role ARN instead."
```

일반 경고 해결

AWS에서는 IAM 역할의 보안 주체 ID 대신 Amazon 리소스 이름(ARN)을 지정할 것을 권장합니다. IAM에서 정책을 저장할 때 ARN을 기존 역할의 보안 주체 ID로 변환합니다. AWS에는 안전 예방 조치가 포함되어 있습니다. 누군가가 역할을 삭제하고 다시 생성하면 역할에 새 ID가 지정되고 정책이 새 역할의 ID와 일치하지 않게 됩니다.

- 보안 주체 지정: IAM 역할
- IAM ARN
- IAM 고유 ID

일반 경고 - 잘못된 사용자 참조

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Invalid user reference: The Principal element includes the IAM user ID {{userid}}. We
 recommend that you use a user ARN instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The Principal element includes the IAM user ID {{userid}}. We
 recommend that you use a user ARN instead."
```

일반 경고 해결

AWS에서는 IAM 사용자의 보안 주체 ID 대신 Amazon 리소스 이름(ARN)을 지정할 것을 권장합니다. IAM에서 정책을 저장할 때 ARN을 기존 사용자의 보안 주체 ID로 변환합니다. AWS에는 안전 예방 조치가 포함되어 있습니다. 누군가가 사용자를 삭제하고 다시 생성하면 사용자에 새 ID가 지정되고 정책이 새 사용자의 ID와 일치하지 않게 됩니다.

- 보안 주체 지정: IAM 사용자
- IAM ARN
- IAM 고유 ID

일반 경고 - 버전 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing version: We recommend that you specify the Version element to help you with
 debugging permission issues.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "We recommend that you specify the Version element to help you with
 debugging permission issues."
```

일반 경고 해결

AWS에서는 정책에 선택적 `Version` 파라미터를 포함할 것을 권장합니다. Version 요소를 포함하지 않을 경우의 기본값은 `2012-10-17`이지만 정책 변수 등 최신 기능을 정책에서 사용할 수 없습니다. 예를 들어 `${aws:username}` 같은 변수가 정책에서 변수로 인식되지 않고 리터럴 문자열로 취급됩니다.

- IAM JSON 정책 요소: Version

일반 경고 - 고유한 Sid 권장

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Unique Sids recommended: We recommend that you use statement IDs that are unique to
 your policy. Update the Sid value.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "We recommend that you use statement IDs that are unique to your
 policy. Update the Sid value."
```

일반 경고 해결

AWS에서는 고유한 문 ID를 사용할 것을 권장합니다. `Sid`(문 ID) 요소를 사용하면 정책 문에 제공하는 선택적 식별자를 입력할 수 있습니다. `SID` 요소를 사용하여 문 배열의 각 문에 문 ID 값을 할당할 수 있습니다.

관련 용어

- IAM JSON 정책 요소: Sid

일반 경고 - Like 연산자가 없는 와일드카드

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Wildcard without like operator: Your condition value includes a * or ? character. If
 you meant to use a wildcard (*, ?), update the condition operator to include Like.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Your condition value includes a * or ? character. If you meant to
 use a wildcard (*, ?), update the condition operator to include Like."
```

일반 경고 해결

`Condition` 요소 구조에서는 조건 연산자와 키 값 페어를 사용해야 합니다. 와일드카드(*, ?)를 사용하는 조건 값을 지정하는 경우 조건 연산자의 `Like` 버전을 사용해야 합니다. 예를 들어 `StringEquals` 문자열 조건 연산자 대신 `StringLike`를 사용합니다.

```
"Condition": {"StringLike": {"aws:PrincipalTag/job-category": "admin-*"}}
```

- IAM JSON 정책 요소: 조건 연산자
- IAM JSON 정책 요소: Condition

이 일반 경고가 포함된 AWS 관리형 정책

AWS 관리형 정책을 사용하면 일반 AWS 사용 사례에 따라 권한을 할당하여 AWS를 시작할 수 있습니다.

다음 AWS 관리형 정책은 패턴 일치를 위한 Like를 포함하는 조건 연산자 없이 조건 값에 와일드카드를 포함합니다. AWS 관리형 정책을 참조로 사용하여 고객 관리형 정책을 생성하는 경우 AWS에서는 StringLike와 같이 와일드카드(*, ?)를 사용한 패턴 일치를 지원하는 조건 연산자를 사용할 것을 권장합니다.

- AWSGlueConsoleSageMakerNotebookFullAccess

일반 경고 - 정책 크기가 자격 증명 정책 할당량 초과

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Policy size exceeds identity policy quota: The {{policySize}} characters in the
 identity policy, excluding whitespace, exceed the {{policySizeQuota}} character
 maximum for inline and managed policies. We recommend that you use multiple granular
 policies.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The {{policySize}} characters in the identity policy, excluding
 whitespace, exceed the {{policySizeQuota}} character maximum for inline and managed
 policies. We recommend that you use multiple granular policies."
```

일반 경고 해결

IAM 자격 증명(사용자, 사용자 그룹 또는 역할)에 최대 10개의 관리형 정책을 연결할 수 있습니다. 하지만 각 관리형 정책의 크기는 기본 할당량인 6,144자를 초과할 수 없습니다. IAM은 이 할당량을 기준으로 정책의 크기를 계산할 때 공백을 계산하지 않습니다. AWS에서 제한이라고도 하는 할당량은 AWS 계정의 리소스, 작업, 항목의 최댓값입니다.

또한 원하는 만큼의 인라인 정책을 IAM 자격 증명에 추가할 수 있습니다. 그러나 자격 증명당 모든 인라인 정책의 크기 합계는 지정된 할당량을 초과할 수 없습니다.

정책이 할당량보다 큰 경우 정책을 여러 문으로 구성하고 문을 여러 정책으로 그룹화할 수 있습니다.

관련 용어

- IAM 및 AWS STS 문자 할당량
- 복수의 문 및 복수의 정책
- IAM 고객 관리형 정책

- JSON 정책 개요
- IAM JSON 정책 문법

이 일반 경고가 포함된 AWS 관리형 정책

AWS 관리형 정책을 사용하면 일반 AWS 사용 사례에 따라 권한을 할당하여 AWS를 시작할 수 있습니다.

다음 AWS 관리형 정책은 여러 AWS 서비스 대상의 작업에 대한 권한을 부여하고 최대 정책 크기를 초과합니다. AWS 관리형 정책을 참조로 사용하여 고객 관리형 정책을 생성하는 경우 해당 정책을 여러 정책으로 분할해야 합니다.

- ReadOnlyAccess
- AWSSupportServiceRolePolicy

일반 경고 - 정책 크기가 리소스 정책 할당량 초과

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Policy size exceeds resource policy quota: The {{policySize}} characters in the
 resource policy exceed the {{policySizeQuota}} character maximum for resource
 policies. We recommend that you use multiple granular policies.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The {{policySize}} characters in the resource policy exceed the
 {{policySizeQuota}} character maximum for resource policies. We recommend that you use
 multiple granular policies."
```

일반 경고 해결

리소스 기반 정책은 Amazon S3 버킷과 같은 리소스에 연결하는 JSON 정책 문서입니다. 이러한 정책은 지정된 보안 주체에 해당 리소스에 대한 특정 작업을 수행할 수 있는 권한을 부여하고 이러한 권한이 적용되는 조건을 정의합니다. 리소스 기반 정책의 크기는 해당 리소스에 대해 설정된 할당량을 초과할 수 없습니다. AWS에서 제한이라고도 하는 할당량은 AWS 계정의 리소스, 작업, 항목의 최댓값입니다.

정책이 할당량보다 큰 경우 정책을 여러 문으로 구성하고 문을 여러 정책으로 그룹화할 수 있습니다.

관련 용어

- 리소스 기반 정책
- Amazon S3 버킷 정책
- 복수의 문 및 복수의 정책
- JSON 정책 개요
- IAM JSON 정책 문법

일반 경고 - 형식 불일치

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Type mismatch: Use the operator type {{allowed}} instead of operator {{operator}} for
 the condition key {{key}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Use the operator type {{allowed}} instead of operator {{operator}}
 for the condition key {{key}}."
```

일반 경고 해결

지원되는 조건 연산자 데이터 형식을 사용하도록 텍스트를 업데이트합니다.

예를 들어 `aws:MultiFactorAuthPresent` 전역 조건 키에는 `Boolean` 데이터 형식의 조건 연산자가 필요합니다. 날짜 또는 정수를 지정하면 데이터 형식이 일치하지 않습니다.

관련 용어

- 전역 조건 키
- IAM JSON 정책 요소: 조건 연산자

일반 경고 - 형식 불일치 부울

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Type mismatch Boolean: Add a valid Boolean value (true or false) for the condition
 operator {{operator}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Add a valid Boolean value (true or false) for the condition operator
 {{operator}}."
```

일반 경고 해결

`true` 또는 `false`와 같은 부울 조건 연산자 데이터 형식을 사용하도록 텍스트를 업데이트합니다.

예를 들어 `aws:MultiFactorAuthPresent` 전역 조건 키에는 `Boolean` 데이터 형식의 조건 연산자가 필요합니다. 날짜 또는 정수를 지정하면 데이터 형식이 일치하지 않습니다.

관련 용어

- 부울 조건 연산자
- IAM JSON 정책 요소: 조건 연산자

일반 경고 - 형식 불일치 날짜

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Type mismatch date: The date condition operator is used with an invalid value. Specify
 a valid date using YYYY-MM-DD or other ISO 8601 date/time format.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The date condition operator is used with an invalid value. Specify a
 valid date using YYYY-MM-DD or other ISO 8601 date/time format."
```

일반 경고 해결

`YYYY-MM-DD` 또는 다른 ISO 8601 날짜 시간 형식의 날짜 조건 연산자 데이터 형식을 사용하도록 텍스트를 업데이트합니다.

관련 용어

- 날짜 조건 연산자
- IAM JSON 정책 요소: 조건 연산자

일반 경고 - 형식 불일치 숫자

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Type mismatch number: Add a valid numeric value for the condition operator
 {{operator}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Add a valid numeric value for the condition operator {{operator}}."
```

일반 경고 해결

숫자 조건 연산자 데이터 형식을 사용하도록 텍스트를 업데이트합니다.

관련 용어

- 숫자 조건 연산자
- IAM JSON 정책 요소: 조건 연산자

일반 경고 - 형식 불일치 문자열

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Type mismatch string: Add a valid base64-encoded string value for the condition
 operator {{operator}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Add a valid base64-encoded string value for the condition operator
 {{operator}}."
```

일반 경고 해결

문자열 조건 연산자 데이터 형식을 사용하도록 텍스트를 업데이트합니다.

관련 용어

- 문자열 조건 연산자
- IAM JSON 정책 요소: 조건 연산자

일반 경고 – 특정 github 리포지토리 및 지점 권장

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Specific github repo and branch recommended: Using a wildcard (*) in
 token.actions.githubusercontent.com:sub can allow requests from more sources than
 you intended. Specify the value of token.actions.githubusercontent.com:sub with the
 repository and branch name.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using a wildcard (*) in token.actions.githubusercontent.com:sub
 can allow requests from more sources than you intended. Specify the value of
 token.actions.githubusercontent.com:sub with the repository and branch name."
```

일반 경고 해결

GitHub를 OIDC IdP로 사용하는 경우 IAM IdP와 연결된 역할을 수임할 수 있는 엔터티를 제한하는 것이 좋습니다. 역할 신뢰 정책에 `Condition` 문을 포함하면 특정 GitHub 조직, 리포지토리 또는 분기로 역할을 제한할 수 있습니다. 조건 키 `token.actions.githubusercontent.com:sub`를 사용하여 액세스를 제한할 수 있습니다. 특정 리포지토리 또는 분기 세트로 조건을 제한하는 것이 좋습니다. `token.actions.githubusercontent.com:sub`에 와일드카드(*)를 사용할 경우, 자신이 제어할 수 없는 조직 또는 리포지토리의 GitHub 작업이 AWS 계정의 GitHub IAM IdP와 연결된 역할을 수임할 수 있습니다.

관련 용어

- GitHub OIDC ID 제공자의 역할 구성

일반 경고 – 정책 크기가 역할 신뢰 정책 할당량 초과

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Policy size exceeds role trust policy quota: The characters in the role trust policy,
 excluding whitespace, exceed the character maximum. We recommend that you request a
```

```
role trust policy length quota increase using Service Quotas and AWS Support Center.
If the quotas have already been increased, then you can ignore this warning.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The characters in the role trust policy, excluding whitespace,
 exceed the character maximum. We recommend that you request a role trust policy length
 quota increase using Service Quotas and AWS Support Center. If the quotas have already
 been increased, then you can ignore this warning."
```

일반 경고 해결

IAM 및 AWS STS는 역할 신뢰 정책의 크기를 제한하는 할당량이 있습니다. 공백을 제외한 역할 신뢰 정책의 문자가 문자 최댓값을 초과합니다. Service Quotas 및 AWS Support Center Console을 사용하여 역할 신뢰 정책 길이 할당량을 높이는 것이 좋습니다.

관련 용어

- IAM 및 AWS STS 할당량, 이름 요구 사항 및 문자 제한

보안 경고 - NotPrincipal을 사용하여 허용

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Allow with NotPrincipal: Using Allow with NotPrincipal can be overly permissive. We
 recommend that you use Principal instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using Allow with NotPrincipal can be overly permissive. We recommend
 that you use Principal instead."
```

보안 경고 해결

`"Effect": "Allow"`를 `NotPrincipal`과 함께 사용하면 자나치게 허용적일 수 있습니다. 예를 들어 익명 보안 주체에게 권한을 부여할 수 있습니다. AWS에서는 `Principal` 요소를 사용하여 액세스 권한이 필요한 보안 주체를 지정할 것을 권장합니다. 또는 광범위한 액세스 권한을 허용한 다음 `NotPrincipal` 요소를 `“Effect”: “Deny”`와 함께 사용하는 다른 문을 추가할 수 있습니다.

- AWS JSON 정책 요소: Principal
- AWS JSON 정책 요소: NotPrincipal

보안 경고 - 단일 값 키의 AllValues

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
ForAllValues with single valued key: Using ForAllValues qualifier with the single-
valued condition key {{key}} can be overly permissive. We recommend that you remove
 ForAllValues:.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using ForAllValues qualifier with the single-valued condition key
 {{key}} can be overly permissive. We recommend that you remove ForAllValues:."
```

보안 경고 해결

AWS에서는 `ForAllValues`를 다중 값 조건에만 사용할 것을 권장합니다. `ForAllValues` 집합 연산자는 요청 집합의 모든 멤버 값이 조건 키 집합의 하위 집합인지를 테스트합니다. 요청의 모든 키 값이 정책에 있는 하나 이상의 값과 일치하면 조건이 true를 반환합니다. 요청에 키가 없거나 키 값이 빈 문자열과 같은 null 데이터 세트로 확인되는 경우에도 true를 반환합니다.

조건이 단일 값을 지원하는지 아니면 다중 값을 지원하는지 알아보려면 서비스에 대한 작업, 리소스 및 조건 키를 검토하세요. `ArrayOf` 데이터 유형 접두사가 있는 조건 키는 다중 값 조건 키입니다. 예를 들어 Amazon SES는 단일 값(`String`)을 가진 키와 `ArrayOfString` 다중 값 데이터 유형을 지원합니다.

- 다중 값 컨텍스트 키

보안 경고 - NotResource를 사용하여 역할 전달

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Pass role with NotResource: Using the iam:PassRole action with NotResource can be
 overly permissive because it can allow iam:PassRole permissions on multiple resources.
 We recommend that you specify resource ARNs instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using the iam:PassRole action with NotResource can be overly
 permissive because it can allow iam:PassRole permissions on multiple resources. We
 recommend that you specify resource ARNs instead."
```

보안 경고 해결

다수의 AWS 서비스를 구성하려면 IAM 역할을 서비스에 전달해야 합니다. 이를 허용하려면 자격 증명(사용자, 사용자 그룹 또는 역할)에 `iam:PassRole` 권한을 부여해야 합니다. `NotResource` 요소가 있는 정책에 `iam:PassRole`을 사용하면 보안 주체가 의도한 것보다 많은 서비스 또는 기능에 액세스할 수 있습니다. AWS에서는 대신 `Resource` 요소에 허용되는 ARN을 지정할 것을 권장합니다. 또한 `iam:PassedToService` 조건 키를 사용하여 권한을 단일 서비스로 줄일 수 있습니다.

- 역할을 서비스로 전달
- iam:PassedToService
- IAM JSON 정책 요소: NotResource
- IAM JSON 정책 요소: Resource

보안 경고 - Action에 별표 및 NotResource를 사용하여 역할 전달

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Pass role with star in action and NotResource: Using an action with a wildcard (*) and
 NotResource can be overly permissive because it can allow iam:PassRole permissions on
 multiple resources. We recommend that you specify resource ARNs instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using an action with a wildcard (*) and NotResource can be overly
 permissive because it can allow iam:PassRole permissions on multiple resources. We
 recommend that you specify resource ARNs instead."
```

보안 경고 해결

다수의 AWS 서비스를 구성하려면 IAM 역할을 서비스에 전달해야 합니다. 이를 허용하려면 자격 증명(사용자, 사용자 그룹 또는 역할)에 `iam:PassRole` 권한을 부여해야 합니다. `NotResource` 요소를

포함하는 정책에서 Action에 와일드카드(*)를 사용하면 보안 주체가 의도한 것보다 많은 서비스 또는 기능에 액세스할 수 있습니다. AWS에서는 대신 Resource 요소에 허용되는 ARN을 지정할 것을 권장합니다. 또한 iam:PassedToService 조건 키를 사용하여 권한을 단일 서비스로 줄일 수 있습니다.

- 역할을 서비스로 전달
- iam:PassedToService
- IAM JSON 정책 요소: NotResource
- IAM JSON 정책 요소: Resource

보안 경고 - NotAction 및 NotResource를 사용하여 역할 전달

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Pass role with NotAction and NotResource: Using NotAction with NotResource can
 be overly permissive because it can allow iam:PassRole permissions on multiple
 resources.. We recommend that you specify resource ARNs instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using NotAction with NotResource can be overly permissive because
 it can allow iam:PassRole permissions on multiple resources.. We recommend that you
 specify resource ARNs instead."
```

보안 경고 해결

다수의 AWS 서비스를 구성하려면 IAM 역할을 서비스에 전달해야 합니다. 이를 허용하려면 자격 증명(사용자, 사용자 그룹 또는 역할)에 iam:PassRole 권한을 부여해야 합니다. NotAction 요소를 사용하여 NotResource 요소의 일부 리소스를 나열하면 보안 주체가 의도한 것보다 많은 서비스 또는 기능에 액세스할 수 있습니다. AWS에서는 대신 Resource 요소에 허용되는 ARN을 지정할 것을 권장합니다. 또한 iam:PassedToService 조건 키를 사용하여 권한을 단일 서비스로 줄일 수 있습니다.

- 역할을 서비스로 전달
- iam:PassedToService
- IAM JSON 정책 요소: NotAction
- IAM JSON 정책 요소: Action

- IAM JSON 정책 요소: NotResource
- IAM JSON 정책 요소: Resource

보안 경고 - Resource에 별표를 사용하여 역할 전달

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Pass role with star in resource: Using the iam:PassRole action with wildcards (*)
 in the resource can be overly permissive because it allows iam:PassRole permissions
 on multiple resources. We recommend that you specify resource ARNs or add the
 iam:PassedToService condition key to your statement.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using the iam:PassRole action with wildcards (*) in the resource can
 be overly permissive because it allows iam:PassRole permissions on multiple resources.
 We recommend that you specify resource ARNs or add the iam:PassedToService condition
 key to your statement."
```

보안 경고 해결

다수의 AWS 서비스를 구성하려면 IAM 역할을 서비스에 전달해야 합니다. 이를 허용하려면 자격 증명(사용자, 사용자 그룹 또는 역할)에 `iam:PassRole` 권한을 부여해야 합니다. `iam:PassRole`을 허용하는 정책에서 `Resource` 요소에 와일드카드(*)를 포함하면 보안 주체가 의도한 것보다 많은 서비스 또는 기능에 액세스할 수 있습니다. AWS에서는 대신 `Resource` 요소에 허용되는 ARN을 지정할 것을 권장합니다. 또한 `iam:PassedToService` 조건 키를 사용하여 권한을 단일 서비스로 줄일 수 있습니다.

일부 AWS 서비스는 해당 역할 이름에 서비스 네임스페이스를 포함합니다. 이 정책 검사는 정책을 분석하는 중 이러한 규칙을 고려하여 결과를 생성합니다. 예를 들어 다음 리소스 ARN은 결과를 생성하지 않을 수 있습니다.

```
arn:aws:iam::*:role/Service*
```

- 역할을 서비스로 전달
- iam:PassedToService
- IAM JSON 정책 요소: Resource

이 보안 경고가 포함된 AWS 관리형 정책

AWS 관리형 정책을 사용하면 일반 AWS 사용 사례에 따라 권한을 할당하여 AWS를 시작할 수 있습니다.

이러한 사용 사례 중 하나는 계정 내의 관리자를 위한 것입니다. 다음 AWS 관리형 정책은 관리자 액세스 권한을 제공하고 모든 서비스에 IAM 역할을 전달할 수 있는 권한을 부여합니다. AWS에서는 다음 AWS 관리형 정책을 관리자로 간주하는 IAM 자격 증명에만 연결할 것을 권장합니다.

- AdministratorAccess-Amplify

다음 AWS 관리형 정책은 리소스에 와일드카드(*)를 사용하는 `iam:PassRole`에 대한 권한을 포함하며 사용 중단 경로에 있습니다. 이러한 각 정책에 대해 권한 지침을 업데이트했습니다(예: 사용 사례를 지원하는 새 AWS 관리형 정책 추천). 이러한 정책의 대안을 확인하려면 각 서비스의 가이드를 참조하세요.

- AWSElasticBeanstalkFullAccess
- AWSElasticBeanstalkService
- AWSLambdaFullAccess
- AWSLambdaReadOnlyAccess
- AWSOpsWorksFullAccess
- AWSOps워크롤
- AWSDataPipelineRole
- AmazonDynamoDBFullAccesswithDataPipeline
- AmazonElasticMapReduceFullAccess
- AmazonDynamoDBFullAccesswithDataPipeline
- AmazonEC2ContainerServiceFullAccess

다음 AWS 관리형 정책은 AWS 서비스에서 사용자 대신 작업을 수행하도록 허용하는 서비스 연결 역할에만 권한을 제공합니다. 이러한 정책은 IAM 자격 증명에 연결할 수 없습니다.

- AWSServiceRoleForAmazonEKSNodegroup

보안 경고 - Action 및 Resource에 별표를 사용하여 역할 전달

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Pass role with star in action and resource: Using wildcards (*) in the action and the
 resource can be overly permissive because it allows iam:PassRole permissions on all
 resources. We recommend that you specify resource ARNs or add the iam:PassedToService
 condition key to your statement.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using wildcards (*) in the action and the resource can be overly
 permissive because it allows iam:PassRole permissions on all resources. We recommend
 that you specify resource ARNs or add the iam:PassedToService condition key to your
 statement."
```

보안 경고 해결

다수의 AWS 서비스를 구성하려면 IAM 역할을 서비스에 전달해야 합니다. 이를 허용하려면 자격 증명(사용자, 사용자 그룹 또는 역할)에 `iam:PassRole` 권한을 부여해야 합니다. 정책에서 `Action`에 와일드카드(*)를 사용하고 `Resource` 요소를 포함하면 보안 주체가 의도한 것보다 많은 서비스 또는 기능에 액세스할 수 있습니다. AWS에서는 대신 `Resource` 요소에 허용되는 ARN을 지정할 것을 권장합니다. 또한 `iam:PassedToService` 조건 키를 사용하여 권한을 단일 서비스로 줄일 수 있습니다.

- 역할을 서비스로 전달
- iam:PassedToService
- IAM JSON 정책 요소: Action
- IAM JSON 정책 요소: Resource

이 보안 경고가 포함된 AWS 관리형 정책

AWS 관리형 정책을 사용하면 일반 AWS 사용 사례에 따라 권한을 할당하여 AWS를 시작할 수 있습니다.

이러한 사용 사례 중 일부는 계정 내의 관리자를 위한 것입니다. 다음 AWS 관리형 정책은 관리자 액세스 권한을 제공하고 모든 AWS 서비스에 IAM 역할을 전달할 수 있는 권한을 부여합니다. AWS에서는 다음 AWS 관리형 정책을 관리자로 간주하는 IAM 자격 증명에만 연결할 것을 권장합니다.

- AdministratorAccess
- IAMFullAccess

보안 경고 - Resource에 별표 및 NotAction을 사용하여 역할 전달

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Pass role with star in resource and NotAction: Using a resource with wildcards (*) and
 NotAction can be overly permissive because it allows iam:PassRole permissions on all
 resources. We recommend that you specify resource ARNs or add the iam:PassedToService
 condition key to your statement.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using a resource with wildcards (*) and NotAction can be overly
 permissive because it allows iam:PassRole permissions on all resources. We recommend
 that you specify resource ARNs or add the iam:PassedToService condition key to your
 statement."
```

보안 경고 해결

다수의 AWS 서비스를 구성하려면 IAM 역할을 서비스에 전달해야 합니다. 이를 허용하려면 자격 증명(사용자, 사용자 그룹 또는 역할)에 `iam:PassRole` 권한을 부여해야 합니다. `Resource` 요소에 와일드카드(*)를 포함하는 정책에서 `NotAction` 요소를 사용하면 보안 주체가 의도한 것보다 많은 서비스 또는 기능에 액세스할 수 있습니다. AWS에서는 대신 `Resource` 요소에 허용되는 ARN을 지정할 것을 권장합니다. 또한 `iam:PassedToService` 조건 키를 사용하여 권한을 단일 서비스로 줄일 수 있습니다.

- 역할을 서비스로 전달
- iam:PassedToService
- IAM JSON 정책 요소: NotAction
- IAM JSON 정책 요소: Action
- IAM JSON 정책 요소: Resource

보안 경고 - 페어링된 조건 키 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing paired condition keys: Using the condition key {{conditionKeyName}}
 can be overly permissive without also using the following condition keys:
```

```
{{recommendedKeys}}. Condition keys like this one are more secure when paired with
a related key. We recommend that you add the related condition keys to the same
condition block.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using the condition key {{conditionKeyName}} can be overly
 permissive without also using the following condition keys: {{recommendedKeys}}.
 Condition keys like this one are more secure when paired with a related key. We
 recommend that you add the related condition keys to the same condition block."
```

보안 경고 해결

일부 조건 키는 다른 관련 조건 키와 페어링하면 더 안전합니다. AWS에서는 기존 조건 키와 동일한 조건 블록에 관련 조건 키를 포함할 것을 권장합니다. 이렇게 하면 정책을 통해 권한이 보다 안전하게 부여됩니다.

`aws:VpcSourceIp` 조건 키를 사용하여 요청이 이루어진 IP 주소와 정책에서 지정한 IP 주소를 비교하는 경우를 예로 들 수 있습니다. AWS에서는 관련 `aws:SourceVPC` 조건 키를 추가할 것을 권장합니다. 이 조건 키는 요청이 사용자가 정책에서 지정한 VPC를 통해 이루어졌는지 여부와 사용자가 지정한 IP 주소를 확인입니다.

관련 용어

- aws:VpcSourceIp 전역 조건 키
- aws:SourceVPC 전역 조건 키
- 전역 조건 키
- 조건 요소
- JSON 정책 개요

보안 경고 - 서비스에 대해 지원되지 않는 태그 조건 키를 사용하여 거부

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Deny with unsupported tag condition key for service: Using the effect Deny with the
 tag condition key {{conditionKeyName}} and actions for services with the following
 prefixes can be overly permissive: {{serviceNames}}. Actions for the listed services
```

```
are not denied by this statement. We recommend that you move these actions to a
different statement without this condition key.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using the effect Deny with the tag condition key
 {{conditionKeyName}} and actions for services with the following prefixes can be
 overly permissive: {{serviceNames}}. Actions for the listed services are not denied
 by this statement. We recommend that you move these actions to a different statement
 without this condition key."
```

보안 경고 해결

`"Effect": "Deny"`가 포함된 정책의 `Condition` 요소에 지원되지 않는 태그 조건 키를 사용하면 해당 서비스에 대해 조건이 무시되므로 지나치게 허용적일 수 있습니다. AWS에서는 조건 키를 지원하지 않는 서비스 작업을 제거하고 해당 작업을 위한 특정 리소스에 대한 액세스를 거부하도록 다른 문을 작성할 것을 권장합니다.

`aws:ResourceTag` 조건 키를 사용할 때 서비스 작업에서 해당 키를 지원하지 않는 경우 요청 컨텍스트에 키가 포함되지 않습니다. 이 경우 `Deny` 문의 조건은 항상 `false`를 반환하고 작업이 거부되지 않습니다. 리소스가 올바로 태깅된 경우에도 이 문제가 발생합니다.

서비스에서 `aws:ResourceTag` 조건 키를 지원하는 경우 태그를 사용하여 해당 서비스 리소스에 대한 액세스를 제어할 수 있습니다. 이를 속성 기반 액세스 제어(ABAC)라고 합니다. 이러한 키를 지원하지 않는 서비스에서는 리소스 기반 액세스 제어(RBAC)를 사용하여 리소스에 대한 액세스를 제어해야 합니다.

> **Note**
>
> 일부 서비스에서는 해당 리소스 및 작업의 하위 집합에 대해 `aws:ResourceTag` 조건 키 지원을 허용합니다. IAM Access Analyzer에서는 지원되지 않는 서비스 작업에 대한 결과를 반환합니다. 예를 들어 Amazon S3는 해당 리소스의 하위 집합에 대해 `aws:ResourceTag`를 지원합니다. Amazon S3에서 사용할 수 있는 리소스 유형 중 `aws:ResourceTag` 조건 키를 지원하는 유형을 모두 보려면 서비스 승인 참조의 Amazon S3에서 정의한 리소스 유형을 참조하세요.

예를 들어 키 값 페어 `status=Confidential`로 태깅된 특정 리소스를 태그 해제, 삭제하는 액세스를 거부하려는 경우를 가정합니다. 또한 AWS Lambda에서 리소스를 태깅하거나 태그 해제할 수 있지만 `aws:ResourceTag` 조건 키를 지원하지 않는다고 가정합니다. 이 태그가 있는 경우 AWS App

Mesh 및 AWS Backup에 대한 삭제 작업을 거부하려면 aws:ResourceTag 조건 키를 사용합니다. Lambda의 경우 "Confidential" 접두사를 포함하는 리소스 명명 규칙을 사용합니다. 그런 다음 해당 명명 규칙을 사용하는 리소스를 삭제하지 못하게 하는 별도의 문을 포함합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyDeleteSupported",
            "Effect": "Deny",
            "Action": [
                "appmesh:DeleteMesh",
                "backup:DeleteBackupPlan"
                ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/status": "Confidential"
                }
            }
        },
        {
            "Sid": "DenyDeleteUnsupported",
            "Effect": "Deny",
            "Action": "lambda:DeleteFunction",
            "Resource": "arn:aws:lambda:*:123456789012:function:status-Confidential*"
        }
    ]
}
```

Warning

이 결과의 해결 방법으로 조건 연산자의 ...IfExists 버전을 사용하지 마세요. 이는 “요청 컨텍스트에 키가 있고 값이 일치하면 작업을 거부합니다. 그렇지 않으면 작업을 거부합니다.”를 의미합니다. 앞의 예제에서 StringEqualsIfExists 연산자가 있는 DenyDeleteSupported 문에 lambda:DeleteFunction 작업을 포함하면 작업이 항상 거부됩니다. 해당 작업의 경우 컨텍스트에 키가 없으며 리소스가 태깅되어 있는지 여부에 관계없이 해당 리소스 유형을 삭제하려는 모든 시도는 거부됩니다.

관련 용어

- 전역 조건 키
- ABAC와 RBAC 비교
- IAM JSON 정책 요소: 조건 연산자
- 조건 요소
- JSON 정책 개요

보안 경고 - 서비스에 대해 지원되지 않는 태그 조건 키를 사용하여 NotAction 거부

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Deny NotAction with unsupported tag condition key for service: Using the effect Deny
 with NotAction and the tag condition key {{conditionKeyName}} can be overly permissive
 because some service actions are not denied by this statement. This is because the
 condition key doesn't apply to some service actions. We recommend that you use Action
 instead of NotAction.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using the effect Deny with NotAction and the tag condition key
 {{conditionKeyName}} can be overly permissive because some service actions are not
 denied by this statement. This is because the condition key doesn't apply to some
 service actions. We recommend that you use Action instead of NotAction."
```

보안 경고 해결

`NotAction` 요소 및 `"Effect": "Deny"`가 포함된 정책의 `Condition` 요소에 태그 조건 키를 사용하면 지나치게 허용적일 수 있습니다. 조건 키를 지원하지 않는 서비스 작업의 경우 조건이 무시됩니다. AWS에서는 작업 목록을 거부하도록 로직을 다시 작성할 것을 권장합니다.

`aws:ResourceTag` 조건 키를 `NotAction`과 함께 사용하는 경우 키를 지원하지 않는 모든 신규 또는 기존 서비스 작업이 거부되지 않습니다. AWS에서는 거부할 작업을 명시적으로 나열할 것을 권장합니다. IAM Access Analyzer에서는 `aws:ResourceTag` 조건 키를 지원하지 않는 나열된 작업에 대해 별도의 결과를 반환합니다. 자세한 내용은 보안 경고 - 서비스에 대해 지원되지 않는 태그 조건 키를 사용하여 거부 단원을 참조하십시오.

서비스에서 `aws:ResourceTag` 조건 키를 지원하는 경우 태그를 사용하여 해당 서비스 리소스에 대한 액세스를 제어할 수 있습니다. 이를 속성 기반 액세스 제어(ABAC)라고 합니다. 이러한 키를 지원하

지 않는 서비스에서는 리소스 기반 액세스 제어(RBAC)를 사용하여 리소스에 대한 액세스를 제어해야 합니다.

관련 용어

- 전역 조건 키
- ABAC와 RBAC 비교
- IAM JSON 정책 요소: 조건 연산자
- 조건 요소
- JSON 정책 개요

보안 경고 - 서비스 보안 주체에 대한 액세스 제한

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Restrict access to service principal: Granting access to a service principal without
 specifying a source is overly permissive. Use aws:SourceArn, aws:SourceAccount,
 aws:SourceOrgID, or aws:SourceOrgPaths condition key to grant fine-grained access.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Granting access to a service principal without specifying a source
 is overly permissive. Use aws:SourceArn, aws:SourceAccount, aws:SourceOrgID, or
 aws:SourceOrgPaths condition key to grant fine-grained access."
```

보안 경고 해결

서비스 보안 주체를 사용하는 리소스 기반 정책의 `Principal` 요소에서 AWS 서비스, 즉 서비스에 대한 식별자를 지정할 수 있습니다. 사용자를 대신하여 작동하도록 서비스 보안 주체에 액세스 권한을 부여한 경우, 액세스를 제한합니다. `aws:SourceArn`, `aws:SourceAccount`, `aws:SourceOrgID`, 또는 `aws:SourceOrgPaths` 조건 키를 사용하여 지나치게 허용 정책을 방지하여 특정 소스(예: 특정 리소스 ARN, AWS 계정, 조직 ID 또는 조직 경로)에 대한 액세스를 제한할 수 있습니다. 액세스를 제한하면 혼동된 대리자 문제라 불리는 보안 문제를 예방할 수 있습니다.

관련 용어

- AWS 서비스 보안 주체
- AWS 전역 조건 키: aws:SourceAccount

- AWS 전역 조건 키: aws:SourceArn
- AWS 전역 조건 키: aws:SourceOrgId
- AWS 전역 조건 키: aws:SourceOrgPaths
- 혼동된 대리자 문제

보안 경고 – OIDC 보안 주체에 대한 조건 키 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing condition key for oidc principal: Using an Open ID Connect principal without a
 condition can be overly permissive. Add condition keys with a prefix that matches your
 federated OIDC principals to ensure that only the intended identity provider assumes
 the role.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using an Open ID Connect principal without a condition can be
 overly permissive. Add condition keys with a prefix that matches your federated OIDC
 principals to ensure that only the intended identity provider assumes the role."
```

보안 경고 해결

조건 없이 Open ID Connect 보안 주체를 사용하면 과하게 권한이 커질 수 있습니다. 연동된 OIDC 부안 주체와 일치하는 접두사로 조건 키를 추가하고, 의도한 ID 공급자만 역할을 수임하도록 합니다.

관련 용어

- 웹 아이덴티티 또는 OpenID Connect 페더레이션을 위한 역할 생성(콘솔)

보안 경고 – github 리포지토리 조건 키 누락

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Missing github repo condition key: Granting a federated GitHub principal permissions
 without a condition key can allow more sources to assume the role than you intended.
 Add the token.actions.githubusercontent.com:sub condition key and specify the branch
 and repository name in the value.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Granting a federated GitHub principal permissions without a
 condition key can allow more sources to assume the role than you intended. Add the
 token.actions.githubusercontent.com:sub condition key and specify the branch and
 repository name in the value."
```

보안 경고 해결

GitHub를 OIDC IdP로 사용하는 경우 IAM IdP와 연결된 역할을 수임할 수 있는 엔터티를 제한하는 것이 좋습니다. 역할 신뢰 정책에 `Condition` 문을 포함하면 특정 GitHub 조직, 리포지토리 또는 분기로 역할을 제한할 수 있습니다. 조건 키 `token.actions.githubusercontent.com:sub`를 사용하여 액세스를 제한할 수 있습니다. 특정 리포지토리 또는 분기 세트로 조건을 제한하는 것이 좋습니다. 이 조건을 포함하지 않으면 제어할 수 없는 조직 또는 리포지토리의 GitHub 작업이 AWS 계정의 GitHub IAM IdP와 연결된 역할을 수임할 수 있습니다.

관련 용어

- GitHub OIDC ID 제공자의 역할 구성

제안 - 빈 배열 작업

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Empty array action: This statement includes no actions and does not affect the policy.
 Specify actions.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "This statement includes no actions and does not affect the policy.
 Specify actions."
```

제안 해결

문에는 작업 집합을 포함하는 `Action` 또는 `NotAction` 요소 중 하나를 포함해야 합니다. 요소가 비어 있으면 정책 문에서 권한을 제공하지 않습니다. `Action` 요소에 작업을 지정합니다.

- IAM JSON 정책 요소: Action

제안 - 빈 배열 조건

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Empty array condition: There are no values for the condition key {{key}} and it does
 not affect the policy. Specify conditions.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "There are no values for the condition key {{key}} and it does not
 affect the policy. Specify conditions."
```

제안 해결

선택적 `Condition` 요소 구조에서는 조건 연산자와 키 값 페어를 사용해야 합니다. 조건 값이 비어 있으면 조건은 `true`를 반환하고 정책 문에서 권한을 제공하지 않습니다. 조건 값을 지정합니다.

- IAM JSON 정책 요소: Condition

제안 - 빈 배열 조건 ForAllValues

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Empty array condition ForAllValues: The ForAllValues prefix with an empty condition key
 matches only if the key {{key}} is missing from the request context. To determine if
 the request context is empty, we recommend that you use the Null condition operator
 with the value of true instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The ForAllValues prefix with an empty condition key matches only
 if the key {{key}} is missing from the request context. To determine if the request
 context is empty, we recommend that you use the Null condition operator with the value
 of true instead."
```

제안 해결

`Condition` 요소 구조에서는 조건 연산자와 키 값 페어를 사용해야 합니다. `ForAllValues` 집합 연산자는 요청 집합의 모든 멤버 값이 조건 키 집합의 하위 집합인지를 테스트합니다.

빈 조건 키로 `ForAllValues`를 사용하면 요청에 키가 없는 경우에만 조건이 일치합니다. AWS에서는 요청 컨텍스트가 비어 있는지 테스트하려면 대신 `Null` 조건 연산자를 사용할 것을 권장합니다.

- 다중 값 컨텍스트 키
- Null 조건 연산자
- IAM JSON 정책 요소: Condition

제안 - 빈 배열 조건 ForAnyValue

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Empty array condition ForAnyValue: The ForAnyValue prefix with an empty condition key
 {{key}} never matches the request context and it does not affect the policy. Specify
 conditions.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The ForAnyValue prefix with an empty condition key {{key}} never
 matches the request context and it does not affect the policy. Specify conditions."
```

제안 해결

`Condition` 요소 구조에서는 조건 연산자와 키 값 페어를 사용해야 합니다. `ForAnyValues` 집합 연산자는 요청 값 집합에서 하나 이상의 멤버가 조건 키 값 집합에서 멤버 1개 이상과 일치하는지 테스트합니다.

빈 조건 키로 `ForAnyValues`를 사용하면 조건이 일치하지 않습니다. 즉, 문이 정책에 영향을 미치지 않습니다. AWS에서는 조건을 다시 작성할 것을 권장합니다.

- 다중 값 컨텍스트 키
- IAM JSON 정책 요소: Condition

제안 - 빈 배열 조건 IfExists

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Empty array condition IfExists: The IfExists suffix with an empty condition key matches
 only if the key {{key}} is missing from the request context. To determine if the
```

```
request context is empty, we recommend that you use the Null condition operator with
the value of true instead.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The IfExists suffix with an empty condition key matches only if the
key {{key}} is missing from the request context. To determine if the request context
is empty, we recommend that you use the Null condition operator with the value of true
instead."
```

제안 해결

`...IfExists` 접미사는 조건 연산자를 편집합니다. 이는 요청 컨텍스트에 정책 키가 있으면 정책에 지정된 대로 키를 처리하고, 키가 없으면 조건 요소를 true로 평가한다는 의미입니다.

빈 조건 키로 `...IfExists`를 사용하면 요청에 키가 없는 경우에만 조건이 일치합니다. AWS에서는 요청 컨텍스트가 비어 있는지 테스트하려면 대신 `Null` 조건 연산자를 사용할 것을 권장합니다.

- ...IfExists 조건 연산자
- IAM JSON 정책 요소: Condition

제안 - 빈 배열 보안 주체

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Empty array principal: This statement includes no principals and does not affect the
policy. Specify principals.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "This statement includes no principals and does not affect the
policy. Specify principals."
```

제안 해결

IAM 역할을 위한 신뢰 정책 및 리소스 기반 정책에서는 `Principal` 또는 `NotPrincipal` 요소를 사용해야 합니다. 리소스 기반 정책은 리소스에 직접 삽입할 수 있는 정책입니다.

문의 `Principal` 요소에 빈 배열을 지정하면 문이 정책에 영향을 미치지 않습니다. AWS에서는 리소스에 액세스할 수 있는 보안 주체를 지정할 것을 권장합니다.

- IAM JSON 정책 요소: Principal
- IAM JSON 정책 요소: NotPrincipal

제안 - 빈 배열 리소스

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Empty array resource: This statement includes no resources and does not affect the
 policy. Specify resources.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "This statement includes no resources and does not affect the policy.
 Specify resources."
```

제안 해결

문장에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다.

문의 리소스 요소에 빈 배열을 지정하면 문이 정책에 영향을 미치지 않습니다. AWS에서는 리소스의 Amazon 리소스 이름(ARN)을 지정할 것을 권장합니다.

- IAM JSON 정책 요소: Resource
- IAM JSON 정책 요소: NotResource

제안 - 빈 객체 조건

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Empty object condition: This condition block is empty and it does not affect the
 policy. Specify conditions.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "This condition block is empty and it does not affect the policy.
 Specify conditions."
```

제안 해결

Condition 요소 구조에서는 조건 연산자와 키 값 페어를 사용해야 합니다.

문의 조건 요소에 빈 객체를 제공하면 해당 문이 정책에 영향을 미치지 않습니다. 선택적 요소를 제거하거나 조건을 지정합니다.

- IAM JSON 정책 요소: Condition

제안 - 빈 객체 보안 주체

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Empty object principal: This statement includes no principals and does not affect the
 policy. Specify principals.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "This statement includes no principals and does not affect the
 policy. Specify principals."
```

제안 해결

IAM 역할을 위한 신뢰 정책 및 리소스 기반 정책에서는 Principal 또는 NotPrincipal 요소를 사용해야 합니다. 리소스 기반 정책은 리소스에 직접 삽입할 수 있는 정책입니다.

문의 Principal 요소에 빈 객체를 지정하면 문이 정책에 영향을 미치지 않습니다. AWS에서는 리소스에 액세스할 수 있어야 하는 보안 주체를 지정할 것을 권장합니다.

- IAM JSON 정책 요소: Principal
- IAM JSON 정책 요소: NotPrincipal

제안 - 빈 Sid 값

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Empty Sid value: Add a value to the empty string in the Sid element.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Add a value to the empty string in the Sid element."
```

제안 해결

선택적 Sid(문 ID) 요소를 사용하면 정책 문에 제공하는 식별자를 입력할 수 있습니다. Sid 값은 문 배열에서 각 문에 할당할 수 있습니다. Sid 요소를 사용할 경우 문자열 값을 제공해야 합니다.

관련 용어

- IAM JSON 정책 요소: Sid

제안 - IP 범위 개선

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Improve IP range: The non-zero bits in the IP address after the masked bits are
 ignored. Replace address with {{addr}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The non-zero bits in the IP address after the masked bits are
 ignored. Replace address with {{addr}}."
```

제안 해결

IP 주소 조건은 203.0.113.0/24 또는 2001:DB8:1234:5678::/64와 같은 표준 CIDR 형식이어야 합니다. 마스킹 처리된 비트 다음에 0이 아닌 비트를 포함하면 조건으로 간주되지 않습니다. AWS에서는 메시지에 포함된 새 주소를 사용할 것을 권장합니다.

- IP 주소 조건 연산자
- IAM JSON 정책 요소: Condition

제안 - 한정자의 Null

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Null with qualifier: Avoid using the Null condition operator with the ForAllValues or
 ForAnyValue qualifiers because they always return a true or false respectively.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Avoid using the Null condition operator with the ForAllValues or
 ForAnyValue qualifiers because they always return a true or false respectively."
```

제안 해결

Condition 요소에서 같음, 보다 작음 등의 조건 연산자를 사용하여 정책의 조건을 요청 컨텍스트의 키 및 값과 비교하는 표현식을 작성합니다. 단일 조건 키의 여러 값을 포함하는 요청의 경우 ForAllValues 또는 ForAnyValue 집합 연산자를 사용해야 합니다.

ForAllValues에 Null 조건 연산자를 사용하는 경우 문은 항상 true를 반환합니다. ForAnyValue에 Null 조건 연산자를 사용하는 경우 문은 항상 false를 반환합니다. AWS에서는 이러한 집합 연산자에 StringLike 조건 연산자를 사용할 것을 권장합니다.

관련 용어

- 다중 값 컨텍스트 키
- Null 조건 연산자
- 조건 요소

제안 - 프라이빗 IP 주소 하위 집합

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Private IP address subset: The values for condition key aws:SourceIp include a mix
 of private and public IP addresses. The private addresses will not have the desired
 effect. aws:SourceIp works only for public IP address ranges. To define permissions
 for private IP ranges, use aws:VpcSourceIp.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The values for condition key aws:SourceIp include a mix of private
 and public IP addresses. The private addresses will not have the desired effect.
 aws:SourceIp works only for public IP address ranges. To define permissions for
 private IP ranges, use aws:VpcSourceIp."
```

제안 해결

전역 조건 키 `aws:SourceIp`는 퍼블릭 IP 주소 범위에만 적용됩니다.

`Condition` 요소에 프라이빗 및 퍼블릭 IP 주소를 함께 포함하면 문에서 원하는 효과를 얻을 수 없습니다. `aws:VpcSourceIP`를 사용하여 프라이빗 IP 주소를 지정할 수 있습니다.

> **Note**
>
> 전역 조건 키 `aws:VpcSourceIP`는 요청이 지정된 IP 주소에서 시작되고 VPC 엔드포인트를 통과하는 경우에만 일치합니다.

- aws:SourceIp 전역 조건 키
- aws:VpcSourceIp 전역 조건 키
- IP 주소 조건 연산자
- IAM JSON 정책 요소: Condition

제안 - 프라이빗 NotIpAddress 하위 집합

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Private NotIpAddress subset: The values for condition key aws:SourceIp include a mix of
 private and public IP addresses. The private addresses have no effect. aws:SourceIp
 works only for public IP address ranges. To define permissions for private IP ranges,
 use aws:VpcSourceIp.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The values for condition key aws:SourceIp include a mix of private
 and public IP addresses. The private addresses have no effect. aws:SourceIp works
 only for public IP address ranges. To define permissions for private IP ranges, use
 aws:VpcSourceIp."
```

제안 해결

전역 조건 키 `aws:SourceIp`는 퍼블릭 IP 주소 범위에만 적용됩니다.

`Condition` 요소에 `NotIpAddress` 조건 연산자와 프라이빗 및 퍼블릭 IP 주소를 함께 포함하면 문에서 원하는 효과를 얻을 수 없습니다. 정책에 지정되지 않은 모든 퍼블릭 IP 주소는 일치합니다. 프라이빗 IP 주소는 일치하지 않습니다. 이 효과를 얻으려면 `aws:VpcSourceIP`에 `NotIpAddress`를 사용하고 일치하지 않아야 하는 프라이빗 IP 주소를 지정할 수 있습니다.

- aws:SourceIp 전역 조건 키
- aws:VpcSourceIp 전역 조건 키
- IP 주소 조건 연산자
- IAM JSON 정책 요소: Condition

제안 - 중복 작업

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Redundant action: The {{redundantActionCount}} action(s) are redundant because they
 provide similar permissions. Update the policy to remove the redundant action such as:
 {{redundantAction}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The {{redundantActionCount}} action(s) are redundant because they
 provide similar permissions. Update the policy to remove the redundant action such as:
 {{redundantAction}}."
```

제안 해결

`Action` 요소에 와일드카드(*)를 사용하는 경우 중복 권한을 포함할 수 있습니다. AWS에서는 정책을 검토하고 필요한 권한만 포함할 것을 권장합니다. 그러면 중복 작업을 제거하는 데 도움이 됩니다.

예를 들어 다음 작업에는 `iam:GetCredentialReport` 작업이 두 번 포함됩니다.

```
"Action": [
        "iam:Get*",
        "iam:List*",
```

```
        "iam:GetCredentialReport"
    ],
```

이 예제에서는 Get 또는 List로 시작하는 모든 IAM 작업에 대해 권한이 정의되어 있습니다. IAM이 가져오기 또는 나열 작업을 추가하면 이 정책에서 해당 작업을 허용합니다. 이러한 읽기 전용 작업을 모두 허용할 수 있습니다. iam:GetCredentialReport 작업은 iam:Get*의 일부로 이미 포함되어 있습니다. 중복 권한을 제거하려면 iam:GetCredentialReport를 제거할 수 있습니다.

작업의 모든 내용이 중복되는 경우 이 정책 검사의 결과가 표시됩니다. 이 예에서 요소에 iam:*CredentialReport가 포함된 경우 중복된 것으로 간주되지 않습니다. 여기에는 중복된 iam:GetCredentialReport 및 중복되지 않은 iam:GenerateCredentialReport가 포함됩니다. iam:Get* 또는 iam:*CredentialReport를 제거하면 정책의 권한이 변경됩니다.

- IAM JSON 정책 요소: Action

이 제안이 포함된 AWS 관리형 정책

AWS 관리형 정책을 사용하면 일반 AWS 사용 사례에 따라 권한을 할당하여 AWS를 시작할 수 있습니다.

중복 작업은 정책에 따라 부여되는 권한에 영향을 미치지 않습니다. AWS 관리형 정책을 참조하여 고객 관리형 정책을 생성하는 경우 AWS는 정책에서 중복 작업을 제거할 것을 권장합니다.

제안 - 중복 조건 값 숫자

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Redundant condition value num: Multiple values in {{operator}} are redundant. Replace
 with the {{greatest/least}} single value for {{key}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Multiple values in {{operator}} are redundant. Replace with the
 {{greatest/least}} single value for {{key}}."
```

제안 해결

조건 키의 비슷한 값에 숫자 조건 연산자를 사용하는 경우 중첩이 발생하여 권한이 중복될 수 있습니다.

예를 들어 다음 Condition 요소는 1,200초의 기간 중첩이 있는 여러 aws:MultiFactorAuthAge 조건을 포함합니다.

```
"Condition": {
        "NumericLessThan": {
          "aws:MultiFactorAuthAge": [
            "2700",
            "3600"
          ]
        }
      }
```

이 예에서는 멀티 팩터 인증(MFA)이 3,600초(1시간) 내에 완료된 경우의 권한이 정의됩니다. 중복된 2700 값을 제거할 수 있습니다.

- 숫자 조건 연산자
- IAM JSON 정책 요소: Condition

제안 - 중복 리소스

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Redundant resource: The {{redundantResourceCount}} resource ARN(s) are redundant
 because they reference the same resource. Review the use of wildcards (*)
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The {{redundantResourceCount}} resource ARN(s) are redundant because
 they reference the same resource. Review the use of wildcards (*)"
```

제안 해결

Amazon 리소스 이름(ARN)에 와일드카드(*)를 사용할 경우 중복 리소스 권한을 생성할 수 있습니다.

예를 들어 다음 Resource 요소는 중복 권한이 있는 여러 ARN을 포함합니다.

```
"Resource": [
            "arn:aws:iam::111122223333:role/jane-admin",
            "arn:aws:iam::111122223333:role/jane-s3only",
```

```
            "arn:aws:iam::111122223333:role/jane*"
        ],
```

이 예제에서는 이름이 jane으로 시작하는 모든 역할에 대해 권한을 정의합니다. 결과로 부여되는 권한을 변경하지 않고 중복되는 jane-admin 및 jane-s3only ARN을 제거할 수 있습니다. 이렇게 하면 동적 정책이 됩니다. 즉, jane으로 시작하는 모든 향후 역할의 권한을 정의합니다. 정책의 의도가 고정된 수의 역할에 액세스를 허용하려는 것인 경우 마지막 ARN을 제거하고 정의해야 하는 ARN만 나열합니다.

- IAM JSON 정책 요소: Resource

이 제안이 포함된 AWS 관리형 정책

AWS 관리형 정책을 사용하면 일반 AWS 사용 사례에 따라 권한을 할당하여 AWS를 시작할 수 있습니다.

중복 리소스는 정책에서 부여된 권한에 영향을 미치지 않습니다. AWS 관리형 정책을 참조하여 고객 관리형 정책을 생성하는 경우 AWS는 정책에서 중복 리소스를 제거할 것을 권장합니다.

제안 - 중복 문

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Redundant statement: The statements are redundant because they provide identical
 permissions. Update the policy to remove the redundant statement.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The statements are redundant because they provide identical
 permissions. Update the policy to remove the redundant statement."
```

제안 해결

Statement 요소는 정책의 주요 요소로서 필수입니다. Statement 요소는 단일 문 또는 개별 문의 배열을 포함할 수 있습니다.

긴 정책에 동일한 문을 두 번 이상 포함하면 문이 중복됩니다. 정책에서 부여하는 권한에 영향을 주지 않고 문 중 하나를 제거할 수 있습니다. 누군가 정책을 편집하면 복제본을 업데이트하지 않고 문 중 하나를 변경할 수 있습니다. 이로 인해 의도한 권한보다 늘어날 수 있습니다.

- IAM JSON 정책 요소: Statement

제안 - 서비스 이름의 와일드카드

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Wildcard in service name: Avoid using wildcards (*, ?) in the service name because it
 might grant unintended access to other AWS services with similar names.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Avoid using wildcards (*, ?) in the service name because it might
 grant unintended access to other AWS services with similar names."
```

제안 해결

정책에 AWS 서비스의 이름을 포함하는 경우 AWS에서는 와일드카드(*, ?)를 포함하지 말 것을 권장합니다. 와일드카드를 포함할 경우 향후 서비스에 의도하지 않은 권한이 추가될 수 있습니다. 예를 들어 이름에 단어 *code*가 포함된 AWS 서비스가 12개가 넘습니다.

```
"Resource": "arn:aws:*code*::111122223333:*"
```

- IAM JSON 정책 요소: Resource

제안 - 서비스에 대해 지원되지 않는 태그 조건 키를 사용하여 허용

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Allow with unsupported tag condition key for service: Using the effect Allow with the
 tag condition key {{conditionKeyName}} and actions for services with the following
 prefixes does not affect the policy: {{serviceNames}}. Actions for the listed service
 are not allowed by this statement. We recommend that you move these actions to a
 different statement without this condition key.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using the effect Allow with the tag condition key
 {{conditionKeyName}} and actions for services with the following prefixes does not
```

```
affect the policy: {{serviceNames}}. Actions for the listed service are not allowed
by this statement. We recommend that you move these actions to a different statement
without this condition key."
```

제안 해결

"Effect": "Allow"가 포함된 정책의 Condition 요소에 지원되지 않는 태그 조건 키를 사용하면 해당 서비스 작업에 대해 조건이 무시되므로 정책에서 부여되는 권한에 영향을 미치지 않습니다. AWS에서는 조건 키를 지원하지 않는 서비스 작업을 제거하고 해당 서비스의 특정 리소스에 대한 액세스를 허용하도록 다른 문을 작성할 것을 권장합니다.

aws:ResourceTag 조건 키를 사용할 때 서비스 작업에서 해당 키를 지원하지 않는 경우 요청 컨텍스트에 키가 포함되지 않습니다. 이 경우 Allow 문의 조건은 항상 false를 반환하고 작업이 허용되지 않습니다. 리소스가 올바로 태깅된 경우에도 이 문제가 발생합니다.

서비스에서 aws:ResourceTag 조건 키를 지원하는 경우 태그를 사용하여 해당 서비스 리소스에 대한 액세스를 제어할 수 있습니다. 이를 속성 기반 액세스 제어(ABAC)라고 합니다. 이러한 키를 지원하지 않는 서비스에서는 리소스 기반 액세스 제어(RBAC)를 사용하여 리소스에 대한 액세스를 제어해야 합니다.

> **Note**
>
> 일부 서비스에서는 해당 리소스 및 작업의 하위 집합에 대해 aws:ResourceTag 조건 키 지원을 허용합니다. IAM Access Analyzer에서는 지원되지 않는 서비스 작업에 대한 결과를 반환합니다. 예를 들어 Amazon S3는 해당 리소스의 하위 집합에 대해 aws:ResourceTag를 지원합니다. Amazon S3에서 사용할 수 있는 리소스 유형 중 aws:ResourceTag 조건 키를 지원하는 유형을 모두 보려면 서비스 승인 참조의 Amazon S3에서 정의한 리소스 유형을 참조하세요.

예를 들어 팀원이 키 값 페어 team=BumbleBee로 태깅된 특정 리소스에 대한 세부 정보를 볼 수 있도록 하려고 한다고 가정합니다. 또한 AWS Lambda에서 리소스를 태깅할 수 있지만 aws:ResourceTag 조건 키를 지원하지 않는다고 가정합니다. 이 태그가 있는 경우 AWS App Mesh 및 AWS Backup에 대한 보기 작업을 허용하려면 aws:ResourceTag 조건 키를 사용합니다. Lambda의 경우 팀 이름을 접두사로 포함하는 리소스 명명 규칙을 사용합니다. 그런 다음에 해당 명명 규칙을 사용하는 리소스 보기를 허용하는 별도의 문을 포함합니다.

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

```
        {
            "Sid": "AllowViewSupported",
            "Effect": "Allow",
            "Action": [
                "appmesh:DescribeMesh",
                "backup:GetBackupPlan"
                ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/team": "BumbleBee"
                }
            }
        },
        {
            "Sid": "AllowViewUnsupported",
            "Effect": "Allow",
            "Action": "lambda:GetFunction",
            "Resource": "arn:aws:lambda:*:123456789012:function:team-BumbleBee*"
        }
    ]
}
```

> **Warning**
>
> 이 결과의 해결 방법으로 `"Effect": "Allow"`에 조건 연산자의 Not 버전을 사용하지 마세요. 이러한 조건 연산자는 부정 일치를 제공합니다. 즉, 조건이 평가된 후 결과가 부정됩니다. 앞의 예제에서 `StringNotEquals` 연산자가 있는 `AllowViewSupported` 문에 `lambda:GetFunction` 작업을 포함하면 리소스가 태깅되어 있는지 여부와 관계없이 항상 작업이 허용됩니다.
>
> 이 결과의 해결 방법으로 조건 연산자의 ...IfExists 버전을 사용하지 마세요. 이는 "요청 컨텍스트에 키가 있고 값이 일치하면 작업을 허용합니다. 그렇지 않으면 작업을 허용합니다."를 의미합니다. 앞의 예제에서 `StringEqualsIfExists` 연산자가 있는 `AllowViewSupported` 문에 `lambda:GetFunction` 작업을 포함하면 작업이 항상 허용됩니다. 해당 작업의 경우 컨텍스트에 키가 없으며 리소스가 태깅되어 있는지 여부에 관계없이 해당 리소스 유형을 보려는 모든 시도는 허용됩니다.

관련 용어

- 전역 조건 키

- IAM JSON 정책 요소: 조건 연산자
- 조건 요소
- JSON 정책 개요

제안 - 서비스에 대해 지원되지 않는 태그 조건 키를 사용하여 NotAction 허용

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Allow NotAction with unsupported tag condition key for service: Using the effect Allow
 with NotAction and the tag condition key {{conditionKeyName}} allows only service
 actions that support the condition key. The condition key doesn't apply to some
 service actions. We recommend that you use Action instead of NotAction.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "Using the effect Allow with NotAction and the tag condition key
 {{conditionKeyName}} allows only service actions that support the condition key. The
 condition key doesn't apply to some service actions. We recommend that you use Action
 instead of NotAction."
```

제안 해결

`NotAction` 요소 및 `"Effect": "Allow"`가 포함된 조건의 `Condition` 요소에 지원되지 않는 태그 조건 키를 사용하면 정책에서 부여되는 권한에 영향을 미치지 않습니다. 조건 키를 지원하지 않는 서비스 작업의 경우 조건이 무시됩니다. AWS에서는 작업 목록을 허용하도록 로직을 다시 작성할 것을 권장합니다.

`aws:ResourceTag` 조건 키를 `NotAction`과 함께 사용하는 경우 키를 지원하지 않는 모든 신규 또는 기존 서비스 작업이 허용되지 않습니다. AWS에서는 허용할 작업을 명시적으로 나열할 것을 권장합니다. IAM Access Analyzer에서는 `aws:ResourceTag` 조건 키를 지원하지 않는 나열된 작업에 대해 별도의 결과를 반환합니다. 자세한 내용은 제안 - 서비스에 대해 지원되지 않는 태그 조건 키를 사용하여 허용 단원을 참조하십시오.

서비스에서 `aws:ResourceTag` 조건 키를 지원하는 경우 태그를 사용하여 해당 서비스 리소스에 대한 액세스를 제어할 수 있습니다. 이를 속성 기반 액세스 제어(ABAC)라고 합니다. 이러한 키를 지원하지 않는 서비스에서는 리소스 기반 액세스 제어(RBAC)를 사용하여 리소스에 대한 액세스를 제어해야 합니다.

관련 용어

- 전역 조건 키
- ABAC와 RBAC 비교
- IAM JSON 정책 요소: 조건 연산자
- 조건 요소
- JSON 정책 개요

제안 - 서비스 보안 주체에 권장되는 조건 키

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Recommended condition key for service principal: To restrict access to the service
 principal {{servicePrincipalPrefix}} operating on your behalf, we recommend
 aws:SourceArn, aws:SourceAccount, aws:SourceOrgID, or aws:SourceOrgPaths instead of
 {{key}}.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "To restrict access to the service principal
 {{servicePrincipalPrefix}} operating on your behalf, we recommend aws:SourceArn,
 aws:SourceAccount, aws:SourceOrgID, or aws:SourceOrgPaths instead of {{key}}."
```

제안 해결

서비스 보안 주체를 사용하는 리소스 기반 정책의 `Principal` 요소에서 AWS 서비스, 즉 서비스에 대한 식별자를 지정할 수 있습니다. `aws:Referer`와(과) 같은 다른 조건 키 대신 서비스 보안 주체에 액세스 권한을 부여할 때는 `aws:SourceArn`, `aws:SourceAccount`, `aws:SourceOrgID`, 또는 `aws:SourceOrgPaths`을(를) 사용해야 합니다. 이렇게 하면 혼동된 대리자 문제라 불리는 보안 문제를 예방할 수 있습니다.

관련 용어

- AWS 서비스 보안 주체
- AWS 전역 조건 키: aws:SourceAccount
- AWS 전역 조건 키: aws:SourceArn

- AWS 전역 조건 키: aws:SourceOrgId
- AWS 전역 조건 키: aws:SourceOrgPaths
- 혼동된 대리자 문제

제안 - 정책의 관련 없는 조건 키

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Irrelevant condition key in policy: The condition key {{condition-key}} is not relevant
 for the {{resource-type}} policy.  Use this key in an identity-based policy to govern
 access to this resource.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The condition key {{condition-key}} is not relevant for the
 {{resource-type}} policy.  Use this key in an identity-based policy to govern access
 to this resource."
```

제안 해결

일부 조건 키는 리소스 기반 정책과 관련이 없습니다. 예를 들어, `s3:ResourceAccount` 조건 키는 Amazon S3 버킷 또는 Amazon S3 액세스 포인트 리소스 유형에 연결된 리소스 기반 정책과 관련이 없습니다.

리소스에 대한 액세스를 제어하려면 자격 증명 기반 정책의 조건을 사용해야 합니다.

관련 용어

- 자격 증명 기반 정책 및 리소스 기반 정책

제안 – 역할 신뢰 정책에 중복된 보안 주체

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Redundant principal in role trust policy: The assumed-role principal
 {{redundant_principal}} is redundant with its parent role {{parent_role}}. Remove the
 assumed-role principal.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The assumed-role principal {{redundant_principal}} is redundant with
 its parent role {{parent_role}}. Remove the assumed-role principal."
```

제안 해결

수임된 역할 보안 주체와 정책 `Principal` 요소의 상위 역할을 지정할 경우 다른 권한을 허용하거나 거부하지 않습니다. 예를 들어 다음의 형식을 사용하여 `Principal` 요소를 지정하면 중복입니다.

```
"Principal": {
            "AWS": [
            "arn:aws:iam::AWS-account-ID:role/rolename",
            "arn:aws:iam::AWS-account-ID:assumed-role/rolename/rolesessionname"
        ]
```

위임된 역할 보안 주체를 제거하는 것이 좋습니다.

관련 용어

- 역할 세션 보안 주체

제안 – 대상 클레임 유형 확인

AWS Management Console에서 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
Confirm audience claim type: The 'aud' (audience) claim key identifies the recipients
 that the JSON web token is intended for. Audience claims can be multivalued or single-
valued. If the claim is multivalued, use a ForAllValues or ForAnyValue qualifier. If
 the claim is single-valued, do not use a qualifier.
```

AWS CLI 또는 AWS API를 프로그래밍 방식으로 호출할 경우 이 검사의 결과에는 다음 메시지가 포함됩니다.

```
"findingDetails": "The 'aud' (audience) claim key identifies the recipients that the
 JSON web token is intended for. Audience claims can be multivalued or single-valued.
 If the claim is multivalued, use a ForAllValues or ForAnyValue qualifier. If the claim
 is single-valued, do not use a qualifier."
```

제안 해결

aud(대상) 클레임 키는 IdP로 앱을 등록할 때 발급된 앱의 고유 식별자이고 JSON 웹 토큰을 제공할 수령자를 식별합니다. 대상 클레임은 다중 값이거나 단일 값일 수 있습니다. 클레임이 다중 값인 경우 `ForAllValues` 또는 `ForAnyValue` 조건 세트 연산자를 사용하세요. 클레임이 단일 값인 경우 조건 세트 연산자를 사용하지 마세요.

관련 용어

- 웹 아이덴티티 또는 OpenID Connect 페더레이션을 위한 역할 생성(콘솔)
- 다중 값 컨텍스트 키
- 단일 값 vs. 다중 값 조건 키

IAM Access Analyzer 정책 확인

AWS Identity and Access Management Access Analyzer 사용자 지정 정책 확인을 사용하여 지정된 보안 표준에 대해 정책을 검증할 수 있습니다. 실행할 수 있는 사용자 지정 정책 확인의 유형은 다음과 같습니다.

- 참조 정책 비교 확인: 정책을 편집할 때 업데이트된 정책이 기존 버전의 정책과 같은 참조 정책과 비교하여 새 액세스를 허용하는지 확인할 수 있습니다. IAM 콘솔의 AWS Command Line Interface(AWS CLI), IAM Access Analyzer API(API) 또는 JSON 정책 편집기를 사용하여 정책을 편집할 때 해당 확인을 실행할 수 있습니다.

  > **Note**
  >
  > IAM Access Analyzer 사용자 지정 정책 검사는 참조 리소스 정책의 `Principal` 요소에 와일드카드를 허용합니다.

- IAM 작업 또는 리소스 목록과 비교 확인: 정책에서 특정 IAM 작업 또는 리소스를 허용하지 않는지 확인할 수 있습니다. 작업만 지정된 경우 IAM Access Analyzer는 정책의 모든 리소스에서 작업의 액세스 권한을 확인합니다. 리소스만 지정된 경우 IAM Access Analyzer는 지정된 리소스에 액세스할 수 있는 작업이 무엇인지 확인합니다. 작업과 리소스가 모두 지정된 경우 IAM Access Analyzer는 지정된 작업 중 지정된 리소스에 액세스할 수 있는 작업이 무엇인지 확인합니다. AWS CLI 또는 API를 사용하여 정책을 생성하거나 편집할 때 해당 확인을 실행할 수 있습니다.
- 퍼블릭 액세스 확인: 리소스 정책이 지정된 리소스 유형에 대한 퍼블릭 액세스 권한을 부여할 수 있는지 여부를 확인할 수 있습니다. AWS CLI 또는 API를 사용하여 정책을 생성하거나 편집할 때 해당 확인을 실행할 수 있습니다. 이 유형의 사용자 지정 정책 확인은 계정 또는 외부 액세스 분석기 컨텍스트가 필요하지 않으므로 액세스 미리 보기와 다릅니다. 액세스 미리 보기를 사용하면 리소스 권한

을 배포하기 전에 IAM Access Analyzer 조사 결과를 미리 볼 수 있으며, 사용자 지정 확인은 정책이 퍼블릭 액세스 권한을 부여할 수 있는지 여부를 확인합니다.

각 사용자 지정 정책 확인에는 요금이 부과됩니다. 요금에 대한 자세한 내용은 IAM Access Analyzer 요금을 참조하세요.

사용자 지정 정책 확인 작동 방식

자격 증명 및 리소스 기반 정책에 대해 사용자 지정 정책 확인을 실행할 수 있습니다. 사용자 지정 정책 확인은 정책에 의해 새 액세스 또는 지정된 액세스가 허용되는지 여부를 결정할 때 패턴 매칭 기법이나 액세스 로그 확인에 의존하지 않습니다. 외부 액세스 조사 결과와 마찬가지로 사용자 지정 정책 확인은 Zelkova를 기반으로 합니다. Zelkova는 IAM 정책을 동등한 논리적 명령문으로 변환하며 문제에 대해 범용 및 특수 논리 해석기(만족성 모듈로 이론)를 실행합니다. 새 액세스 또는 지정된 액세스를 확인하기 위해 IAM Access Analyzer는 Zelkova를 정책에 반복적으로 적용합니다. 정책의 내용에 따라 정책이 허용하는 동작 클래스를 특성화하기 위해 쿼리가 점점 더 구체화되고 있습니다. 만족성 모듈로 이론에 대한 자세한 내용은 만족성 모듈로 이론을 참조하세요.

드물지만 IAM Access Analyzer에서 정책 설명이 새 액세스 권한을 부여하는지 또는 지정된 액세스 권한을 부여하는지 여부를 완전히 확인할 수 없을 수 있습니다. 이러한 경우 사용자 지정 정책 확인에 실패하여 거짓 긍정을 선언하는 오류가 발생합니다. IAM Access Analyzer는 포괄적인 정책 평가를 제공하도록 설계되었으며 거짓 부정을 최소화하기 위해 노력합니다. 이 접근 방식은 IAM Access Analyzer가 확인을 통과해도 정책에 의해 액세스 권한이 부여되지 않았을 정도로 높은 수준의 보증을 제공한다는 것을 의미합니다. IAM Access Analyzer의 응답에 보고된 정책 설명을 검토하여 실패한 확인을 수동으로 검사할 수 있습니다.

새 액세스를 확인할 참조 정책 예제

GitHub의 IAM Access Analyzer 사용자 지정 정책 확인 샘플 리포지토리에서 참조 정책의 예를 찾고 새 액세스에 대한 사용자 지정 정책 확인을 설정 및 실행하는 방법을 배울 수 있습니다.

> **이 예제를 사용하기 전에**
>
> 샘플 참조 정책을 사용하기 전에 다음 단계를 수행해야 합니다.
>
> - 신중하게 참조 정책을 검토하고 자신의 요구 사항에 맞게 사용자 지정합니다.
> - 자신의 환경과 사용 중인 AWS 서비스에서 참조 정책을 철저히 테스트합니다.
>
> 참조 정책은 사용자 지정 정책 확인을 구현하고 사용하는 방법을 보여줍니다. 그러나 제시된 그대로 실행할 수 있는 공식적인 AWS 권장 사항 또는 모범 사례로 해석해서는 안 됩니다. 고

객은 자신의 환경이 가진 보안 요구 사항을 해결하기 위해 참조 정책의 적합성을 테스트할 책임이 있습니다.

- 사용자 지정 정책 확인은 분석 시 환경에 구애받지 않습니다. 분석에서는 입력 정책에 포함된 정보만 고려합니다. 예를 들어 사용자 지정 정책 확인으로는 계정이 특정 AWS 조직의 구성원인지 여부를 확인할 수 없습니다. 따라서 사용자 지정 정책 확인에서는 `aws:PrincipalOrgId` 및 `aws:PrincipalAccount` 조건 키 값을 기반으로 새 액세스를 비교할 수 없습니다.

실패한 사용자 지정 정책 확인 점검

사용자 지정 정책 확인이 실패할 경우 IAM Access Analyzer의 응답에는 확인 실패를 초래한 정책 설명의 명령문 ID(`Sid`)가 포함됩니다. 명령문 ID는 선택적 정책 요소이지만 모든 정책 설명에 대해 명령문 ID를 추가하는 것이 좋습니다. 또한 사용자 지정 정책 확인은 확인 실패 원인을 식별하는 데 도움이 되는 명령문 색인을 반환합니다. 명령문 색인은 0 기준 번호 매기기를 따르며, 여기서 첫 번째 명령문은 0으로 참조됩니다. 확인 실패의 원인이 되는 명령문이 여러 개 있는 경우 확인에서는 한 번에 하나의 명령문 ID만 반환합니다. 원인에 강조 표시된 설명을 수정하고 통과할 때까지 확인을 다시 실행하는 것이 좋습니다.

사용자 지정 정책 확인을 통한 정책 검증(콘솔)

선택 단계로 IAM 콘솔의 JSON 정책 편집기에서 정책을 편집할 때 사용자 지정 정책 확인을 실행할 수 있습니다. 업데이트된 정책이 기존 버전과 비교하여 새 액세스를 허용하는지 확인할 수 있습니다.

IAM JSON 정책을 편집할 때 새 액세스 권한을 확인하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 정책을 선택합니다.
3. 정책 목록에서 편집하려는 정책명을 선택합니다. 검색 상자를 사용하여 정책 목록을 필터링할 수 있습니다.
4. 권한 탭을 선택한 다음 편집을 선택합니다.
5. JSON 옵션을 선택하고 정책을 업데이트합니다.
6. 정책 아래의 정책 검증 창에서 새 액세스 확인 탭을 선택한 다음 정책 확인을 선택합니다. 수정된 권한으로 새 액세스 권한이 부여되는 경우 정책 검증 창에서 해당 설명이 강조 표시됩니다.
7. 새 액세스 권한을 부여하지 않으려면 새 액세스가 감지되지 않을 때까지 정책 설명을 업데이트하고 정책 확인을 선택합니다.

> **Note**
>
> 새 액세스를 확인할 때마다 요금이 부과됩니다. 요금에 대한 자세한 내용은 IAM Access Analyzer 요금을 참조하세요.

8. Next(다음)를 선택합니다.
9. 검토 및 저장 페이지에서 이 정책에 정의된 권한을 검토한 다음 변경 사항 저장을 선택합니다.

사용자 지정 정책 확인(AWS CLI 또는 API)으로 정책 검증

AWS CLI 또는 IAM Access Analyzer API에서 IAM Access Analyzer 사용자 지정 정책 확인을 실행할 수 있습니다.

IAM Access Analyzer 정책 확인을 수행하려면(AWS CLI)

- 기존 정책과 비교하여 업데이트된 정책에 새 액세스가 허용되는지 확인하려면 다음 명령을 실행합니다. check-no-new-access
- 지정된 액세스가 정책에서 허용되지 않는지 확인하려면 다음 명령을 실행합니다. check-access-not-granted
- 리소스 정책이 지정된 리소스 유형에 대한 퍼블릭 액세스 권한을 부여할 수 있는지 확인하려면 다음 명령을 실행합니다. check-no-public-access

IAM Access Analyzer 사용자 지정 정책 확인(API)을 실행하려면

- 기존 정책과 비교하여 업데이트된 정책에 새 액세스가 허용되는지 확인하려면 CheckNoNewAccess API 작업을 사용하세요.
- 지정된 액세스가 정책에서 허용되지 않는지 확인하려면 CheckAccessNotGranted API 작업을 사용하세요.
- 리소스 정책이 지정된 리소스 유형에 대한 퍼블릭 액세스 권한을 부여할 수 있는지 확인하려면 CheckNoPublicAccess API 작업을 사용하세요.

IAM Access Analyzer 정책 생성

관리자나 개발자는 IAM 엔터티(사용자 또는 역할)에 필요한 것 이상의 권한을 부여할 수 있습니다. IAM에서는 부여하는 권한을 구체화하는 데 도움이 되는 몇 가지 옵션을 제공합니다. 그중 하나는 엔터

티에 대한 액세스 활동을 기반으로 하는 IAM 정책을 생성하는 것입니다. IAM Access Analyzer는 사용자의 AWS CloudTrail 로그를 검토하고 지정된 날짜 범위에 엔터티에서 사용한 권한을 포함하는 정책 템플릿을 생성합니다. 템플릿을 사용하여 특정 사용 사례를 지원하는 데 필요한 권한만 부여하는 세분화된 권한을 가진 정책을 만들 수 있습니다.

주제

정책 생성 작동 원리

IAM Access Analyzer는 CloudTrail 이벤트를 분석하여 IAM 엔터티(사용자 또는 역할)에서 사용한 작업 및 서비스를 식별합니다. 그런 다음 해당 활동을 기반으로 하는 IAM 정책을 생성합니다. 엔터티에 연결된 광범위한 사용 권한 정책을 생성된 정책으로 대체할 때 엔터티의 권한을 세분화할 수 있습니다. 다음은 정책 생성 프로세스에 대한 고급 개요입니다.

- 정책 템플릿 생성 설정 – 과거 AWS CloudTrail 이벤트를 IAM Access Analyzer에서 분석할 기간을 최대 90일까지 지정합니다. 기존 서비스 역할을 지정하거나 새 역할을 생성해야 합니다. 서비스 역할은 CloudTrail 추적 및 서비스에 마지막으로 액세스한 정보에 대한 IAM Access Analyzer 액세스를 제공하여 사용된 서비스 및 작업을 식별합니다. 정책을 생성하려면 먼저 계정에 대한 이벤트를 로깅하는 CloudTrail 추적을 지정해야 합니다. IAM Access Analyzer의 CloudTrail 데이터 할당량에 대한 자세한 내용은 IAM Access Analyzer 할당량을 참조하세요.
- 정책 생성 – IAM Access Analyzer는 CloudTrail 이벤트의 액세스 활동을 기반으로 정책을 생성합니다.
- 정책 검토 및 사용자 지정 – 정책이 생성된 후 지정된 날짜 범위 동안 엔터티가 사용한 서비스 및 작업을 검토할 수 있습니다. 권한을 추가 또는 제거하고, 리소스를 지정하고, 정책 템플릿에 조건을 추가하여 정책을 추가로 사용자 지정할 수 있습니다.

- 정책 생성 및 연결 – 관리형 정책을 생성하여 생성된 정책을 저장하는 옵션이 있습니다. 정책을 생성하는 데 활동이 사용된 사용자 또는 역할에 생성한 정책을 연결할 수 있습니다.

서비스 및 작업 수준 정보

IAM Access Analyzer에서 IAM 정책을 생성하면 정책을 추가로 사용자 지정하는 데 도움이 되는 정보가 반환됩니다. 정책이 생성되면 다음 두 가지 범주의 정보가 반환될 수 있습니다.

- 작업 수준 정보가 포함된 정책 - Amazon EC2와 같은 일부 AWS 서비스의 경우 IAM Access Analyzer는 CloudTrail 이벤트에서 발견된 작업을 식별하고 생성된 정책에 사용된 작업을 나열할 수 있습니다. 지원되는 서비스의 목록은 IAM Access Analyzer 정책 생성 서비스 단원을 참조하세요. 일부 서비스의 경우 생성된 정책에 서비스에 대한 작업을 추가하라는 IAM Access Analyzer 메시지가 표시됩니다.
- 서비스 수준 정보 –이(가) 포함된 정책 IAM Access Analyzer는 마지막으로 액세스한 정보를 사용하여 최근에 사용한 모든 서비스가 포함된 정책 템플릿을 생성합니다. AWS Management Console을(를) 사용할 때 서비스를 검토하고 정책을 완료하기 위한 작업을 추가하라는 메시지가 표시됩니다.

각 서비스의 작업 목록은 서비스 승인 참조의 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

정책 생성에 대해 알아야 할 사항

정책을 생성하기 전에 다음과 같은 중요한 세부 정보를 검토합니다.

- CloudTrail 추적 사용 – 액세스 활동을 기반으로 정책을 생성하려면 계정에 대해 CloudTrail 추적을 사용하도록 설정해야 합니다. CloudTrail 추적을 생성할 때 CloudTrail은(는) 지정한 Amazon S3 버킷으로 추적과 관련된 이벤트를 보냅니다. CloudTrail 추적을 생성하는 방법을 알아보려면 AWS CloudTrail 사용 설명서의 AWS 계정에 대한 추적 생성을 참조하세요.
- 데이터 이벤트를 사용할 수 없음 - IAM Access Analyzer는 생성된 정책에서 Amazon S3 데이터 이벤트와 같은 데이터 이벤트에 대한 작업 수준 활동을 식별하지 않습니다.
- PassRole – CloudTrail에 의해 `iam:PassRole` 작업이 추적되지 않으며 생성된 정책에 포함되지 않습니다.
- 정책 생성 시간 단축 – 정책을 더 빠르게 생성하려면 정책 생성을 위해 설정하는 동안 지정하는 날짜 범위를 줄입니다.

- 감사에 CloudTrail 사용 – 감사 목적으로 정책 생성을 사용하지 마세요. 대신 CloudTrail을(를) 사용합니다. CloudTrail 사용에 대한 자세한 내용은 AWS CloudTrail로 IAM 및 AWS STS API 호출 로깅을 참조하세요.
- 거부된 작업 — 정책 생성은 거부된 작업을 포함하여 모든 CloudTrail 이벤트를 검토합니다.
- 하나의 정책 IAM 콘솔 – IAM 콘솔에 한 번에 하나의 정책을 생성할 수 있습니다.
- 생성된 정책 가용성 IAM 콘솔 – 생성된 후 최대 7일 동안 IAM 콘솔에서 생성된 정책을 검토할 수 있습니다. 7일 후에는 새 정책을 생성해야 합니다.
- 정책 생성 할당량 – IAM Access Analyzer 정책 생성 할당량에 대한 자세한 내용은 IAM Access Analyzer 할당량을 참조하세요.
- Amazon S3 Standard 요금 적용 - 정책 생성 기능을 사용하면 IAM Access Analyzer가 S3 버킷의 CloudTrail 로그를 검토합니다. 정책 생성을 위해 CloudTrail 로그에 액세스하는 데 드는 추가 스토리지 요금은 없습니다. AWS는 S3 버킷에 저장된 CloudTrail 로그의 요청 및 데이터 전송에 대해 표준 Amazon S3 요금을 부과합니다.
- AWS Control Tower 지원 — 정책 생성은 AWS Control Tower의 정책 생성을 지원하지 않습니다.

정책을 생성하는 데 필요한 권한

처음으로 정책을 생성하는 데 필요한 권한은 후속 사용을 위해 정책을 생성하는 데 필요한 권한과 다릅니다.

최초 설정

정책을 처음 생성하는 경우 계정에서 적합한 기존 서비스 역할을 선택하거나 새 서비스 역할을 생성해야 합니다. 서비스 역할은 계정에서 CloudTrail 및 서비스에 마지막으로 액세스한 정보에 대한 IAM Access Analyzer 액세스를 제공합니다. 관리자만 역할을 생성하고 구성하는 데 필요한 권한이 있어야 합니다. 따라서 관리자가 처음 설치하는 동안 서비스 역할을 생성하는 것이 좋습니다. 서비스 역할을 생성하는 데 필요한 권한에 대한 자세한 내용은 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

서비스 역할에 필요한 권한

서비스 역할을 생성할 때 역할에 대해 두 개의 정책을 구성합니다. 역할이 수행할 수 있는 작업을 지정하는 역할에 IAM 권한 정책을 연결합니다. 또한 역할을 사용할 수 있는 보안 주체를 지정하는 역할에 역할 신뢰 정책을 연결합니다.

첫 번째 예시 정책은 정책을 생성하는 데 필요한 서비스 역할에 대한 권한 정책을 보여 줍니다. 두 번째 예시 정책은 서비스 역할에 필요한 역할 신뢰 정책을 보여 줍니다. 이러한 정책을 사용하여 AWS API

또는 AWS CLI을(를) 사용하거나 정책을 생성할 때 서비스 역할을 생성할 수 있습니다. IAM 콘솔을 사용하여 정책 생성 프로세스의 일부로 서비스 역할을 생성하면 이러한 정책이 자동으로 생성됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "cloudtrail:GetTrail",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetServiceLastAccessedDetails",
                "iam:GenerateServiceLastAccessedDetails"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

다음 예시 정책은 IAM Access Analyzer이(가) 역할을 수임할 수 있게 하는 권한이 있는 역할 신뢰 정책을 보여 줍니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "access-analyzer.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

후속 사용

AWS Management Console에서 정책을 생성하려면 IAM 사용자에게 정책 생성에 사용되는 서비스 역할을 IAM Access Analyzer에 전달할 수 있는 권한 정책이 있어야 합니다. `iam:PassRole`은 일반적으로 사용자가 전달할 역할의 세부 정보를 얻을 수 있도록 `iam:GetRole`과 함께 제공됩니다. 이 예시에서 사용자는 지정된 계정에 있으며 다음과 같이 이름이 `AccessAnalyzerMonitorServiceRole*`(으)로 시작하는 역할만 전달할 수 있습니다. AWS 서비스에 IAM 역할 전달에 대한 자세한 내용은 사용자에게 AWS 서비스에 역할을 전달할 수 있는 권한 부여를 참조하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUserToPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/service-role/
AccessAnalyzerMonitorServiceRole*"
    }
  ]
}
```

또한 다음 정책 문에 나와 있는 대로 AWS Management Console, AWS API 또는 AWS CLI에서 정책을 생성하려면 다음 IAM Access Analyzer 권한이 있어야 합니다.

```
{
  "Sid": "AllowUserToGeneratePolicy",
  "Effect": "Allow",
  "Action": [
    "access-analyzer:CancelPolicyGeneration",
    "access-analyzer:GetGeneratedPolicy",
```

```
    "access-analyzer:ListPolicyGenerations",
    "access-analyzer:StartPolicyGeneration"
  ],
  "Resource": "*"
}
```

첫 번째 및 후속 사용 모두

AWS Management Console을 사용하여 정책을 생성하는 경우 다음 정책 문에 나와 있는 것과 같이 계정의 CloudTrail 추적을 나열할 수 있는 `cloudtrail:ListTrails` 권한이 있어야 합니다.

```
{
  "Sid": "AllowUserToListTrails",
  "Effect": "Allow",
  "Action": [
    "CloudTrail:ListTrails"
  ],
  "Resource": "*"
}
```

CloudTrail 활동을 기반으로 정책 생성(콘솔)

IAM 사용자 또는 역할에 대한 정책을 생성할 수 있습니다.

1단계: CloudTrail 활동을 기반으로 정책 생성

다음 절차에서는 AWS Management Console을(를) 사용하여 역할에 대한 정책을 생성하는 방법에 관해 설명합니다.

IAM 역할에 대한 정책 생성

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 역할을 선택합니다.

   > **Note**
   >
   > IAM 사용자에 대한 활동을 기반으로 정책을 생성하는 단계는 거의 동일합니다. 이렇게 하려면 역할(Roles) 대신 사용자(Users)를 선택합니다.

3. 계정의 역할 목록에서 정책을 생성하는 데 사용할 활동을 가진 역할의 이름을 선택합니다.

4. 권한(Permissions) 탭의 CloudTrail 이벤트 기반 정책 생성 섹션에서 정책 생성(Generate policy)을 선택합니다.
5. 정책 생성 페이지에서 해당 역할로 수행된 작업에 대해 IAM Access Analyzer이(가) CloudTrail 이벤트를 분석할 기간을 지정합니다. 최대 90일 범위를 선택할 수 있습니다. 정책 생성 시간을 줄이려면 가능한 한 가장 짧은 기간을 선택하는 것이 좋습니다.
6. CloudTrail 액세스(CloudTrail access) 섹션에서 적절한 기존 역할을 선택하거나 적절한 역할이 없는 경우 새 역할을 만듭니다. 이 역할은 액세스 활동을 검토하고 사용된 서비스 및 작업을 식별하기 위해 사용자를 대신해 CloudTrail 추적에 액세스할 수 있는 권한을 IAM Access Analyzer에 부여합니다. 이 역할에 필요한 권한에 대해 알아보려면 정책을 생성하는 데 필요한 권한 섹션을 참조하세요.
7. 분석할 CloudTrail 추적 섹션에서 계정에 대한 이벤트를 기록하는 CloudTrail 추적을 지정합니다.

   로그를 다른 계정에 저장하는 CloudTrail 추적을 선택하면 크로스 계정 액세스에 대한 정보 상자가 표시됩니다. 크로스 계정 액세스를 사용하려면 추가로 설정해야 합니다. 자세한 내용은 이 주제 뒷부분의 Choose a role for cross-account access 섹션을 참조하세요.
8. 정책 생성(Generate policy)을 선택합니다.
9. 정책 생성이 진행되는 동안 권한(Permissions) 탭의 역할 요약(Roles Summary) 페이지로 돌아갑니다. 정책 요청 세부 정보 섹션의 상태가 성공(Success)으로 표시될 때까지 기다린 다음 생성된 정책 보기(View generated policy)를 선택합니다. 최대 7일간 생성된 정책을 볼 수 있습니다. 다른 정책을 생성하면 기존 정책이 생성된 새 정책으로 대체됩니다.

2단계: 권한 검토 및 사용된 서비스에 대한 작업 추가

IAM Access Analyzer이(가) 역할이 사용되었음을 확인한 서비스 및 작업을 검토합니다. 생성된 정책 템플릿에 사용된 모든 서비스에 대한 작업을 추가할 수 있습니다.

1. 다음 섹션을 검토하세요.

   - 권한 검토 페이지에서 생성된 정책에 포함된 작업(Actions included in the generated policy) 목록을 검토합니다. 목록에는 지정된 날짜 범위 내에서 역할에 의해 사용된 것으로 IAM Access Analyzer이(가) 확인한 서비스 및 작업이 표시됩니다.
   - 사용된 서비스(Services used) 섹션에는 지정된 날짜 범위 내에서 역할에 의해 사용된 것으로 IAM Access Analyzer이(가) 확인한 추가 서비스가 표시됩니다. 사용된 작업에 대한 정보는 이 섹션에 나열된 서비스에 대해 제공되지 않을 수 있습니다. 나열된 각 서비스에 대한 메뉴를 사용하여 정책에 포함할 작업을 수동으로 선택합니다.
2. 작업 추가를 완료했으면 다음(Next)을 선택합니다.

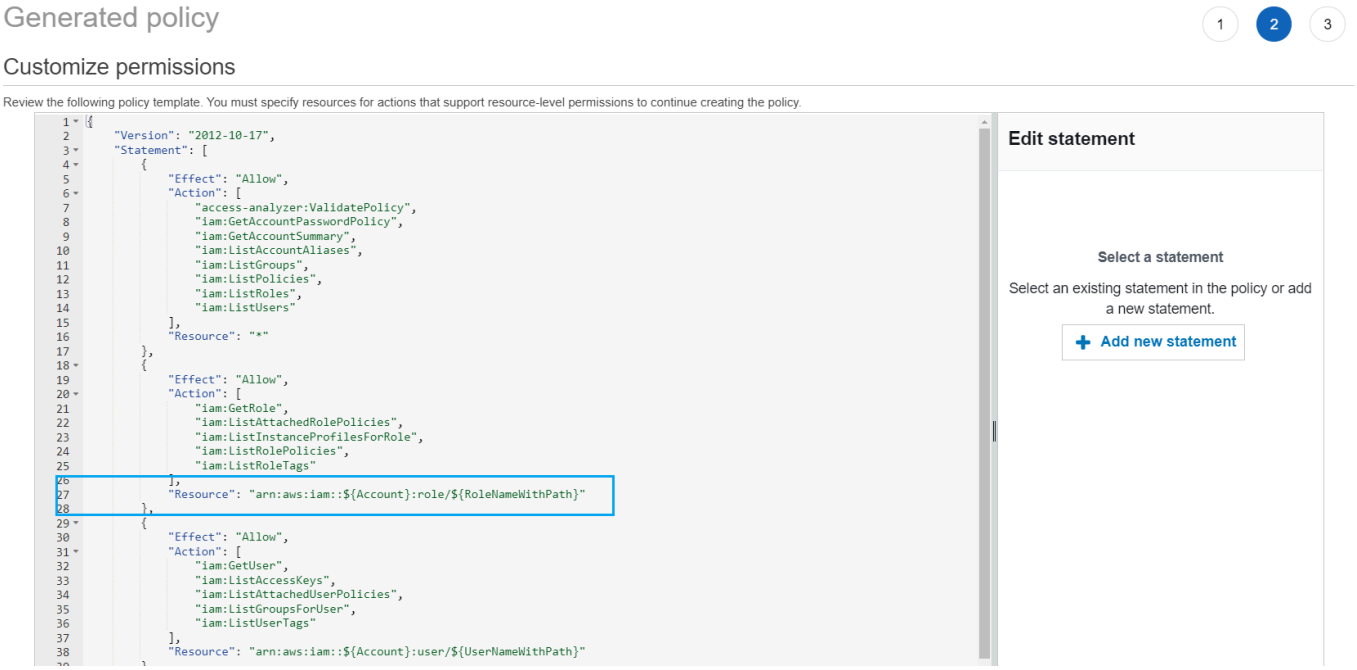
3단계: 생성된 정책 추가 사용자 지정

권한을 추가 또는 제거하거나 리소스를 지정하여 정책을 추가로 사용자 지정할 수 있습니다.

생성된 정책 사용자 지정

1. 정책 템플릿을 업데이트합니다. 정책 템플릿에는 다음 그림과 같이 리소스 수준 권한을 지원하는 작업에 대한 리소스 ARN 자리 표시자가 포함되어 있습니다. 리소스 수준 권한이란 사용자가 작업을 수행할 수 있는 리소스를 지정하는 기능을 말합니다. 리소스 수준 권한을 지원하는 작업에 대해서는 ARN을 사용하여 정책에서 개별 리소스를 지정하는 것이 좋습니다. 자리 표시자 리소스 ARN을 사용 사례에 맞는 유효한 리소스 ARN으로 바꿀 수 있습니다.

   작업이 리소스 수준 권한을 지원하지 않는 경우 와일드카드(*)를 사용하여 모든 리소스가 작업의 영향을 받을 수 있도록 지정해야 합니다. 어떤 AWS 서비스가 리소스 수준 권한을 지원하는지 알아보려면 IAM으로 작업하는 AWS 서비스를 참조하세요. 각 서비스의 작업 목록과 어떤 작업이 리소스 수준 권한을 지원하는지에 대한 자세한 내용은 AWS 서비스의 작업, 리소스 및 조건 키를 참조하세요.



2. (선택 사항) 템플릿에서 JSON 정책 설명을 추가, 수정 또는 제거합니다. JSON 정책 작성에 대한 자세한 내용은 IAM 정책 생성(콘솔)을 참조하세요.
3. 정책 템플릿을 사용자 지정했으면 다음 옵션을 사용할 수 있습니다.

- (선택 사항) 템플릿에서 JSON을 복사하여 생성된 정책 페이지 외부에서 별도로 사용할 수 있습니다. 예를 들어 JSON을 사용하여 다른 계정에 정책을 생성하려는 경우 템플릿의 정책이 JSON 정책에 대한 6,144자 제한을 초과하면 정책이 여러 정책으로 분할됩니다.
- 다음(Next)을 선택하여 동일한 계정에서 관리형 정책을 검토하고 생성합니다.

4단계: 관리형 정책 검토 및 생성

IAM 정책을 생성하고 연결할 수 있는 권한이 있는 경우 생성된 정책에서 관리형 정책을 생성할 수 있습니다. 그런 다음 정책을 계정의 사용자 또는 역할에 연결할 수 있습니다.

정책 검토 및 생성

1. 관리형 정책 검토 및 생성 페이지에서 생성하려는 정책의 이름(Name)과 설명(Description)(선택 사항)을 입력합니다.
2. (선택 사항) 요약(Summary) 섹션에서 정책에 포함될 사용 권한을 검토할 수 있습니다.
3. (선택 사항) 태그를 키 값 페어로 연결하여 메타데이터를 정책에 추가합니다. IAM에서 태그를 사용하는 방법에 대한 자세한 내용은 IAM 리소스 태깅을 참조하세요.
4. 작업을 마쳤으면 다음 중 하나를 수행합니다.
   - 정책을 생성하는 데 사용된 역할에 새 정책을 직접 연결할 수 있습니다. 이렇게 하려면 페이지 하단에서 ***YourRoleName***에 정책 연결 옆에 있는 확인란을 선택합니다. 그런 다음 정책 생성 및 연결(Create and attach policy)을 선택합니다.
   - 그렇지 않은 경우 정책 생성(Create policy)을 선택합니다. IAM 콘솔의 정책 탐색 창의 정책 목록에서 생성한 정책을 찾을 수 있습니다.
5. 생성한 정책을 계정의 엔터티에 연결할 수 있습니다. 정책을 연결한 후, 지나치게 광범위한 기타 정책이 엔터티에 연결될 수 있는 경우 이를 제거할 수 있습니다. 관리형 정책을 연결하는 방법은 IAM 자격 증명 권한 추가(콘솔)를 참조하세요.

다른 계정의 AWS CloudTrail 데이터를 사용하여 정책 생성

중앙 계정에 데이터를 저장하는 CloudTrail 추적을 생성하여 관리 활동을 간소화할 수 있습니다. 예를 들어 AWS Organizations를 사용하여 해당 조직에서 모든 AWS 계정의 모든 이벤트를 로그하는 추적을 생성할 수 있습니다. 이 추적은 중앙 계정에 속합니다. CloudTrail 로그 데이터가 저장되는 계정과 다른 계정의 사용자 또는 역할을 위한 정책을 생성하려면 크로스 계정 액세스 권한을 부여해야 합니다. 이를 위해서는 IAM Access Analyzer에 CloudTrail 로그에 대한 권한을 부여하는 역할 및 버킷 정책이

모두 필요합니다. Organizations 추적 생성에 대한 자세한 내용은 조직에 대한 추적 생성을 참조하세요.

이 예에서는 계정 A의 사용자 또는 역할에 대한 정책을 생성한다고 가정합니다. 계정 A의 CloudTrail 추적은 계정 B의 버킷에 CloudTrail 로그를 저장합니다. 정책을 생성하려면 먼저 다음을 업데이트해야 합니다.

1. 기존 역할을 선택하거나 계정 B의 버킷(CloudTrail 로그가 저장되는 위치)에 대한 액세스 권한을 IAM Access Analyzer에 부여하는 새 서비스 역할을 생성합니다.
2. IAM Access Analyzer가 버킷의 객체에 액세스할 수 있도록 계정 B에서 Amazon S3 버킷 객체 소유권 및 버킷 권한 정책을 확인합니다.

1단계: 크로스 계정 액세스를 위한 역할 선택 또는 생성

- 계정에 필요한 권한이 있는 역할이 이미 있는 경우 정책 생성(Generate policy) 화면에서 기존 역할 사용(Use an existing role)이 미리 선택되어 있습니다. 그렇지 않으면 새 서비스 역할 생성 및 사용(Create and use a new service role)을 선택합니다. 새 역할은 계정 B의 CloudTrail 로그에 대한 액세스 권한을 IAM Access Analyzer에 부여하는 데 사용됩니다.

2단계: 계정 B의 Amazon S3 버킷 구성 확인 또는 업데이트

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/s3/에서 Amazon S3 콘솔을 엽니다.
2. 버킷(Buckets) 목록에서 CloudTrail 추적 로그가 저장된 버킷의 이름을 선택합니다.
3. 권한(Permissions) 탭을 클릭하고 객체 소유권(Object ownership) 섹션으로 이동합니다.

   Amazon S3 객체 소유권 버킷 설정을 사용하여 버킷에 업로드하는 객체의 소유권을 제어합니다. 기본적으로 다른 AWS 계정이 버킷에 객체를 업로드할 때 업로드하는 계정이 객체를 소유합니다. 정책을 생성하려면 버킷 소유자가 버킷의 모든 객체를 소유해야 합니다. ACL 사용 사례에 따라 버킷의 객체 소유권(Object Ownership) 설정을 변경해야 할 수도 있습니다. 객체 소유권(Object Ownership)을 다음 옵션 중 하나로 설정합니다.

   - 버킷 소유자 적용(Bucket owner enforced)(권장)
   - 버킷 소유자 선호(Bucket owner preferred)

> **Important**
>
> 정책을 성공적으로 생성하려면 버킷의 객체를 버킷 소유자가 소유해야 합니다. 버킷 소유자 선호(Bucket owner preferred)를 사용하도록 선택하는 경우 객체 소유권이 변경된 후 해당 기간 동안만 정책을 생성할 수 있습니다.

Amazon S3의 객체 소유권에 대해 자세히 알아보려면 Amazon S3 사용 설명서의 객체 소유권 제어 및 버킷에 대해 ACL 사용 중지를 참조하세요.

4. 계정 B의 Amazon S3 버킷 정책에 권한을 추가하여 계정 A의 역할에 대한 액세스를 허용합니다.

   다음 정책 예에서는 `amzn-s3-demo-bucket`이라는 버킷에 대한 `ListBucket` 및 `GetObject`를 허용합니다. 버킷에 액세스하는 역할이 조직의 계정에 속하고 이름이 `AccessAnalyzerMonitorServiceRole`로 시작하는 경우 액세스가 허용됩니다. `aws:PrincipalArn`을 `Resource` 요소의 `Condition`으로 사용하면 역할이 계정 A에 속한 경우에만 계정의 활동에 액세스할 수 있습니다. `amzn-s3-demo-bucket`을 실제 버킷 이름으로, `optional-prefix`를 버킷에 대한 선택적 접두사로, `organization-id`를 조직 ID로 바꿀 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PolicyGenerationBucketPolicy",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/optional-prefix/AWSLogs/organization-id/
${aws:PrincipalAccount}/*"
      ],
      "Condition": {
        "StringEquals": {
```

```
          "aws:PrincipalOrgID": "organization-id"
        },
        "StringLike": {
          "aws:PrincipalArn": "arn:aws:iam::${aws:PrincipalAccount}:role/service-
role/AccessAnalyzerMonitorServiceRole*"
        }
      }
    }
  ]
}
```

5. AWS KMS를 사용하여 로그를 암호화하는 경우, 다음 정책 예시에서처럼 키를 사용할 수 있는 액세스 권한을 IAM Access Analyzer에 부여하도록 CloudTrail 로그를 저장하는 계정에서 AWS KMS 키 정책을 업데이트합니다. `CROSS_ACCOUNT_ORG_TRAIL_FULL_ARN`을 추적의 ARN으로, `organization-id`를 조직 ID로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "kms:Decrypt",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:aws:cloudtrail:arn":
 "CROSS_ACCOUNT_ORG_TRAIL_FULL_ARN",
          "aws:PrincipalOrgID": "organization-id"
        },
        "StringLike": {
          "kms:ViaService": [
            "access-analyzer.*.amazonaws.com",
            "s3.*.amazonaws.com"
          ]
          "aws:PrincipalArn": "arn:aws:iam::${aws:PrincipalAccount}:role/service-
role/AccessAnalyzerMonitorServiceRole*"
        }
      }
    }
  ]
```

```
}
```

CloudTrail 활동을 기반으로 정책 생성(AWS CLI)

다음 명령을 사용하여 AWS CLI을(를) 사용하는 정책을 생성할 수 있습니다.

정책 생성

- aws accessanalyzer start-policy-generation

생성된 정책 보기

- aws accessanalyzer get-generated-policy

정책 생성 요청 취소

- aws accessanalyzer cancel-policy-generation

정책 생성 요청 목록 보기

- aws accessanalyzer list-policy-generations

CloudTrail 활동을 기반으로 정책 생성(AWS API)

다음 작업을 사용하여 AWS API를 사용하는 정책을 생성할 수 있습니다.

정책 생성

- StartPolicyGeneration

생성된 정책 보기

- GetGeneratedPolicy

정책 생성 요청 취소

- CancelPolicyGeneration

정책 생성 요청 목록 보기

- ListPolicyGenerations

IAM Access Analyzer 정책 생성 서비스

다음 테이블에는 IAM Access Analyzer에서 작업 수준 정보가 포함된 정책을 생성하는 AWS 서비스가 나와 있습니다. 각 서비스의 작업 목록은 서비스 승인 참조의 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

서비스	서비스 접두사
AWS Identity and Access Management Access Analyzer	access-analyzer
AWS Account Management	account
AWS Certificate Manager	acm
Amazon Managed Workflows for Apache Airflow	airflow
AWS Amplify	amplify
AWS Amplify UI 빌더	amplifyuibuilder
Amazon AppIntegrations	app-integrations
AWS AppConfig	appconfig
Amazon AppFlow	appflow
AWS Application Cost Profiler	application-cost-profiler
Amazon CloudWatch Application Insights	applicationinsights
AWS App Mesh	appmesh
Amazon AppStream 2.0	appstream
AWS AppSync	appsync

서비스	서비스 접두사
Amazon Managed Service for Prometheus	aps
Amazon Athena	athena
AWS Audit Manager	auditmanager
AWS Auto Scaling	Auto Scaling
AWS Marketplace	aws-marketplace
AWS Backup	백업
AWS Batch	일괄
Amazon Braket	braket
AWS Budgets	예산
AWS Cloud9	cloud9
AWS CloudFormation	cloudformation
Amazon CloudFront	cloudfront
AWS CloudHSM	cloudhsm
Amazon CloudSearch	cloudsearch
AWS CloudTrail	cloudtrail
Amazon CloudWatch	cloudwatch
AWS CodeArtifact	codeartifact
AWS CodeDeploy	codedeploy
Amazon CodeGuru Profiler	codeguru-profiler
Amazon CodeGuru Reviewer	codeguru-reviewer

서비스	서비스 접두사
AWS CodePipeline	CodePipeline
AWS CodeStar	codestar
AWS CodeStar 알림	codestar-notifications
Amazon Cognito 자격 증명	cognito-identity
Amazon Cognito 사용자 풀	cognito-idp
Amazon Cognito Sync	cognito-sync
Amazon Comprehend Medical	comprehendmedical
AWS Compute Optimizer	compute-optimizer
AWS Config	config
Amazon Connect	연결
AWS Cost and Usage Report	cur
AWS Glue DataBrew	databrew
AWS Data Exchange	dataexchange
AWS Data Pipeline	datapipeline
DynamoDB Accelerator	dax
AWS Device Farm	devicefarm
Amazon DevOps Guru	devops-guru
AWS Direct Connect	directconnect
Amazon Data Lifecycle Manager	dlm

서비스	서비스 접두사
AWS Database Migration Service	DMS
Amazon DocumentDB Elastic Clusters	docdb-elastic
AWS Directory Service	ds
Amazon DynamoDB	dynamodb
Amazon Elastic Block Store	ebs
Amazon Elastic Compute Cloud	EC2
Amazon Elastic 컨테이너 레지스트리	ecr
Amazon Elastic Container Registry Public	ecr-public
Amazon Elastic Container Service	ecs
Amazon Elastic Kubernetes Service	eks
Amazon Elastic Inference	elastic-inference
Amazon ElastiCache	elasticache
AWS Elastic Beanstalk	elasticbeanstalk
Amazon Elastic File System	elasticfilesystem
Elastic Load Balancing	elasticloadbalancing
Amazon Elastic Transcoder	elastictranscoder
Amazon EMR on EKS(EMR 컨테이너)	emr-containers
Amazon EMR Serverless	emr-serverless
Amazon OpenSearch Service	es
Amazon EventBridge	이벤트

서비스	서비스 접두사
Amazon CloudWatch Evidently	evidently
Amazon FinSpace	finspace
Amazon Data Firehose	firehose
AWS Fault Injection Service	fis
AWS Firewall Manager	fms
Amazon Fraud Detector	frauddetector
Amazon FSx	fsx
Amazon GameLift	gamelift
Amazon Location Service	geo
Amazon S3 Glacier	glacier
Amazon Managed Grafana	grafana
AWS IoT Greengrass	greengrass
AWS Ground Station	groundstation
Amazon GuardDuty	guardduty
AWS HealthLake	healthlake
Amazon Honeycode	honeycode
AWS Identity and Access Management	iam
AWS 자격 증명 스토어	identitystore
EC2 Image Builder	imagebuilder
Amazon Inspector Classic	inspector

서비스	서비스 접두사
Amazon Inspector	inspector2
AWS IoT	iot
AWS IoT Analytics	iotanalytics
AWS IoT Core Device Advisor	iotdeviceadvisor
AWS IoT Events	iotevents
AWS IoT Fleet Hub	iotfleethub
AWS IoT SiteWise	iotsitewise
AWS IoT TwinMaker	iottwinmaker
AWS IoT Wireless	iotwireless
Amazon Interactive Video Service	ivs
Amazon Interactive Video Service Chat	ivschat
Amazon Managed Streaming for Apache Kafka	kafka
Amazon Managed Streaming for Kafka Connect	kafkaconnect
Amazon Kendra	kendra
Amazon Kinesis	kinesis
Amazon Kinesis Analytics V2	kinesisanalytics
AWS Key Management Service	kms
AWS Lambda	lambda
Amazon Lex	lex
AWS License Manager Linux Subscriptions Manager	license-manager-linux-subscriptions

서비스	서비스 접두사
Amazon Lightsail	lightsail
Amazon CloudWatch Logs	로그
Amazon Lookout for Equipment	lookoutequipment
Amazon Lookout for Metrics	lookoutmetrics
Amazon Lookout for Vision	lookoutvision
AWS Mainframe Modernization	m2
Amazon Managed Blockchain	managedblockchain
AWS Elemental MediaConnect	mediaconnect
AWS Elemental MediaConvert	mediaconvert
AWS Elemental MediaLive	medialive
AWS Elemental MediaStore	mediastore
AWS Elemental MediaTailor	mediatailor
Amazon MemoryDB	memorydb
AWS Application Migration Service	mgn
AWS Migration Hub	mgh
AWS Migration Hub Strategy Recommendations	migrationhub-strategy
Amazon Pinpoint	mobiletargeting
Amazon MQ	mq
AWS Network Manager	networkmanager

서비스	서비스 접두사
Amazon Nimble Studio	nimble
AWS HealthOmics	omics
AWS OpsWorks	opsworks
AWS OpsWorks CM	opsworks-cm
AWS Outposts	outposts
AWS Organizations	organizations
AWS Panorama	panorama
AWS 성능 개선 도우미	pi
Amazon EventBridge 파이프	pipes
Amazon Polly	polly
Amazon Connect Customer Profiles	profile
Amazon QLDB	qldb
AWS Resource Access Manager	ram
AWS 휴지통	rbin
Amazon Relational Database Service	RDS
Amazon Redshift	redshift
Amazon Redshift 데이터 API	redshift-data
AWS Migration Hub Refactor Spaces	refactor-spaces
Amazon Rekognition	rekognition
AWS Resilience Hub	resiliencehub

서비스	서비스 접두사
AWS 리소스 탐색기	resource-explorer-2
AWS Resource Groups	resource-groups
AWS RoboMaker	robomaker
AWS Identity and Access Management Roles Anywhere	rolesanywhere
Amazon Route 53	route53
Amazon Route 53 Recovery Controls	route53-recovery-control-config
Amazon Route 53 Recovery Readiness	route53-recovery-readiness
Amazon Route 53 Resolver	route53resolver
AWS CloudWatch RUM	rum
Amazon Simple Storage Service(S3)	s3
Amazon S3 on Outposts	s3-outposts
Amazon SageMaker 지리 공간 기능	sagemaker-geospatial
Savings Plans	savingsplans
Amazon EventBridge 스키마	schemas
Amazon SimpleDB	sdb
AWS Secrets Manager	secretsmanager
AWS Security Hub	securityhub
Amazon Security Lake	securitylake

서비스	서비스 접두사
AWS Serverless Application Repository	serverlessrepo
AWS Service Catalog	servicecatalog
AWS Cloud Map	servicediscovery
Service Quotas	servicequotas
Amazon Simple Email Service	ses
AWS Shield	shield
AWS Signer	signer
AWS SimSpace Weaver	simspaceweaver
AWS Server Migration Service	sms
Amazon Pinpoint SMS 및 음성 서비스	sms-voice
AWS Snowball	snowball
Amazon Simple Queue Service	sqs
AWS Systems Manager	ssm
AWS Systems Manager Incident Manager	ssm-incidents
AWS Systems Manager for SAP	ssm-sap
AWS Step Functions	states
AWS Security Token Service	sts
Amazon Simple Workflow Service	swf
Amazon CloudWatch Synthetics	synthetics
AWS Resource Groups Tagging API	tag

서비스	서비스 접두사
Amazon Textract	textract
Amazon Timestream	timestream
AWS 통신 네트워크 빌더	tnb
Amazon Transcribe	transcribe
AWS Transfer Family	전송
Amazon Translate	translate
Amazon Connect Voice ID	voiceid
Amazon VPC Lattice	vpc-lattice
AWS WAFV2	wafv2
AWS Well-Architected Tool	wellarchitected
Amazon Connect Wisdom	wisdom
Amazon WorkLink	worklink
Amazon WorkSpaces	WorkSpaces
AWS X-Ray	xray

IAM Access Analyzer 할당량

IAM Access Analyzer에는 다음과 같은 할당량이 있습니다.

Resource	기본 할당량	최대 할당량
리전별 AWS 계정 당 분석기 유형에 따른 최대 계정 수준 분석기	1	1

Resource	기본 할당량	최대 할당량
리전별 AWS 계정 당 분석기 유형에 따른 최대 조직 수준 분석기	5	20[1]
분석기당 최대 아카이브 규칙 수	100 각 아카이브 규칙에 기준당 최대 20개의 값이 있을 수 있습니다.	1,000[1]
시간당 분석기별 최대 액세스 미리 보기 수	1,000	1,000
정책 생성별로 처리된 AWS CloudTrail 로그 파일 수	100,000건	100,000건
동시 정책 생성	1	1
정책 생성 AWS CloudTrail 데이터 크기	25GB	25GB
정책 생성 AWS CloudTrail 시간 범위	90일	90일

Resource	기본 할당량	최대 할당량
일일 정책 생성	아프리카(케이프타운): 5 아시아 태평양(홍콩): 5 유럽(밀라노): 5 중동(바레인): 5 기타 지원되는 모든 리전: 50 **Note** 취소된 정책 생성 요청은 일일 할당량에 적용됩니다.	아프리카(케이프타운): 5 아시아 태평양(홍콩): 5 유럽(밀라노): 5 중동(바레인): 5 기타 지원되는 모든 리전: 50

[1]일부 할당량은 Service Quotas를 사용하여 고객이 구성할 수 있습니다.

IAM 문제 해결

여기에 있는 정보를 사용하면 AWS Identity and Access Management(IAM)를 사용하여 작업할 때 발생하는 일반적인 문제를 진단하고 해결할 수 있습니다.

문제

내 AWS 계정에 로그인할 수 없음

올바른 자격 증명이 있고 올바른 방법을 사용하여 로그인하는지 확인합니다. 자세한 내용을 알아보려면 AWS 로그인 사용 설명서의 로그인 문제 해결을 참조하세요.

액세스 키를 분실했습니다

액세스 키는 다음 두 부분으로 구성됩니다.

- 액세스 키 식별자. 이 식별자는 비밀이 아니며 사용자 요약 페이지 등 IAM 콘솔에서 액세스 키가 나열되어 있는 곳 어디서나 확인할 수 있습니다.

- 보안 액세스 키. 액세스 키 페어를 처음 만들 때 제공됩니다. 암호와 마찬가지로 나중에 검색할 수 없습니다. 하지만 보안 액세스 키를 분실한 경우에는 새로운 액세스 키 페어를 생성해야 합니다. 이미 액세스 키의 최대 수를 가진 경우에는 기존 페어를 삭제해야만 다른 페어를 생성할 수 있습니다.

자세한 내용은 분실하거나 잊어버린 AWS 암호 또는 액세스 키 재설정 단원을 참조하십시오.

정책 변수가 작동하지 않습니다

정책 변수가 작동하지 않는 경우 다음 오류 중 하나가 발생한 것입니다.

Version 정책 요소의 날짜가 잘못되었습니다.

변수가 포함된 모든 정책에 버전 번호 `"Version": "2012-10-17"`이 있는지 확인하세요. 올바른 버전 번호가 없으면 평가 도중에 변수가 대체되지 않습니다. 대신 변수는 문자 그대로 평가됩니다. 최신 버전 번호를 포함할 때 변수를 포함하지 않은 정책이 계속 작동합니다.

`Version` 정책 요소는 정책 버전과 다릅니다. `Version` 정책 요소는 정책 내에서 사용되며 정책 언어의 버전을 정의합니다. 정책 버전은 IAM에서 고객 관리형 정책을 수정할 때 생성됩니다. 변경된 정책은 기존 정책을 덮어쓰지 않습니다. 대신 IAM에서 관리형 정책의 새 버전을 생성합니다. `Version` 정책 요소에 대한 자세한 정보는 IAM JSON 정책 요소: Version을 참조하세요. 정책 버전에 대한 자세한 정보는 the section called “IAM 정책 버전 관리” 섹션을 참조하세요.

변수 문자의 대/소문자가 잘못되었습니다.

정책 변수가 올바른지 확인합니다. 자세한 내용은 IAM 정책 요소: 변수 및 태그 섹션을 참조하세요.

변경 사항이 매번 즉시 표시되는 것은 아닙니다

사용자들이 전세계 데이터 센터의 컴퓨터들을 통해 액세스하는 서비스인 IAM은 최종 일관성이라고 하는 분산 컴퓨팅 모델을 사용합니다. 속성 기반 액세스 제어(ABAC) 태그를 포함하여 IAM(또는 기타 AWS 서비스)에서 변경한 사항은 가능한 모든 엔드포인트에서 보게 될 때까지는 시간이 걸립니다. 약간의 지연은 서버에서 서버로, 복제 영역에서 복제 영역으로, 리전에서 리전으로 데이터를 보내는 데 걸리는 시간으로 인해 발생합니다. 또한 IAM은 성능 향상을 위해 캐싱을 사용하지만, 어떤 경우에는 이로 인해 시간이 더 걸릴 수 있습니다. 이러한 변화는 이전에 캐싱된 데이터가 끝날 때까지 가시화되지 않을 수 있습니다.

이러한 잠재적 지연을 고려하도록 전역 애플리케이션을 설계해야 합니다. 한 위치에서 변경한 내용이 다른 위치에서 즉시 보이지 않을 때조차도 예상대로 작동하는지 확인합니다. 그러한 변경 사항에는 사용자, 그룹, 역할 또는 정책을 만들거나 업데이트한 것이 포함됩니다. 이러한 IAM 변경 사항을 애플리

케이션의 중요한 고가용성 코드 경로에 포함시키지 않는 것이 좋습니다. 대신 자주 실행하지 않는 별도의 초기화 루틴이나 설정 루틴에서 IAM을 변경하세요. 또한 프로덕션 워크플로우에서 변경 사항을 적용하기 전에 변경 사항이 전파되었는지 확인하세요.

이로 인해 일부 다른 AWS 서비스가 받게 되는 영향에 대한 자세한 정보는 다음 자료를 참조하세요.

- Amazon DynamoDB: DynamoDB 개발자 안내서의 읽기 일관성, Amazon DynamoDB 개발자 안내서의 읽기 일관성.
- Amazon EC2: Amazon EC2 API 참조의 Amazon EC2 최종 일관성
- Amazon EMR: AWS 빅 데이터 블로그의 Ensuring Consistency When Using Amazon S3 and Amazon EMR for ETL Workflows.
- Amazon Redshift: Amazon Redshift 데이터베이스 개발자 안내서의 데이터 일관성 관리
- Amazon S3: Amazon Simple Storage Service 사용 설명서의 Amazon S3 데이터 일관성 모델

iam:DeleteVirtualMFADevice를 수행할 권한이 없음

본인 또는 다른 사용자를 위해 가상 MFA 디바이스를 할당하거나 제거하려고 하면 다음과 같은 오류가 발생할 수 있습니다.

```
User: arn:aws:iam::123456789012:user/Diego is not authorized to
 perform: iam:DeleteVirtualMFADevice on resource: arn:aws:iam::123456789012:mfa/Diego
 with an explicit deny
```

이는 IAM 콘솔에서 이전에 다른 누군가가 가상 MFA 디바이스를 사용자에게 할당하기 시작했다가 프로세스를 취소한 경우 발생할 수 있습니다. 이를 통해 IAM에서 사용자에 대한 가상 MFA 디바이스가 생성되지만 사용자에게 할당하지는 않습니다. 동일한 디바이스 이름으로 새 가상 MFA 디바이스를 생성하려면 먼저 기존 가상 MFA 디바이스를 삭제합니다.

이 문제를 해결하려면 관리자가 정책 권한을 편집하지 않아야 합니다. 대신 관리자는 AWS CLI 또는 AWS API를 사용하여 할당되지 않은 기존 가상 MFA 디바이스를 삭제해야 합니다.

할당되지 않은 기존 가상 MFA 디바이스를 삭제하려면

1. 계정에서 가상 MFA 디바이스를 확인합니다.
   - AWS CLI: `aws iam list-virtual-mfa-devices`
   - AWS API: `ListVirtualMFADevices`
2. 응답에서 수정하려는 사용자의 가상 MFA 디바이스 ARN을 찾습니다.

3. 가상 MFA 디바이스를 삭제합니다.

   - AWS CLI: `aws iam delete-virtual-mfa-device`
   - AWS API: `DeleteVirtualMFADevice`

IAM 사용자를 안전하게 생성하려면 어떻게 해야 하나요?

AWS에 액세스해야 하는 직원이 있는 경우 IAM 사용자를 생성하거나 IAM Identity Center를 사용하여 인증하도록 선택할 수 있습니다. IAM을 사용하는 경우 AWS에서는 IAM 사용자를 생성하고 해당 직원에게 자격 증명을 안전하게 전달할 것을 권장합니다. 실제로 직원 옆에 있지 않은 경우 보안 워크플로를 사용하여 직원에게 자격 증명을 전달합니다.

다음 보안 워크플로를 사용하여 IAM에서 새 사용자 생성:

1. AWS Management Console을 사용하여 새 사용자를 생성합니다. 생성된 암호로 AWS Management Console 액세스 권한을 부여하도록 선택합니다. 필요한 경우 다음 로그인 시 사용자가 새 암호를 생성해야 함(Users must create a new password at next sign-in) 확인란을 선택합니다. 사용자가 암호를 변경하기 전까지 사용자에게 권한 정책을 추가하지 마십시오.
2. 사용자가 추가되면 새 사용자의 로그인 URL, 사용자 이름 및 암호를 복사합니다. 암호를 보려면 표시(Show)를 선택합니다.
3. 이메일, 채팅 또는 티켓 시스템과 같은 회사의 보안 통신 방법을 사용하여 직원에게 암호를 보냅니다. 사용자에게 IAM 사용자 콘솔 링크와 해당하는 사용자 이름을 별도로 제공합니다. 직원에게 권한을 부여하기 전에 성공적으로 로그인할 수 있는지 확인하도록 직원에게 알립니다.
4. 직원이 확인하면 필요한 권한을 추가합니다. 보안 모범 사례로 MFA를 사용하여 자격 증명을 관리할 것을 사용자에게 요구하는 정책을 추가합니다. 정책에 대한 예는 AWS: MFA 인증 IAM 사용자가 보안 인증 페이지에서 자신의 보안 인증을 관리할 수 있도록 허용 섹션을 참조하세요.

추가 리소스

다음 리소스는 AWS로 직접 작업할 때 문제 해결에 도움이 될 수 있습니다.

- AWS CloudTrail 사용 설명서 AWS CloudTrail을 사용하여 AWS에 대한 API 호출 기록을 추적하고 로그 파일에 해당 정보를 저장합니다. 이를 통해 계정의 리소스에 액세스한 사용자와 계정, 호출이 발생한 시기, 요청된 작업 등을 확인할 수 있습니다. 자세한 내용은 AWS CloudTrail을 사용하여 IAM 및 AWS STS API 호출 로깅 단원을 참조하십시오.
- AWS 지식 센터 – 문제를 해결할 때 도움이 되는 FAQ와 기타 리소스 링크를 찾을 수 있습니다.

- AWS 지원 센터 – 기술 지원을 받을 수 있습니다.
- AWS 프리미엄 기술 지원 센터 – 프리미엄 기술 지원을 받을 수 있습니다.

액세스 거부 오류 메시지 문제 해결

다음 정보는 AWS Identity and Access Management에서 액세스 거부 오류를 식별, 진단 및 해결하는 데 도움이 될 수 있습니다. 액세스 거부 오류는 AWS가 권한 부여 요청을 명시적 또는 암시적으로 거부할 때 나타납니다.

- 명시적 거부는 정책에 특정 AWS 작업에 대한 `Deny` 문이 포함되어 있을 때 발생합니다.
- 적용 가능한 `Deny` 문이 없고 적용 가능한 `Allow` 문도 없다면 암시적 거부가 발생합니다. IAM 정책은 기본적으로 IAM 보안 주체를 거부하므로 정책은 보안 주체가 작업을 수행하도록 명시적으로 허용해야 합니다. 그렇지 않으면 정책이 암시적으로 액세스를 거부합니다. 자세한 내용은 명시적 거부와 묵시적 거부 차이 단원을 참조하십시오.

서비스 또는 리소스에 요청하는 경우 요청에 여러 정책이 적용될 수 있습니다. 오류 메시지에 지정된 정책 외에 적용 가능한 모든 정책을 검토하세요.

- 동일한 정책 유형의 여러 정책에서 요청을 거부하는 경우 액세스 거부 오류 메시지에서 평가된 정책 수를 지정하지 않습니다.
- 여러 정책 유형이 권한 부여 요청을 거부하는 경우 AWS는 오류 메시지에 이러한 정책 유형 중 하나만 포함됩니다.

> **Important**
>
> AWS에 로그인하는 데 문제가 있나요? 사용자 유형에 맞는 올바른 AWS 로그인 페이지에 있는지 확인합니다. AWS 계정 루트 사용자(계정 소유자)인 경우 AWS 계정을 생성할 때 설정한 보안 인증을 사용하여 AWS에 로그인할 수 있습니다. IAM 사용자인 경우 계정 관리자가 AWS 로그인 자격 증명을 제공할 수 있습니다. 지원을 요청해야 하는 경우 이 페이지의 피드백 링크를 사용하지 않도록 하세요. 양식은 AWS Support이 아닌 AWS 설명서 팀에서 접수합니다. 대신 Contact Us(문의처) 페이지에서 Still unable to log into your AWS account(여전히 계정에 로그인할 수 없음)을 선택한 다음 사용 가능한 지원 옵션 중 하나를 선택합니다.

AWS 서비스에 요청하면 '액세스 거부됨' 오류 메시지가 표시됨

- 오류 메시지에 액세스 거부의 원인이 되는 정책 유형이 포함되어 있는지 확인합니다. 예를 들어, 오류 메시지에 서비스 제어 정책(SCP)으로 인해 액세스가 거부되었다고 표시되면 SCP 문제를 해결하는 데 집중할 수 있습니다. 정책 유형을 식별한 후에는 해당 정책 유형에서 거부 문이나 누락된 허용 작업이 있는지 확인할 수 있습니다. 오류 메시지에 액세스 거부의 원인이 되는 정책 유형이 언급되지 않은 경우 이 섹션의 나머지 지침을 사용하여 추가 문제를 해결합니다.
- 요청한 작업 및 리소스를 호출할 자격 증명 기반 정책 권한이 있는지 확인합니다. 조건이 설정된 경우 요청을 보낼 때 그러한 조건 또한 충족해야 합니다. IAM 사용자, 그룹 또는 역할에 대한 정책을 보거나 수정하는 방법에 대한 자세한 내용은 IAM 정책 관리 섹션을 참조하세요.
- AWS Management Console에서 작업을 수행할 권한이 없다는 메시지를 반환하는 경우 관리자에게 문의하여 도움을 받아야 합니다. 관리자가 로그인 자격 증명 또는 로그인 링크를 제공했습니다.

  다음 예제 오류는 `mateojackson` IAM 사용자가 콘솔을 사용하여 가상 *`my-example-widget`* 리소스에 대한 세부 정보를 보려고 하지만 가상 `widgets:`*`GetWidget`* 권한이 없을 때 발생합니다.

  ```
  User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
   widgets:GetWidget on resource: my-example-widget
  ```

  이 경우 Mateo는 *`my-example-widget`* 작업을 사용하여 `widgets:`*`GetWidget`* 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.
- 리소스 기반 정책을 지원하는 서비스(예: Amazon S3, Amazon SNS 또는 Amazon SQS)에 액세스하려 하나요? 그러한 경우 정책에서 사용자를 보안 주체로 지정하고 액세스 권한을 부여하는지 확인하세요. 자신의 계정 내에서 서비스를 요청하는 경우 자격 증명 기반 정책이나 리소스 기반 정책에서 요청자에게 권한을 부여할 수 있습니다. 다른 계정에서 서비스를 요청하는 경우 자격 증명 기반 정책 및 리소스 기반 정책 모두에서 요청자에게 권한을 부여해야 합니다. 리소스 기반 정책을 지원하는 서비스를 보려면 AWS IAM으로 작업하는 서비스 섹션을 참조하세요.
- 정책에 키 값 페어가 있는 조건이 포함된 경우 이를 주의하여 검토하세요. `aws:RequestTag/`*`tag-key`* 전역 조건 키, AWS KMS `kms:EncryptionContext:`*`encryption_context_key`* 및 여러 서비스에서 지원하는 `ResourceTag/`*`tag-key`* 조건 키가 그 예입니다. 키 이름이 여러 개의 결과와 일치하지 않도록 하세요. 조건 키 이름이 대/소문자를 구분하지 않으므로 이름이 `foo`인 키를 검사하는 조건은 `foo`, `Foo` 또는 `F00`와 일치합니다. 대/소문자로만 구분되는 키 이름을 가진 여러 키 값 페어가 요청에 포함된 경우 액세스가 예기치 않게 거부될 수 있습니다. 자세한 내용은 IAM JSON 정책 요소: Condition 단원을 참조하십시오.
- 권한 경계가 있다면, 권한 경계에 사용된 정책이 요청을 허용하는지 확인합니다. 자격 증명 기반 정책에서는 요청이 허용되지만 권한 경계에서는 허용되지 않는 경우 요청이 거부됩니다. 이 권한 경계

는 IAM 보안 주체(사용자나 역할)에 부여할 수 있는 최대 권한을 제어합니다. 리소스 기반 정책은 권한 경계에 제한을 받지 않습니다. 권한 경계는 일반적이지 않습니다. AWS가 정책을 평가하는 방식에 대한 자세한 정보는 정책 평가 로직 섹션을 참조하세요.

- (AWS SDK를 사용하지 않고) 요청에 수동으로 서명할 경우, 요청에 올바르게 서명했는지 확인합니다.

임시 보안 자격 증명으로 요청하면 "액세스 거부"가 발생합니다

- 우선, 임시 자격 증명과 무관한 이유로 액세스가 거부되지는 않았는지 확인합니다. 자세한 내용은 AWS 서비스에 요청하면 '액세스 거부됨' 오류 메시지가 표시됨 단원을 참조하십시오.
- AWS IAM으로 작업하는 서비스 단원을 참조하여 서비스에서 임시 보안 자격 증명을 허용하는지 확인합니다.
- 요청에 올바르게 서명했고 요청이 잘 구성되었는지 확인합니다. 자세한 정보는 도구 키트 문서 또는 AWS 리소스에서 임시 자격 증명 사용 섹션을 참조하세요.
- 임시 보안 자격 증명이 만료되지 않았는지 확인합니다. 자세한 내용은 IAM의 임시 보안 자격 증명 단원을 참조하십시오.
- IAM 사용자 또는 역할 권한이 올바른지 확인합니다. 임시 보안 자격 증명에 대한 권한은 IAM 사용자 또는 역할에서 파생됩니다. 결과적으로 위임한 역할(임시 자격 증명이 제공됨)에 부여된 권한으로 제한됩니다. 임시 보안 자격 증명의 권한이 결정되는 방법에 대한 자세한 정보는 사용자 임시 보안 자격 증명에 대한 권한 제어 섹션을 참조하세요.
- 역할을 수임한 경우 역할 세션이 세션 정책에 의해 제한되었을 수 있습니다. AWS STS 사용을 통해 프로그래밍 방식으로 임시 보안 자격 증명을 요청한 경우 인라인 또는 관리형 세션 정책을 전달할 수 있습니다. 세션 정책은 역할 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. `Policy` 파라미터를 사용하여 단일 JSON 인라인 세션 정책 문서를 전달할 수 있습니다. `PolicyArns` 파라미터를 사용하여 최대 10개까지 관리형 세션 정책을 지정할 수 있습니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 자격 증명 기반 정책의 교집합과 세션 정책입니다. 또는 관리자 또는 사용자 프로그램에서 임시 자격 증명을 제공한 경우 세션 정책에 포함되어 액세스를 제한했을 수 있습니다.
- 페더레이션 사용자인 경우 세션이 세션 정책에 의해 제한되었을 수 있습니다. IAM 사용자로 AWS에 로그인하여 페더레이션 사용자가 된 후 연동 토큰을 요청합니다. 페더레이션 사용자에 대한 자세한 내용은 GetFederationToken - 사용자 지정 아이덴티티 브로커를 통한 페더레이션 섹션을 참조하세요. 사용자 또는 사용자의 자격 증명 브로커가 연동 토큰을 요청하는 동안 세션 정책을 전달한 경우 세션이 이러한 정책에 의해 제한됩니다. 결과적으로 얻는 세션의 권한은 IAM 사용자 자격 증명 기반 정책의 교집합과 세션 정책입니다. 세션 정책에 대한 자세한 정보는 세션 정책 섹션을 참조하세요.

- 역할을 사용하여 리소스 기반 정책이 있는 리소스에 액세스할 경우, 해당 정책에서 역할에 권한을 부여하는지 확인합니다. 예를 들어, 다음과 같은 정책에서는 계정 MyRole의 111122223333이 MyBucket에 액세스하도록 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "S3BucketPolicy",
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::111122223333:role/MyRole"]},
    "Action": ["s3:PutObject"],
    "Resource": ["arn:aws:s3:::MyBucket/*"]
  }]
}
```

액세스 거부 오류 메시지 예제

대부분의 액세스 거부 오류 메시지는 `User` *`user`* `is not authorized to perform` *`action`* `on` *`resource`* `because` *`context`* 형식으로 나타납니다. 이 예제에서 *`user`*는 액세스를 수신하지 않는 Amazon 리소스 이름(ARN)이고, *`action`*은 정책이 거부하는 서비스 작업이고, *`resource`*는 정책이 작용하는 리소스의 ARN입니다. *`context`* 필드는 정책이 액세스를 거부한 이유를 설명하는 정책 유형에 대한 추가 컨텍스트를 나타냅니다.

정책에 Deny 문이 포함되어 있어 정책이 액세스를 명시적으로 거부하는 경우 AWS는 액세스 거부 오류 메시지에 `with an explicit deny in a` *`type`* `policy`라는 구절을 포함합니다. 정책이 암시적으로 액세스를 거부하는 경우 AWS는 액세스 거부 오류 메시지에 `because no` *`type`* `policy allows the` *`action`* `action`이라는 구절을 포함합니다.

> **Note**
>
> 일부 AWS 서비스는 이 액세스 거부 오류 메시지 형식을 지원하지 않습니다. 액세스 거부 오류 메시지의 내용은 권한 부여를 요청하는 서비스에 따라 달라질 수 있습니다.

다음 예제는 다양한 유형의 액세스 거부 오류 메시지의 형식을 보여줍니다.

서비스 제어 정책으로 인한 액세스 거부 - 암묵적 거부

1. 서비스 제어 정책(SCP)에서 작업에 대한 누락된 `Allow` 문이 있는지 확인합니다. 다음 예제에서 작업은 `codecommit:ListRepositories`입니다.
2. `Allow` 문을 추가하여 SCP를 업데이트합니다. 자세한 내용은 AWS Organizations 사용 설명서의 SCP 업데이트를 참조하세요.

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no service control policy allows the
 codecommit:ListRespositories action
```

서비스 제어 정책으로 인한 액세스 거부 - 명시적 거부

1. 서비스 제어 정책(SCP)에서 작업에 대한 `Deny` 문이 있는지 확인합니다. 다음 예제에서 작업은 `codecommit:ListRepositories`입니다.
2. `Deny` 문을 제거하여 SCP를 업데이트합니다. 자세한 내용은 AWS Organizations 사용 설명서의 SCP 업데이트를 참조하세요.

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories with an explicit deny in a service control policy
```

VPC 엔드포인트 정책으로 인한 액세스 거부 - 암시적 거부

1. Virtual Private Cloud(VPC) 엔드포인트 정책에서 작업에 대한 누락된 `Allow` 문을 확인합니다. 다음 예제에서 작업은 `codecommit:ListRepositories`입니다.
2. `Allow` 문을 추가하여 VPC 엔드포인트 정책을 업데이트합니다. 자세한 정보는 AWS PrivateLink 안내서의 VPC 엔드포인트 정책 업데이트를 참조하세요.

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no VPC endpoint policy allows the
 codecommit:ListRepositories action
```

VPC 엔드포인트 정책으로 인한 액세스 거부 - 명시적 거부

1. Virtual Private Cloud(VPC) 엔드포인트 정책에서 작업에 대한 명시적 Deny 문을 확인합니다. 다음 예제에서 작업은 codedeploy:ListDeployments입니다.
2. Deny 문을 제거하여 VPC 엔드포인트 정책을 업데이트합니다. 자세한 정보는 AWS PrivateLink 안내서의 VPC 엔드포인트 정책 업데이트를 참조하세요.

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* with an explicit deny in a VPC endpoint policy
```

권한 경계로 인한 액세스 거부 - 암시적 거부

1. 권한 경계에서 작업에 대한 누락된 Allow 문을 확인합니다. 다음 예제에서 작업은 codedeploy:ListDeployments입니다.
2. IAM 정책에 Allow 문을 추가하여 권한 경계를 업데이트합니다. 자세한 내용은 IAM 엔터티의 권한 범위 및 IAM 정책 편집 단원을 참조하세요.

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* because no permissions boundary allows the
 codedeploy:ListDeployments action
```

권한 경계로 인한 액세스 거부 - 명시적 거부

1. 권한 경계에서 작업에 대한 명시적 Deny 문을 확인합니다. 다음 예제에서 작업은 sagemaker:ListModels입니다.
2. IAM 정책에서 Deny 문을 제거하여 권한 경계를 업데이트합니다. 자세한 내용은 IAM 엔터티의 권한 범위 및 IAM 정책 편집 단원을 참조하세요.

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
sagemaker:ListModels with an explicit deny in a permissions boundary
```

세션 정책으로 인한 액세스 거부 - 암시적 거부

1. 세션 정책에서 작업에 대한 누락된 `Allow` 문을 확인합니다. 다음 예제에서 작업은 `codecommit:ListRepositories`입니다.
2. `Allow` 문을 추가하여 세션 정책을 업데이트합니다. 자세한 정보는 세션 정책 및 IAM 정책 편집의 내용을 참조하세요.

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no session policy allows the
 codecommit:ListRepositories action
```

세션 정책으로 인한 액세스 거부 - 명시적 거부

1. 세션 정책에서 작업에 대한 명시적 `Deny` 문을 확인합니다. 다음 예제에서 작업은 `codedeploy:ListDeployments`입니다.
2. `Deny` 문을 제거하여 세션 정책을 업데이트합니다. 자세한 정보는 세션 정책 및 IAM 정책 편집의 내용을 참조하세요.

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* with an explicit deny in a sessions policy
```

리소스 기반 정책으로 인한 액세스 거부 - 암시적 거부

1. 리소스 기반 정책에서 작업에 대한 누락된 `Allow` 문을 확인합니다. 다음 예제에서 작업은 `secretsmanager:GetSecretValue`입니다.
2. `Allow` 문을 추가하여 정책을 업데이트합니다. 자세한 내용은 리소스 기반 정책 및 IAM 정책 편집의 내용을 참조하세요.

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
secretsmanager:GetSecretValue because no resource-based policy allows the
 secretsmanager:GetSecretValue action
```

리소스 기반 정책으로 인한 액세스 거부 - 명시적 거부

1. 리소스 기반 정책에서 작업에 대한 명시적 Deny 문을 확인합니다. 다음 예제에서 작업은 secretsmanager:GetSecretValue입니다.
2. Deny 문을 제거하여 정책을 업데이트합니다. 자세한 내용은 리소스 기반 정책 및 IAM 정책 편집의 내용을 참조하세요.

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
secretsmanager:GetSecretValue on resource: arn:aws:secretsmanager:us-
east-1:123456789012:secret:* with an explicit deny in a resource-based policy
```

역할 신뢰 정책으로 인한 액세스 거부 - 암시적 거부

1. 역할 신뢰 정책에서 작업에 대한 누락된 Allow 문을 확인합니다. 다음 예제에서 작업은 sts:AssumeRole입니다.
2. Allow 문을 추가하여 정책을 업데이트합니다. 자세한 내용은 리소스 기반 정책 및 IAM 정책 편집의 내용을 참조하세요.

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
sts:AssumeRole because no role trust policy allows the sts:AssumeRole action
```

역할 신뢰 정책으로 인한 액세스 거부 - 명시적 거부

1. 역할 신뢰 정책에서 작업에 대한 명시적 Deny 문을 확인합니다. 다음 예제에서 작업은 sts:AssumeRole입니다.
2. Deny 문을 제거하여 정책을 업데이트합니다. 자세한 내용은 리소스 기반 정책 및 IAM 정책 편집의 내용을 참조하세요.

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
sts:AssumeRole with an explicit deny in the role trust policy
```

자격 증명 기반 정책으로 인한 액세스 거부 - 암시적 거부

1. 자격 증명에 연결된 자격 증명 기반 정책에서 작업에 대한 누락된 Allow 문을 확인합니다. 다음 예제에서 작업은 역할 HR에 연결된 codecommit:ListRepositories입니다.

2. Allow 문을 추가하여 정책을 업데이트합니다. 자세한 내용은 자격 증명 기반 정책 및 IAM 정책 편집의 내용을 참조하세요.

```
User: arn:aws:iam::123456789012:role/HR is not authorized to perform:
codecommit:ListRepositories because no identity-based policy allows the
 codecommit:ListRepositories action
```

ID 기반 정책으로 인한 액세스 거부 - 명시적 거부

1. 자격 증명에 연결된 자격 증명 기반 정책에서 작업에 대한 명시적 Deny 문을 확인합니다. 다음 예제에서 작업은 역할 HR에 연결된 codedeploy:ListDeployments입니다.
2. Deny 문을 제거하여 정책을 업데이트합니다. 자세한 내용은 자격 증명 기반 정책 및 IAM 정책 편집의 내용을 참조하세요.

```
User: arn:aws:iam::123456789012:role/HR is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* with an explicit deny in an identity-based policy
```

다른 정책으로 인한 VPC 요청 실패 시 액세스 거부

1. 서비스 제어 정책(SCP)에서 작업에 대한 명시적 Deny 문을 확인합니다. 다음 예제에서 작업은 SNS:Publish입니다.
2. Deny 문을 제거하여 SCP를 업데이트합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 SCP 업데이트를 참조하세요.

```
User: arn:aws:sts::111122223333:assumed-role/role-name/role-session-name is not
 authorized to perform:
SNS:Publish on resource: arn:aws:sns:us-east-1:444455556666:role-name-2
with an explicit deny in a VPC endpoint policy transitively through a service control
 policy
```

루트 사용자 관련 문제 해결

여기의 정보를 사용하면 AWS 계정의 루트 사용자와 관련된 문제를 해결하는 데 도움이 됩니다.

계정 루트 사용자로 로그인했을 때 수행할 수 있을 것으로 예상한 작업을 수행할 수 없습니다.

계정은 AWS Organizations 조직의 구성원일 수 있습니다. 조직 관리자에게 계정의 권한을 제한할 서비스 제어 정책(SCP)이 있을 수 있습니다. SCP는 루트 사용자를 포함한 모든 사용자에게 영향을 미칩니다. 자세한 내용은 AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.

AWS 계정의 루트 사용자 암호를 잊음

루트 사용자인 경우 AWS 계정의 암호를 분실했거나 잊어버렸으면 암호를 재설정할 수 있습니다. AWS 계정을 생성하는 데 사용된 이메일 주소를 알고 있어야 하며 이메일 계정에 액세스할 수 있어야 합니다. 자세한 내용은 잊거나 분실한 루트 사용자 암호 재설정 단원을 참조하십시오.

내 AWS 계정의 이메일에 액세스할 수 없음

AWS 계정을 생성할 때 이메일 주소와 암호를 입력합니다. AWS 계정 루트 사용자의 자격 증명입니다. AWS 계정에 연결된 이메일 주소를 잘 모르는 경우 `@signin.aws` 또는 `@verify.signin.aws`에서 AWS 계정을 개설하는 데 사용되었을 수 있는 조직의 모든 이메일 주소로 보낸 메시지를 검색합니다.

이메일 주소를 알고 있지만 더 이상 이메일에 액세스할 수 없는 경우 먼저 이메일 액세스 복구를 시도하세요. 다음 옵션 중 하나를 사용하여 이메일에 대한 액세스 복구:

- 이메일 주소의 도메인을 소유한 경우 삭제된 이메일 주소를 복원할 수 있습니다. 아니면 이메일 계정에 대한 완전 포착(catch-all) 조건을 설정할 수 있습니다. catch-all은 더 이상 메일 서버에 존재하지 않는 이메일 주소로 보낸 메시지를 모두 수집하고 이를 다른 이메일 주소로 리디렉션합니다.
- 계정의 이메일 주소가 회사 이메일 시스템에 속한 경우라면 IT 시스템 관리자에게 문의하는 것이 좋습니다. 시스템 관리자가 이메일 주소에 대한 액세스 권한을 다시 받을 수 있도록 도와 줄 것입니다.

여전히 AWS 계정에 로그인할 수 없는 경우 문의하기에서 다른 지원 옵션을 찾을 수 있습니다.

IAM 정책 문제 해결

???정책은 자격 증명 또는 리소스에 연결될 때 해당 권한을 정의하는 AWS의 엔터티입니다. AWS는 사용자와 같은 보안 주체가 요청할 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 정책은 JSON 문서로 AWS에 저장되며 자격 증명 기반 정책으로 보안 주체에 연결되거나 리소스 기반 정책으로 리소스에 연결됩니다. 자격 증명 기반 정책을 IAM 그룹, 사용자 또는 역할과 같은 보안 주체(또는 자격 증명)에 연결할 수 있습니다. 자격 증명 기반 정책에는 AWS

관리형 정책, 고객 관리형 정책 및 인라인 정책이 포함됩니다. AWS Management Console에서 시각적 편집기 옵션 또는 JSON 편집기 옵션을 통해 고객 관리형 정책을 생성하고 편집할 수 있습니다. AWS Management Console에서 정책을 볼 때 정책에서 부여된 권한의 요약을 볼 수 있습니다. 시각적 편집기 및 정책 요약을 사용하여 IAM 정책을 관리하는 동안 발생한 일반 오류를 진단하고 해결할 수 있습니다.

모든 IAM 정책은 JavaScript Object Notation(JSON) 규칙으로 시작하는 구문을 사용하여 저장됩니다. 정책을 생성 또는 관리하기 위해 이 구문을 이해할 필요가 없습니다. AWS Management Console에서 시각적 편집기를 사용하여 정책을 생성하고 편집할 수 있습니다. IAM 정책의 JSON 구문에 대한 자세한 정보는 IAM JSON 정책 언어의 문법 섹션을 참조하세요.

IAM 정책 주제 문제 해결

- 시각적 편집기를 사용하여 문제 해결
  - 정책 재구성
  - 시각적 편집기에서 리소스 ARN 선택
  - 시각적 편집기에서 권한 거부
  - 시각적 편집기에서 여러 서비스 지정
  - 시각적 편집기에서 정책의 크기 줄이기
  - 시각적 편집기에서 인식할 수 없는 서비스, 작업 또는 리소스 유형 수정
- 정책 요약을 사용하여 문제 해결
  - 정책 요약 누락
  - 정책 요약에 인식할 수 없는 서비스, 작업 또는 리소스 유형 포함됨
  - 서비스가 IAM 정책 요약을 지원하지 않음
  - 정책이 필요한 권한을 부여하지 않음
- 정책 관리 문제 해결
  - IAM 계정에서 정책 연결 또는 분리
  - 작업 기반 IAM 자격 증명 관련 정책 변경
- JSON 정책 문서 문제 해결
  - 정책 검증
  - JSON 편집기에서 정책 검증에 대한 권한 없음
  - JSON 정책 객체가 둘 이상인 경우
  - JSON Statement 요소가 둘 이상인 경우
  - JSON Statement 요소의 Effect, Action 또는 Resource 요소가 둘 이상인 경우

- JSON 버전 요소 누락

시각적 편집기를 사용하여 문제 해결

고객 관리형 정책을 생성 또는 편집할 때 시각적 편집기의 정보를 사용하여 정책의 오류를 해결할 수 있습니다. 시각적 편집기를 사용하여 정책을 생성하는 예시를 보려면 the section called “자격 증명에 대한 액세스 제어” 섹션을 참조하세요.

정책 재구성

정책을 생성할 때 AWS는 정책을 검증, 처리 및 변환한 후 저장합니다. 정책이 검색되면 AWS는 권한을 변경하지 않고 정책을 사람이 읽을 수 있는 형식으로 변환합니다. 이로 인해 정책 시각적 편집기 또는 JSON 탭에 표시되는 내용이 달라질 수 있습니다.

- 시각적 편집기 권한 블록이 추가, 제거 또는 재정렬될 수 있고, 블록 내의 내용이 최적화될 수 있습니다.
- JSON 탭에서 사소한 공백은 제거되며, JSON 맵 내의 요소는 재정렬될 수 있습니다. 또한 보안 주체 요소 내의 AWS 계정 ID는 AWS 계정 루트 사용자의 Amazon 리소스 이름(ARN)으로 교체할 수 있습니다.

이러한 변경 가능성 때문에 JSON 정책 문서를 문자열로 비교하면 안 됩니다.

AWS Management Console에서 고객 관리형 정책을 생성할 때 JSON 편집기에서 완전히 작업하도록 선택할 수 있습니다. 시각적 편집기에서 정책을 변경하지 않고 JSON 편집기에서 다음을 선택하면 정책을 재구성할 가능성이 적습니다. 시각적 편집기를 사용하면 IAM에서 정책을 재구성하여 그 표시를 최적화할 수 있습니다. 이러한 재구성은 편집 세션에만 존재하며 자동으로 저장되지 않습니다.

편집 세션에서 정책이 재구성되면 IAM이 다음 상황에 따라 재구성을 저장할지 여부를 결정합니다.

이 편집기 옵션 사용	정책을 편집한 경우	그런 다음 이 탭에서 다음을 선택합니다	변경 사항 저장을 선택한 경우
시각적	편집됨	시각적	정책이 재구성됨
시각적	편집됨	JSON	정책이 재구성됨
시각적	편집되지 않음	시각적	정책이 재구성됨

이 편집기 옵션 사용	정책을 편집한 경우	그런 다음 이 탭에서 다음을 선택합니다	변경 사항 저장을 선택한 경우
JSON	편집됨	시각적	정책이 재구성됨
JSON	편집됨	JSON	정책 구성이 변경되지 않음
JSON	편집되지 않음	JSON	정책 구성이 변경되지 않음

IAM은 여러 서비스, 리소스 유형 또는 조건 키를 허용하는 문이나 권한 블록이 있는 정책 또는 복잡한 정책을 재구성할 수 있습니다.

시각적 편집기에서 리소스 ARN 선택

시각적 편집기를 사용하여 정책을 생성하거나 편집할 때 먼저 서비스를 선택한 다음 해당 서비스에서 작업을 선택해야 합니다. 선택한 서비스 및 작업이 특정 리소스 선택을 지원하는 경우에는 시각적 편집기에 지원되는 리소스 유형이 나열됩니다. 그런 다음 Add ARN(ARN 추가)를 선택하여 리소스에 대한 세부 정보를 제공합니다. 리소스 유형에 대한 ARN을 추가하기 위해 다음 옵션에서 선택할 수 있습니다.

- ARN 빌더 사용 - 리소스 유형에 따라 ARN을 빌드하는 여러 필드가 표시될 수 있습니다. 모두 선택을 선택하여 지정된 설정의 값에 대한 권한을 제공할 수도 있습니다. 예를 들어, Amazon EC2 읽기 액세스 레벨 그룹을 선택하면 정책의 작업이 `instance` 리소스 유형을 지원합니다. 리소스에 대해 Region, Account 및 InstanceId 값을 입력합니다. 계정 ID를 입력하지만 리전 및 인스턴스 ID에 대해 모두 선택을 선택한 경우 정책은 계정의 모든 인스턴스에 대해 권한을 부여합니다.
- ARN 입력 또는 붙여넣기 - Amazon 리소스 이름(ARN)별로 리소스를 지정할 수 있습니다. ARN의 필드(각 콜론 쌍 사이)에 와일드카드 문자(*)를 포함할 수 있습니다. 자세한 내용은 IAM JSON 정책 요소: Resource 단원을 참조하십시오.

시각적 편집기에서 권한 거부

기본적으로 시각적 편집기를 사용하여 생성하는 정책은 사용자가 선택하는 작업을 허용합니다. 대신 선택한 작업을 거부하려면 Switch to deny permissions(권한 거부로 전환)을 선택합니다. 요청은 기본적으로 거부되므로 사용자에게 필요한 작업과 리소스에만 권한을 허용하는 것이 좋습니다. 다른 문이나 정책에서 허용되는 권한을 별도로 재정의하려는 경우에만 거부 문을 생성해야 합니다. 권한 거부의

수가 늘어나면 권한 문제를 해결하기가 더 어려워질 수 있기 때문에 그 수를 최소한으로 제한하는 것이 좋습니다. IAM이 정책 로직을 평가하는 방법에 대한 자세한 정보는 정책 평가 로직 섹션을 참조하세요.

> **Note**
>
> 기본적으로 AWS 계정 루트 사용자만 해당 계정의 모든 리소스에 액세스할 수 있습니다. 따라서 루트 사용자로 로그인하지 않은 경우 정책이 부여한 권한이 있어야 합니다.

시각적 편집기에서 여러 서비스 지정

시각적 편집기를 사용하여 정책을 생성할 때 한 번에 서비스 하나만 선택할 수 있습니다. 시각적 편집기는 해당 서비스 하나에 대한 작업에서 선택할 수 있도록 허용하므로 이렇게 하는 것이 모범 사례입니다. 그런 다음 해당 서비스 및 선택한 작업에서 지원되는 리소스 중에서 선택합니다. 이렇게 하면 정책을 쉽게 생성하고 문제를 해결할 수 있습니다.

와일드카드 문자(*)를 사용하여 여러 서비스를 수동으로 지정할 수도 있습니다. 예를 들어, **`Code*`**를 입력하여 `CodeBuild` 및 `CodeCommit`과 같이 `Code`로 시작하는 모든 서비스에 대한 권한을 제공합니다. 그러나 정책을 완료하려면 작업 및 리소스 ARN을 입력해야 합니다. 또한 정책을 저장하면 각 서비스를 별도의 권한 블록에 포함하도록 정책이 재구성될 수 있습니다.

또는 서비스에 대해 JSON 구문(예: 와일드카드)을 사용하기 위해 JSON 편집기 옵션을 통해 정책을 생성, 편집 및 저장합니다.

시각적 편집기에서 정책의 크기 줄이기

시각적 편집기를 사용하여 정책을 생성할 때 IAM은 정책을 저장하기 위해 JSON 문서를 생성합니다. JSON 편집기 옵션으로 전환하여 이 문서를 볼 수 있습니다. 이 JSON 문서가 정책의 크기 제한을 초과할 경우, 시각적 편집기에 오류 메시지가 표시됩니다. 정책을 검토하고 저장할 수 없습니다. 관리형 정책의 크기에 대한 IAM 제한을 보려면 IAM 및 STS 문자 제한 섹션을 참조하세요.

시각적 편집기에서 정책의 크기를 줄이려면 정책을 편집하거나 권한 블록을 다른 정책으로 옮깁니다. 오류 메시지에는 정책 문서에 포함된 문자 수가 제공됩니다. 이 정보를 사용하여 정책의 크기를 줄일 수 있습니다.

시각적 편집기에서 인식할 수 없는 서비스, 작업 또는 리소스 유형 수정

시각적 편집기에서 정책에 인식할 수 없는 서비스, 작업 또는 리소스 유형이 포함되어 있다는 경고가 표시될 수 있습니다.

> **Note**
>
> IAM은 정책 요약을 지원하는 서비스의 이름, 작업 및 리소스 유형을 검토합니다. 그러나 존재하지 않는 리소스 값이나 조건이 정책 요약에 포함될 수 있습니다. 항상 정책 시뮬레이터로 정책을 테스트합니다.

정책에 인식할 수 없는 서비스, 작업 또는 리소스 유형이 포함되는 경우 다음 오류 중 하나가 발생한 것입니다.

- 미리 보기 서비스 - 미리 보기에 있는 서비스는 시각적 편집기를 지원하지 않습니다. 미리 보기에 참여하고 있는 경우 정책을 완료하려면 작업 및 리소스 ARN을 수동으로 입력해야 합니다. 모든 경고를 무시하고 계속할 수는 있습니다. 또는 JSON 편집기 옵션을 선택하여 JSON 정책 문서를 입력하거나 붙여 넣을 수 있습니다.
- 사용자 지정 서비스 - 사용자 지정 서비스는 시각적 편집기를 지원하지 않습니다. 사용자 지정 서비스를 사용하고 있는 경우 정책을 완료하려면 작업 및 리소스 ARN을 수동으로 입력해야 합니다. 모든 경고를 무시하고 계속할 수는 있습니다. 또는 JSON 편집기 옵션을 선택하여 JSON 정책 문서를 입력하거나 붙여 넣을 수 있습니다.
- 시각적 편집기를 지원하지 않는 서비스 - 정책에 시각적 편집기를 지원하지 않는 정식 버전(GA) 서비스가 포함되어 있는 경우 정책을 완료하려면 작업 및 리소스 ARN을 수동으로 입력해야 합니다. 모든 경고를 무시하고 계속할 수는 있습니다. 또는 JSON 편집기 옵션을 선택하여 JSON 정책 문서를 입력하거나 붙여 넣을 수 있습니다.

  일반적으로 사용할 수 있는 서비스는 공개적으로 출시된 서비스이며 프리뷰 또는 사용자 지정 서비스가 아닙니다. 인식할 수 없는 서비스를 일반적으로 사용할 수 있고 이름을 올바르게 입력한 경우 서비스는 시각적 편집기를 지원하지 않습니다. GA 서비스에 대한 시각적 편집기 또는 정책 요약 지원을 요청하는 방법을 알아보려면 서비스가 IAM 정책 요약을 지원하지 않음 단원을 참조하십시오.
- 시각적 편집기를 지원하지 않는 작업 - 지원되지 않는 작업과 함께 지원되는 서비스가 정책에 포함되어 있는 경우 정책을 완료하려면 리소스 ARN을 수동으로 입력해야 합니다. 모든 경고를 무시하고 계속할 수는 있습니다. 또는 JSON 편집기 옵션을 선택하여 JSON 정책 문서를 입력하거나 붙여 넣을 수 있습니다.

  지원되지 않는 작업과 함께 지원되는 서비스가 정책에 포함되어 있으면 서비스가 시각적 편집기를 완전히 지원하지 않습니다. GA 서비스에 대한 시각적 편집기 또는 정책 요약 지원을 요청하는 방법을 알아보려면 서비스가 IAM 정책 요약을 지원하지 않음 단원을 참조하십시오.

- 시각적 편집기를 지원하지 않는 리소스 유형 - 지원되지 않는 리소스 유형과 함께 지원되는 작업이 정책에 포함되어 있는 경우 경고를 무시하고 계속 진행할 수 있습니다. 그러나 IAM은 선택한 모든 작업에 대한 리소스를 포함했는지를 확인할 수 없으며, 추가 경고가 표시될 수 있습니다.
- 오타 - 시각적 편집기에 서비스, 작업 또는 리소스를 수동으로 입력할 경우 오타가 포함된 정책이 생성될 수 있습니다. 서비스 및 작업 목록에서 선택하여 시각적 편집기를 사용하는 것이 좋습니다. 이후 프롬프트에 따라 리소스 섹션을 작성합니다. 서비스가 시각적 편집기를 완전히 지원하지 않는 경우 정책의 부분을 수동으로 입력해야 할 수 있습니다.

  정책에 위와 같은 오류가 없다는 것을 확신한다면 오타가 포함된 것일 수 있습니다. 다음과 같은 문제가 있는지 확인합니다.
  - 철자가 틀린 서비스, 작업 및 리소스 유형 이름(예: `s3` 대신 `s2` 또는 `ListAllMyBuckets` 대신 `ListMyBuckets`)
  - ARN의 불필요한 텍스트(예: `arn:aws:s3: : :*`)
  - 작업의 콜론 누락(예: `iam.CreateUser`)

  다음을 선택하여 정책 요약을 검토하고 정책에 오타가 있는지 평가할 수 있습니다. 이후 정책에서 원하는 권한을 제공하는지 여부를 확인합니다.

정책 요약을 사용하여 문제 해결

정책 요약과 관련된 문제를 진단하고 해결할 수 있습니다.

정책 요약 누락

IAM 콘솔에는 정책에서 각 서비스에 대해 허용되거나 거부되는 액세스 레벨, 리소스, 조건을 설명하는 정책 요약 테이블이 포함되어 있습니다. 정책은 3가지 테이블, 즉 정책 요약, 서비스 요약, 작업 요약으로 요약됩니다. 정책 요약 테이블에는 서비스 목록과 선택한 정책에 의해 정의된 권한의 요약이 포함되어 있습니다. 해당 정책에 대한 정책 세부 정보 페이지에서 엔터티에 연결된 정책에 대한 정책 요약을 볼 수 있습니다. 정책 페이지에서 관리형 정책에 대한 정책 요약을 볼 수 있습니다. AWS에서 정책 요약을 렌더링할 수 없는 경우 JSON 정책 문서가 제공되며 다음 오류가 표시됩니다.

이 정책에 대한 요약을 생성할 수 없습니다. 하지만 JSON 정책 문서를 보거나 편집할 수 있습니다.(A summary for this policy cannot be generated. You can still view or edit the JSON policy document.)

정책이 요약을 포함하지 않을 경우 다음 오류 중 하나가 발생한 것입니다.

- 지원되지 않는 정책 요소 - IAM은 다음 정책 요소 중 하나를 포함하는 정책에 대해 정책 요약 생성을 지원하지 않습니다.

  - `Principal`
  - `NotPrincipal`
  - `NotResource`
- 정책 권한 없음 - 정책이 유효한 권한을 제공하지 않을 경우 정책 요약을 생성할 수 없습니다. 예를 들어 정책이 요소 `"NotAction": "*"`과 함께 단일 명령문을 포함하는 경우 이 정책은 "모든 작업"(*)을 제외한 모든 작업에 대한 액세스 권한을 부여합니다. 즉 어떤 작업에 대해서도 `Deny` 또는 `Allow` 액세스 권한을 부여하지 않습니다.

> **Note**
>
> `NotPrincipal`, `NotAction`, `NotResource` 등의 이러한 정책 요소를 사용할 때는 주의해야 합니다. 정책 요소 사용에 대한 자세한 정보는 IAM JSON 정책 요소 참조 단원을 참조하십시오.

일치하지 않는 서비스와 리소스를 입력한 경우 유효한 권한을 제공하지 않는 정책을 생성할 수 있습니다. 이는 한 서비스의 작업과 다른 서비스의 리소스를 지정하는 경우에 발생할 수 있습니다. 이 경우에는 정책 요약이 나타납니다. 요약의 리소스 열에 다른 서비스의 리소스를 포함할 수 있는 경우에만 문제가 있다는 표시가 나타납니다. 이 열에 일치하지 않는 리소스가 포함되어 있으면 정책에 오류가 있는지 검토해야 합니다. 정책을 더 잘 이해하려면 항상 정책 시뮬레이터로 테스트합니다.

정책 요약에 인식할 수 없는 서비스, 작업 또는 리소스 유형 포함됨

IAM 콘솔에서 정책 요약에 경고 기호
(⚠)
가 있으면 정책 요약에 인식할 수 없는 서비스, 작업 또는 리소스 유형이 포함되었을 수 있습니다. 정책 요약 내의 경고에 대해 알아보려면 정책 요약(서비스 목록)을 참조하십시오.

> **Note**
>
> IAM은 정책 요약을 지원하는 서비스의 이름, 작업 및 리소스 유형을 검토합니다. 그러나 존재하지 않는 리소스 값이나 조건이 정책 요약에 포함될 수 있습니다. 항상 정책 시뮬레이터로 정책을 테스트합니다.

정책에 인식할 수 없는 서비스, 작업 또는 리소스 유형이 포함되는 경우 다음 오류 중 하나가 발생한 것입니다.

- 미리 보기 서비스 - 미리 보기에 있는 서비스는 정책 요약을 지원하지 않습니다.
- 사용자 지정 서비스 - 사용자 지정 서비스는 정책 요약을 지원하지 않습니다.
- 서비스가 요약을 지원하지 않음 - 정책 요약을 지원하지 않는 정식 버전(GA) 서비스가 정책에 포함되어 있으면 서비스가 정책 요약 테이블의 알 수 없는 서비스(Unrecognized services) 섹션에 포함됩니다. 일반적으로 사용할 수 있는 서비스는 공개적으로 출시된 서비스이며 프리뷰 또는 사용자 지정 서비스가 아닙니다. 인식할 수 없는 서비스가 정식 버전이고 이름을 올바르게 입력한 경우에는 서비스에서 IAM 정책 요약을 지원하지 않습니다. GA 서비스에 대한 정책 요약 지원을 요청하는 방법을 알아보려면 서비스가 IAM 정책 요약을 지원하지 않음을 참조하십시오.
- 작업이 요약을 지원하지 않음 - 지원되지 않는 작업과 함께 지원되는 서비스가 정책에 포함되어 있으면 작업이 서비스 요약 테이블의 알 수 없는 작업(Unrecognized actions) 섹션에 포함됩니다. 서비스 요약 내의 경고에 대해 알아보려면 서비스 요약(작업 목록)을 참조하십시오.
- 리소스 유형이 요약을 지원하지 않음 - 정책에 지원되지 않는 리소스 유형을 가진 지원되는 작업이 포함된 경우, 서비스 요약 테이블의 알 수 없는 리소스 유형(Unrecognized resource types) 섹션에 리소스가 포함됩니다. 서비스 요약 내의 경고에 대해 알아보려면 서비스 요약(작업 목록)을 참조하십시오.
- 오타 – AWS는 정책 검증의 일환으로 JSON이 구문적으로 올바른지, 정책에 오타 또는 기타 오류가 있는지 확인합니다.

> **Note**
>
> 가장 좋은 방법은 IAM Access Analyzer를 사용하여 IAM 정책을 검증하여 안전하고 기능적인 권한을 보장하는 것입니다. 기존 정책을 열고 정책 검증 권장 사항을 검토하고 해결하는 것이 좋습니다.

서비스가 IAM 정책 요약을 지원하지 않음

IAM 정책 요약 또는 시각적 편집기가 일반 공개(GA) 서비스 또는 작업을 지원하지 않을 수 있습니다. 일반적으로 사용할 수 있는 서비스는 공개적으로 출시된 서비스이며 프리뷰 또는 사용자 지정 서비스가 아닙니다. 인식할 수 없는 서비스를 일반적으로 사용할 수 있고 이름을 올바르게 입력한 경우 서비스는 이러한 기능을 지원하지 않습니다. 지원되지 않는 작업과 함께 지원되는 서비스가 정책에 포함되어 있으면 서비스가 IAM 정책 요약을 완전히 지원하지 않습니다.

서비스에서 IAM 정책 요약 또는 시각적 편집기 지원을 추가하도록 요청하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 지원되지 않는 서비스가 포함된 정책을 찾습니다.
   - 그 정책이 관리형 정책인 경우, 탐색 창에서 정책을 선택합니다. 정책 목록에서 보려는 정책의 이름을 선택합니다.
   - 사용자에게 연결된 인라인 정책인 경우, 탐색 창에서 사용자를 선택합니다. 사용자 목록에서 정책을 보려는 사용자의 이름을 선택합니다. 사용자에 대한 정책 테이블에서 보려는 정책 요약의 헤더를 확장합니다.
3. 왼쪽의 AWS Management Console 바닥글에서 의견을 선택합니다. IAM에 대한 피드백 상자에 **I request that the <ServiceName> service add support for IAM policy summaries and the visual editor**를 입력합니다. 요약 지원을 바라는 서비스가 두 개 이상인 경우 **I request that the <ServiceName1>, <ServiceName2>, and <ServiceName3> services add support for IAM policy summaries and the visual editor**라고 입력합니다

서비스에서 누락된 작업에 대한 IAM 정책 요약 지원을 추가하도록 요청하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 지원되지 않는 서비스가 포함된 정책을 찾습니다.
   - 그 정책이 관리형 정책인 경우, 탐색 창에서 정책을 선택합니다. 정책 목록에서 보려는 정책의 이름을 선택합니다.
   - 사용자에게 연결된 인라인 정책인 경우, 탐색 창에서 사용자를 선택합니다. 사용자 목록에서 정책을 보려는 사용자의 이름을 선택합니다. 사용자에 대한 정책 표에서 보려는 정책의 이름을 선택하여 정책 요약을 펼칩니다.
3. 정책 요약에서 지원되지 않는 작업을 포함하는 서비스의 이름을 선택합니다.
4. 왼쪽의 AWS Management Console 바닥글에서 의견을 선택합니다. IAM에 대한 피드백 상자에 **I request that the <ServiceName> service add IAM policy summary and the visual editor support for the <ActionName> action**를 입력합니다. 지원되지 않는 작업을 두 개 이상 보고하는 경우 **I request that the <ServiceName> service add IAM policy summary and the visual editor support for the <ActionName1>, <ActionName2>, and <ActionName3> actions**라고 입력합니다

다른 서비스에 누락된 작업을 포함하도록 요청하려면 마지막 세 단계를 반복합니다.

정책이 필요한 권한을 부여하지 않음

사용자, 그룹, 역할 또는 리소스에 권한을 할당하려면 권한을 정의하는 문서인 정책을 생성해야 합니다. 정책 문서에는 다음 요소가 포함됩니다.

- 효과 - 정책에서 액세스를 허용하는지 또는 거부하는지 여부
- 작업 - 정책에서 허용하거나 거부하는 작업 목록
- 리소스 - 작업이 발생할 수 있는 리소스 목록
- 조건(선택 사항) - 정책에서 권한을 부여하는 상황

이러한 요소와 기타 정책 요소에 대한 자세한 정보는 IAM JSON 정책 요소 참조 단원을 참조하십시오.

액세스 권한을 부여하려면 정책이 지원되는 리소스를 가진 작업을 정의해야 합니다. 정책에 조건도 있는 경우 이 조건은 전역 조건 키를 포함해야 하거나, 작업에 적용해야 합니다. 작업에서 어떤 리소스를 지원하는지 확인하려면 해당 서비스의 AWS 설명서를 참조하십시오. 작업에서 지원되는 조건을 알아보려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

정책에서 권한을 부여하지 않는 작업, 리소스 또는 조건을 정의하는지 확인하세요. https://console.aws.amazon.com/iam/의 IAM 콘솔을 사용하여 정책에 대한 정책 요약 확인합니다. 정책 요약을 사용하여 정책의 문제를 식별하고 수정할 수 있습니다.

IAM 정책에 정의되었는데도 요소가 권한을 부여하지 않는 몇 가지 이유는 다음과 같습니다.

- 적용 가능한 리소스 없이 작업이 정의된 경우
- 적용 가능한 작업 없이 리소스가 정의된 경우
- 적용 가능한 작업 없이 조건이 정의된 경우

경고를 포함하는 정책 요약의 예를 보려면 the section called “정책 요약(서비스 목록)” 단원을 참조하십시오.

적용 가능한 리소스 없이 작업이 정의된 경우

아래 정책은 모든 `ec2:Describe*` 작업과 특정 리소스를 정의합니다. 이러한 작업 중에서 리소스 수준 권한을 지원하는 작업이 없기 때문에 어떤 `ec2:Describe` 작업도 부여되지 않습니다. 리소스 수준 권한이란 작업이 정책의 `Resource` 요소에 있는 ARN을 사용하여 리소스를 지원함을 의미합니다. 작업이 리소스 수준 권한을 지원하지 않는 경우에는 정책의 이 명령문에서 * 요소에 와일드카드

(Resource)를 사용해야 합니다. 리소스 수준 권한을 서비스에 대해 알아보려면 AWS IAM으로 작업하는 서비스 단원을 참조하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "ec2:Describe*",
        "Resource": "arn:aws:ec2:us-east-2:ACCOUNT-ID:instance/*"
    }]
}
```

이 정책은 어떤 권한도 제공하지 않으며, 정책 요약에는 다음 오류가 포함됩니다.

```
This policy does not grant any permissions. To grant access, policies must
have an action that has an applicable resource or condition.
```

정책을 수정하려면 * 요소에 Resource를 사용해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "ec2:Describe*",
        "Resource": "*"
    }]
}
```

적용 가능한 작업 없이 리소스가 정의된 경우

아래 정책은 Amazon S3 버킷 리소스를 정의하지만, 해당 리소스에서 수행할 수 있는 S3 작업을 포함하지 않습니다. 이 정책은 또한 모든 Amazon CloudFront 작업에 대한 전체 액세스 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "cloudfront:*",
        "Resource": [
            "arn:aws:cloudfront:*",
            "arn:aws:s3:::examplebucket"
```

```
        ]
    }]
}
```

이 정책은 모든 CloudFront 작업에 대한 권한을 제공합니다. 하지만 정책에서 S3 작업을 정의하지 않고 S3 examplebucket 리소스를 정의하기 때문에 정책 요약에 다음 경고가 표시됩니다.

```
This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition.
```

이 정책을 수정하여 S3 버킷 권한을 제공하려면 버킷 리소스에서 수행할 수 있는 S3 작업을 정의해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "cloudfront:*",
            "s3:CreateBucket",
            "s3:ListBucket*",
            "s3:PutBucket*",
            "s3:GetBucket*"
        ],
        "Resource": [
            "arn:aws:cloudfront:*",
            "arn:aws:s3:::examplebucket"
        ]
    }]
}
```

또는 이 정책을 수정하여 CloudFront 권한만 제공하려면 S3 리소스를 제거합니다.

적용 가능한 작업 없이 조건이 정의된 경우

아래 정책은 S3 접두사가 custom이고 버전 ID가 1234일 경우 모든 S3 리소스에 대해 2개의 Amazon S3 작업을 정의합니다. 하지만 s3:VersionId 조건 키가 객체 버전 태그 지정에 사용되었으며, 정의된 버킷 작업이 이 조건 키를 지원하지 않습니다. 작업에서 지원되는 조건을 알아보려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하고 서비스를 선택하여 조건 키에 대한 서비스 설명서를 확인하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucketVersions",
                "s3:ListBucket"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:prefix": [
                        "custom"
                    ],
                    "s3:VersionId": [
                        "1234"
                    ]
                }
            }
        }
    ]
}
```

이 정책은 버킷 이름에 `s3:ListBucketVersions` 접두사가 있는 경우 `s3:ListBucket` 작업 및 `custom` 작업에 대한 권한을 제공합니다. 하지만 정의된 작업 중 `s3:VersionId` 조건을 지원하는 작업이 없기 때문에 정책 요약에 다음 오류가 표시됩니다.

```
This policy does not grant any permissions. To grant access, policies must have an action that has an applicable resource or condition.
```

이 정책을 수정하여 S3 객체 버전 태그 지정을 사용하려면, `s3:VersionId` 조건 키를 지원하는 S3 작업을 정의해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucketVersions",
                "s3:ListBucket",
```

```
                "s3:GetObjectVersion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:prefix": [
                        "custom"
                    ],
                    "s3:VersionId": [
                        "1234"
                    ]
                }
            }
        }
    ]
}
```

이 정책은 정책의 모든 작업과 조건에 대한 권한을 제공합니다. 하지만 하나의 작업이 모든 조건을 충족하는 경우가 없기 때문에 어떤 권한도 제공하지 않습니다. 이렇게 하는 대신, 적용할 조건을 갖는 작업만 각각 포함하도록 두 개의 구문을 별도로 작성해야 합니다.

이 정책을 수정하려면 두 개의 구문을 작성합니다. 첫째 구문에는 `s3:prefix` 조건을 지원하는 작업이 포함되고, 둘째 구문에는 `s3:VersionId` 조건을 지원하는 작업이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucketVersions",
                "s3:ListBucket"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:prefix": "custom"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "s3:GetObjectVersion",
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:VersionId": "1234"
                }
            }
        }
    ]
}
```

정책 관리 문제 해결

정책 관리와 관련된 문제를 진단하고 해결할 수 있습니다.

IAM 계정에서 정책 연결 또는 분리

일부 AWS 관리형 정책은 서비스에 연결되어 있습니다. 이러한 정책은 해당 서비스에 대한 서비스 연결 역할에서만 사용됩니다. IAM 콘솔에서 정책 세부 정보 페이지를 보면 페이지에 정책이 서비스에 연결되어 있음을 나타내는 배너가 포함되어 있습니다. 이 정책을 IAM 내의 사용자, 그룹 또는 역할에 연결할 수 없습니다. 서비스에 대한 서비스 연결 역할을 생성하면 이 정책이 새 역할에 자동으로 연결됩니다. 정책이 필요하므로 서비스 연결 역할에서 정책을 분리할 수 없습니다.

작업 기반 IAM 자격 증명 관련 정책 변경

작업에 따라 IAM 자격 증명(사용자, 그룹 및 역할)에 대한 정책을 업데이트할 수 있습니다. 이렇게 하려면 CloudTrail 이벤트 기록(Event history)에서 계정 이벤트를 봅니다. CloudTrail 이벤트 로그에는 정책의 권한을 변경하는 데 사용할 수 있는 자세한 이벤트 정보가 포함되어 있습니다.

사용자 또는 역할이 AWS에서 작업을 수행하려 하고 요청이 거부되었습니다.

> 사용자 또는 역할에 작업을 수행할 권한이 있어야 하는지 여부를 고려합니다. 권한을 부여해야 하는 경우 해당 작업과 이들이 액세스하려고 했던 리소스의 ARN도 정책에 추가할 수 있습니다.

사용자나 역할에 사용하지 않는 권한이 있습니다.

> 정책에서 해당 권한을 제거하는 것을 고려해 보세요. 정책은 필요한 작업을 수행하는 데 필요한 최소 권한만 부여해야 합니다.

CloudTrail 사용에 관한 자세한 내용은 AWS CloudTrail사용 설명서의 CloudTrail 콘솔에서 CloudTrail 이벤트 보기를 참조하세요.

JSON 정책 문서 문제 해결

JSON 정책 문서와 관련된 문제를 진단하고 해결할 수 있습니다.

정책 검증

JSON 정책을 생성 또는 편집할 때 IAM은 효과적인 정책을 생성하는 데 도움이 되는 정책 유효성 검사를 수행할 수 있습니다. IAM은 JSON 구문 오류를 식별하는 반면, IAM Access Analyzer는 정책을 더욱 구체화하는 데 도움이 되는 권장 사항과 함께 추가 정책 검사를 제공합니다. 정책 검증에 대한 자세한 내용은 IAM 정책 검증 섹션을 참조하세요. IAM Access Analyzer 정책 검사기 및 실행 가능한 권장 사항에 대한 자세한 내용은 IAM Access Analyzer 정책 검증을 참조하세요.

JSON 편집기에서 정책 검증에 대한 권한 없음

AWS Management Console에서 IAM Access Analyzer 정책 검증 결과를 볼 수 있는 권한이 없는 경우 다음 오류가 나타날 수 있습니다.

```
You need permissions. You do not have the permissions required to perform this operation. Ask your administrator to add permissions.
```

이 오류를 해결하려면 관리자에게 `access-analyzer:ValidatePolicy` 권한을 추가해 달라고 요청합니다.

JSON 정책 객체가 둘 이상인 경우

IAM 정책은 단 하나의 JSON 객체로 구성되어야 합니다. 객체는 중괄호 { }로 묶어 표시합니다. 대괄호([]) 안에 중괄호({})를 추가로 삽입하여 JSON 객체 내에 다른 객체를 중첩시킬 수 있습니다. 정책에서 중괄호({})를 묶는 대괄호([])는 단 하나만 있어야 합니다. 다음 예시는 최상위 레벨에 객체 2개가 추가되었기 때문에 올바르지 않습니다(###으로 표시).

```
{
      "Version": "2012-10-17",
      "Statement":
      {
         "Effect":"Allow",
         "Action":"ec2:Describe*",
         "Resource":"*"
      }
   }
   {
     "Statement": {
        "Effect": "Allow",
```

```
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::my-bucket/*"
      }
    }
```

하지만 올바른 정책 문법을 사용하여 위 예시의 의도를 만족하는 방법도 있습니다. 2개의 정책 객체에 Statement 요소를 각각 추가하지 않고 두 블록을 단일 Statement 요소로 결합하면 됩니다. 그러면 다음 예시와 같이 Statement 요소가 두 객체의 배열을 값으로 인식합니다(굵은체로 표시).

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": "ec2:Describe*",
          "Resource":" *"
        },
        {
          "Effect": "Allow",
          "Action": "s3:*",
          "Resource": "arn:aws:s3:::my-bucket/*"
        }
      ]
    }
```

JSON Statement 요소가 둘 이상인 경우

이 오류는 처음에는 위 오류의 변형으로 보일 수도 있습니다. 하지만 구문으로 보면 다른 유형의 오류입니다. 다음 예시에는 중괄호 { } 한 쌍이 최상위 레벨로 정책 객체 하나만 표시하고 있습니다. 하지만 객체에 포함된 Statement 요소는 2개입니다.

IAM 정책에서는 콜론 왼쪽의 이름(Statement)과 오른쪽의 값으로 구성된 Statement 요소 1개만 추가할 수 있습니다. 그리고, Statement 요소의 값은 Effect 요소 1개와 Action 요소 1개, 그리고 Resource 요소 1개가 중괄호 { }로 묶여 구성된 객체가 되어야 합니다. 다음 예시는 정책 객체에 Statement 요소가 2개 포함되었기 때문에 올바르지 않습니다(###으로 표시).

```
{
      "Version": "2012-10-17",
      "Statement": {
        "Effect": "Allow",
        "Action": "ec2:Describe*",
        "Resource": "*"
```

```
      },
      "Statement": {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::my-bucket/*"
      }
    }
```

값 객체는 여러 값 객체의 배열일 수 있습니다. 이 문제를 해결하려면, 다음 예시와 같이 객체 배열을 사용하여 2개의 Statement 요소를 하나로 결합합니다(굵은체로 표시).

```
{
      "Version": "2012-10-17",
      "Statement": [    {
          "Effect": "Allow",
          "Action": "ec2:Describe*",
          "Resource":"*"
        },
        {
          "Effect": "Allow",
          "Action": "s3:*",
          "Resource": "arn:aws:s3:::my-bucket/*"
        }
     ]
    }
```

Statement 요소 값이 객체 배열이 되었습니다. 이제 위 예시의 배열은 두 객체로 구성되며, 각 객체 자체가 정확한 Statement 요소 값으로 인식됩니다. 배열을 구성하는 각 객체는 쉼표로 구분합니다.

JSON Statement 요소의 Effect, Action 또는 Resource 요소가 둘 이상인 경우

Statement 이름/값 쌍에서 값 부분을 보면 객체가 Effect 요소 1개, Action 요소 1개, 그리고 Resource 요소 1개로 구성되어야 합니다. 다음 정책은 Statement에 2개의 Effect 요소가 있기 때문에 잘못된 정책입니다.

```
{
      "Version": "2012-10-17",
      "Statement": {
        "Effect": "Deny",
        "Effect": "Allow",
        "Action": "ec2:* ",
        "Resource": "*"
```

```
    }
  }
```

> **Note**
>
> 정책 엔진은 새로운 정책이나 편집된 정책에서 이러한 오류를 허용하지 않습니다. 하지만 정책 엔진은 엔진 업데이트 이전에 저장된 정책은 계속 허용합니다. 오류와 관련한 기존 정책 특성은 아래와 같습니다.
>
> - `Effect` 요소가 다수일 때: 마지막 `Effect` 요소만 따릅니다. 나머지 요소는 모두 무시됩니다.
> - `Action` 요소가 여러 개일 때: `Action` 요소가 모두 내부적으로 결합되어 마치 단일 목록인 것처럼 처리됩니다.
> - `Resource` 요소가 여러 개일 때: `Resource` 요소가 모두 내부적으로 결합되어 마치 단일 목록인 것처럼 처리됩니다.
>
> 정책 엔진은 구문 오류 정책을 저장하도록 허용하지 않습니다. 저장하기 전에 정책 오류를 수정하세요. 정책에 대한 모든 정책 검증 권장 사항을 검토하고 수정하세요.

모든 경우 해결책은 잘못 추가된 요소를 삭제하는 것입니다. `Effect` 요소일 때는 삭제 방법이 간단합니다. 앞의 예시에서 Amazon EC2 인스턴스에 대한 권한을 거부하고 싶다면 다음과 같이 정책에서 `"Effect": "Allow",` 라인을 삭제하면 됩니다.

```
{
      "Version": "2012-10-17",
      "Statement": {
        "Effect": "Deny",
        "Action": "ec2:* ",
        "Resource": "*"
      }
    }
```

하지만 중복 요소가 `Action` 또는 `Resource`인 경우에는 해결 방법이 더욱 복잡합니다. 권한을 허용(또는 거부)하려는 작업이 다수일 수도 있고, 여러 리소스에 대한 액세스를 제어할 수도 있기 때문입니다. 예를 들어 다음 예시는 `Resource` 요소가 여러 개이기 때문에 올바르지 않습니다(###으로 표시).

```
{
```

```
      "Version": "2012-10-17",
      "Statement": {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::my-bucket",
        "Resource": "arn:aws:s3:::my-bucket/*"
      }
    }
```

Statement 요소의 값 객체에서 필요한 요소는 각각 한 번만 표시할 수 있습니다. 해결책은 객체 배열에 값을 하나씩만 지정하는 것입니다. 다음 예시는 배열을 값 객체로 사용하여 2개의 리소스 요소를 1개의 Resource 요소로 결합함으로써 이를 설명합니다(굵은체로 표시).

```
{
      "Version": "2012-10-17",
      "Statement": {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": [
          "arn:aws:s3:::my-bucket",
          "arn:aws:s3:::my-bucket/*"
        ]
      }
    }
```

JSON 버전 요소 누락

Version 정책 요소는 정책 버전과 다릅니다. Version 정책 요소는 정책 내에서 사용되며 정책 언어의 버전을 정의합니다. 비교를 위해 정책 버전이 IAM에서 고객 관리형 정책을 변경할 때 생성됩니다. 변경된 정책은 기존 정책을 덮어쓰지 않습니다. 대신 IAM에서 관리형 정책의 새 버전을 생성합니다. Version 정책 요소에 대한 자세한 정보는 IAM JSON 정책 요소: Version을 참조하세요. 정책 버전에 대한 자세한 정보는 the section called “IAM 정책 버전 관리” 섹션을 참조하세요.

AWS 기능이 점차 진화하면서 IAM 정책에도 이를 지원할 수 있도록 새로운 기능이 추가되었습니다. 간혹 정책 구문이 업데이트될 때마다 새로운 버전 번호가 추가됩니다. 정책 문법에서 최신 기능을 사용하는 경우에는 정책 구문 분석 엔진에게 사용 버전을 알려주어야 합니다. 기본 정책 버전은 "2008-10-17"입니다. 이때 이후 추가된 정책 기능을 사용하려면 원하는 기능을 지원하는 버전 번호를 지정해야 합니다. 따라서 항상 최신 정책 구문 버전 번호("Version": "2012-10-17")를 추가할 것을 권장합니다. 예를 들어 다음 정책은 리소스에 대해 ARN의 ${...} 변수를 사용하기 때문에 올바르지 않습니다. 정책 변수를 지원하는 정책 구문 버전을 지정하는 데 실패합니다(*red*에서 호출).

```
{
  "Statement":
  {
    "Action": "iam:*AccessKey*",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
  }
}
```

다음과 같이 정책 상단에 정책 변수를 지원하는 첫 번째 IAM API 버전인 2012-10-17 값과 함께 Version 요소를 추가하면 이 문제가 해결됩니다(굵은 글씨체로 표시).

```
{
  "Version": "2012-10-17",
  "Statement":
  {
    "Action": "iam:*AccessKey*",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
  }
}
```

FIDO 보안 키 문제 해결

여기 정보를 사용하여 FIDO2 보안 키 작업 시 공통적으로 발생할 수 있는 문제를 진단하세요.

주제

- FIDO 보안 키를 활성화할 수 없습니다.
- FIDO 보안 키를 사용해 로그인할 수 없습니다.
- FIDO 보안 키를 분실했거나 고장 났습니다.
- 기타 문제

FIDO 보안 키를 활성화할 수 없습니다.

IAM 사용자인지 시스템 관리자인지 자신의 상태에 따라 다음 해결 방법을 문의하세요.

IAM 사용자

FIDO 보안 키가 활성화되지 않으면 다음 사항을 확인하세요.

- 지원되는 구성을 사용 중입니까?

  WebAuthn 및 AWS와 함께 사용할 수 있는 디바이스와 브라우저에 대한 자세한 내용을 알아보려면 패스키 및 보안 키 사용이 지원되는 구성 섹션을 참조하세요.

- Mozilla Firefox를 사용 중입니까?

  현재 Firefox 버전은 기본적으로 WebAuthn을 지원합니다. Firefox에서 WebAuthn 지원을 활성화하려면 다음을 수행하세요.

  1. Firefox 주소 표시줄에 **`about:config`**를 입력합니다.
  2. 열리는 화면의 검색줄에 기본 **`webauthn`**를 입력합니다.
  3. security.webauth.webauthn을 선택하고 값을 true로 변경합니다.

- 브라우저 플러그인을 사용 중입니까?

  AWS는 플러그인 사용을 통한 WebAuthn 브라우저 지원 추가를 지원하지 않습니다. 대신 WebAuthn 표준을 기본적으로 지원하는 브라우저를 사용하세요.

  지원되는 브라우저를 사용하더라도 WebAuthn과 호환되지 않는 플러그인이 있을 수 있습니다. 호환되지 않는 플러그인이 있으면 FIDO 준수 보안 키를 활성화하고 사용하지 못할 수 있습니다. 호환되지 않는 플러그인을 모두 비활성화하고 브라우저를 새로 시작합니다. 그런 다음 FIDO 보안 키를 다시 활성화합니다.

- 적절한 권한이 있습니까?

  위 호환성 문제가 없는 경우 적절한 권한이 없는 경우일 수 있습니다. 시스템 관리자에게 문의하십시오.

시스템 관리자

IAM 사용자가 지원되는 구성을 사용 중인데도 FIDO 보안 키를 활성화할 수 없다면 권한을 확인하세요. 자세한 예제는 IAM 자습서: 사용자가 자신의 자격 증명 및 MFA 설정을 관리하도록 허용 단원을 참조하십시오.

FIDO 보안 키를 사용해 로그인할 수 없습니다.

FIDO 보안 키를 사용해서 AWS Management Console에 로그인할 수 없는 경우 먼저 패스키 및 보안 키 사용이 지원되는 구성의 내용을 확인하세요. 지원되는 구성을 사용하는데 로그인이 안 되면 시스템 관리자에게 연락해 도움을 받으십시오.

FIDO 보안 키를 분실했거나 고장 났습니다.

현재 지원되는 MFA 유형을 조합하여 최대 8개의 MFA 디바이스를 사용자에게 할당할 수 있습니다. MFA 디바이스가 여러 개인 경우 AWS Management Console에 로그인하는 데 하나의 MFA 디바이스만 있으면 됩니다. FIDO 보안 키의 교체는 하드웨어 TOTP 토큰의 교체와 비슷합니다. 임의 유형의 MFA 디바이스 분실 또는 손상의 경우 IAM에서 MFA로 보호되는 ID 복구의 내용을 참조하세요.

기타 문제

여기에 나오지 않은 FIDO 보안 키의 문제는 다음 중 하나를 수행해 보십시오.

- IAM 사용자: 시스템 관리자에게 문의하세요.
- AWS 계정 루트 사용자: AWS Support에 문의하세요.

IAM 역할 문제 해결

이 정보를 사용하여 IAM 역할 작업 시 공통적으로 발생할 수 있는 문제를 진단 및 수정하세요.

주제

- 역할을 수임할 수 없음
- 내 AWS 계정에 표시되는 새 역할
- AWS 계정에서 역할을 편집하거나 삭제할 수 없음
- iam:PassRole를 수행하도록 인증되지 않음
- 12시간 길이 세션을 선택한 경우 역할을 수임할 수 없는 이유 (AWS CLI, AWS API)
- IAM 콘솔에서 역할을 전환하려고 하는데 오류가 발생합니다.
- 역할에 작업 수행을 허용하는 정책이 있지만 “액세스 거부”가 표시됩니다.
- 서비스에서 역할의 기본 정책 버전을 만들지 않았습니다.
- 콘솔에 서비스 역할에 대한 사용 사례가 없음

역할을 수임할 수 없음

다음을 확인하세요.

- 사용자가 역할 세션 내에서 현재 역할을 다시 수임할 수 있도록 허용하려면 역할 ARN 또는 AWS 계정 ARN을 역할 신뢰 정책의 보안 주체로 지정합니다. Amazon EC2, Amazon ECS, Amazon EKS,

Lambda와 같이 컴퓨팅 리소스를 제공하는 AWS 서비스는 임시 보안 인증을 제공하며 해당 보안 인증을 자동으로 업데이트합니다. 이렇게 하면 항상 유효한 자격 증명 집합을 가질 수 있습니다. 이러한 서비스의 경우 임시 자격 증명을 획득하기 위해 현재 역할을 다시 수임할 필요는 없습니다. 하지만 세션 태그 또는 세션 정책을 전달하려는 경우 현재 역할을 다시 수임해야 합니다. 보안 주체 역할 ARN 또는 AWS 계정 ARN을 추가하기 위해 역할 신뢰 정책을 수정하는 방법을 알아보려면 역할 트러스트 정책 업데이트 을 참조하세요.

- AWS Management Console을 사용해 역할을 수임하는 경우 역할 이름을 정확하게 사용해야 합니다. 역할을 수임할 때 역할 이름은 대소문자를 구분합니다.
- AWS STS API 또는 AWS CLI를 사용해 역할을 수임하는 경우 ARN 내의 역할 이름을 정확하게 사용해야 합니다. 역할을 수임할 때 역할 이름은 대소문자를 구분합니다.
- 해당 IAM 정책에서 사용자가 위임하려는 역할에 대해 `sts:AssumeRole`을 호출할 수 있는 권한을 부여하는지 확인하세요. IAM 정책의 `Action` 요소는 `AssumeRole` 작업을 호출할 수 있어야 합니다. 또한 IAM 정책의 `Resource` 요소는 위임하려는 역할을 지정해야 합니다. 예를 들어 `Resource` 요소는 ARN(Amazon Resource Name) 또는 와일드카드(*)를 통해 역할을 지정할 수 있습니다. 예를 들어 사용자에게 적용되는 하나 이상의 정책은 다음과 유사한 권한을 부여해야 합니다.

```
"Effect": "Allow",
"Action": "sts:AssumeRole",
"Resource": "arn:aws:iam::account_id_number:role/role-name-you-want-to-assume"
```

- IAM 자격 증명에 IAM 정책에 필요한 태그가 있는지 확인하세요. 예를 들어 다음 정책 권한에서 `Condition` 요소는 역할을 위임하도록 요청할 보안 주체에게 특정 태그가 있어야 한다는 것을 요구합니다. `department = HR` 또는 `department = CS` 태그가 지정되어 있어야 합니다. 그렇지 않으면 역할을 위임할 수 없습니다. IAM 사용자 및 역할 태그 지정에 대한 자세한 내용은 the section called “IAM 리소스용 태그” 섹션을 참조하세요.

```
"Effect": "Allow",
"Action": "sts:AssumeRole",
"Resource": "*",
"Condition": {"StringEquals": {"aws:PrincipalTag/department": [
        "HR",
        "CS"
    ]}}
```

- 역할의 신뢰 정책에 지정된 모든 조건을 만족하고 있는지 확인하십시오. `Condition` 요소는 만료 날짜와 외부 ID를 지정하거나, 반드시 특정 IP 주소를 이용해야만 요청이 가능하도록 지정할 수 있습니다. 다음 예시를 고려하십시오. 현재 날짜가 지정된 날짜 이후의 시간이면 이 정책은 일치하지 않으며 해당 역할을 수임할 권한을 사용자에게 부여할 수 없습니다.

```
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::account_id_number:role/role-name-you-want-to-assume"
    "Condition": {
        "DateLessThan" : {
            "aws:CurrentTime" : "2016-05-01T12:00:00Z"
        }
    }
```

- AssumeRole을 호출하는 AWS 계정이 위임하려는 역할에 대해 신뢰할 수 있는 엔터티인지 확인하십시오. 신뢰할 수 있는 대상이라면 역할의 신뢰 정책에 Principal로 정의되어 있습니다. 다음은 수임할 역할에 연결된 신뢰 정책의 예입니다. 이 예에서 IAM 사용자가 로그인한 계정 ID는 123456789012여야 합니다. 계정 번호가 역할의 신뢰 정책의 Principal 요소에 명시되어 있지 않은 경우 해당 역할을 수임할 수 없습니다. 액세스 정책에서 어떤 권한이 부여되었는지는 중요하지 않습니다. 예시 정책은 2017년 7월 1일부터 2017년 12월 31일(UTC)까지 발생한 작업에 대한 권한을 제한합니다. 이 날짜 전이나 후에 로그인한 경우에는 정책이 일치하지 않기 때문에 해당 역할을 수행할 수 없습니다.

```
    "Effect": "Allow",
    "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
    "Action": "sts:AssumeRole",
    "Condition": {
      "DateGreaterThan": {"aws:CurrentTime": "2017-07-01T00:00:00Z"},
      "DateLessThan": {"aws:CurrentTime": "2017-12-31T23:59:59Z"}
    }
```

- 소스 자격 증명 - 관리자는 소스 자격 증명을 호출하는 작업을 AWS에서 수행하는 사용자 또는 애플리케이션을 식별하도록 사용자 지정 문자열을 전달하기 위해 자격 증명을 요구하도록 역할을 구성할 수 있습니다. 위임되는 역할에 소스 자격 증명을 설정할 필요가 있는지 확인합니다. 소스 자격 증명에 대한 자세한 내용은 위임된 역할로 수행한 작업 모니터링 및 제어 섹션을 참조하세요.

내 AWS 계정에 표시되는 새 역할

일부 AWS 서비스에서는 해당 서비스에 직접 연결된 고유한 유형의 서비스 역할을 사용해야 합니다. 이 서비스 연결 역할은 해당 서비스에서 사전 정의하며 해당 서비스에 필요한 모든 권한을 포함합니다. 필요한 권한을 수동으로 추가할 필요가 없으므로 서비스를 더 쉽게 설정할 수 있습니다. 서비스 연결 역할에 대한 일반적인 내용은 서비스 연결 역할 생성 섹션을 참조하세요.

서비스 연결 역할을 지원하려 할 때 이미 서비스를 사용 중일 수 있습니다. 그런 경우 계정의 새 역할에 대해 알리는 이메일을 받을 수 있습니다. 이 역할에는 서비스에서 사용자를 대신하여 작업을 수행하는 데 필요한 모든 권한이 포함되어 있습니다. 따라서 이 역할을 지원하기 위해 별도의 조치를 취할 필요가 없습니다. 그러나 계정에서 역할을 삭제하면 안 됩니다. 그렇게 하면 서비스가 AWS 리소스에 액세스하는 데 필요한 권한을 제거할 수 있습니다. IAM 콘솔의 IAM 역할 페이지에서 계정의 서비스 연결 역할을 볼 수 있습니다. 서비스 연결 역할은 테이블의 Trusted entities(신뢰할 수 있는 개체) 열에 (Service-linked role)((서비스 연결 역할))로 표시됩니다.

서비스 연결 역할을 지원하는 서비스에 대한 자세한 내용을 알아보려면 AWS IAM으로 작업하는 서비스를 참조하고 서비스 연결 역할(Service-Linked Role) 열에 예(Yes)가 있는 서비스를 찾습니다. 서비스의 서비스 연결 역할 사용에 대한 정보를 보려면 [예]링크를 선택합니다.

AWS 계정에서 역할을 편집하거나 삭제할 수 없음

IAM에서 서비스 연결 역할에 대한 권한을 삭제하거나 편집할 수 없습니다. 이러한 역할에는 사용자 대신 작업을 수행하기 위해 서비스에 필요한 신뢰 및 권한이 미리 지정되어 포함됩니다. IAM 콘솔, AWS CLI, API 등을 사용하여 서비스 연결 역할의 설명만 편집할 수 있습니다. 콘솔의 IAM 역할 페이지에서 계정의 서비스 연결 역할을 볼 수 있습니다. 서비스 연결 역할은 테이블의 Trusted entities(신뢰할 수 있는 개체) 열에 (Service-linked role)((서비스 연결 역할))로 표시됩니다. 역할의 요약 페이지 배너에도 해당 역할이 서비스 역할임이 표시됩니다. 이러한 역할은 해당 서비스가 관리 및 삭제 작업을 지원할 경우 연결 서비스를 통해서만 관리하고 삭제할 수 있습니다. 서비스 연결 역할을 수정하거나 삭제하면 서비스에서 AWS 리소스에 액세스하는 데 필요한 권한이 제거될 수 있으므로 주의하십시오.

서비스 연결 역할을 지원하는 서비스에 대한 자세한 내용을 알아보려면 AWS IAM으로 작업하는 서비스를 참조하고 서비스 연결 역할(Service-Linked Role) 열에 예(Yes)가 있는 서비스를 찾습니다.

iam:PassRole를 수행하도록 인증되지 않음

서비스 연결 역할을 생성하는 경우 해당 역할을 서비스에 전달할 권한이 있어야 합니다. 일부 서비스는 서비스에서 작업을 수행할 때 계정에 서비스 연결 역할을 자동으로 생성합니다. 예를 들어 Amazon EC2 Auto Scaling에서는 사용자가 Auto Scaling 그룹을 처음으로 생성할 때 사용자를 대신해 `AWSServiceRoleForAutoScaling` 서비스 연결 역할을 생성합니다. `PassRole` 권한 없이 Auto Scaling 그룹을 생성하려고 하면 다음 오류가 발생합니다.

```
ClientError: An error occurred (AccessDenied) when calling the
PutLifecycleHook operation: User: arn:aws:sts::111122223333:assumed-role/
Testrole/Diego is not authorized to perform: iam:PassRole on resource:
arn:aws:iam::111122223333:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling
```

이 오류를 해결하려면 관리자에게 `iam:PassRole` 권한을 추가해 달라고 요청합니다.

서비스 연결 역할을 지원하는 서비스를 알아보려면 AWS IAM으로 작업하는 서비스 섹션을 참조하세요. 서비스가 자동으로 서비스 연결 역할을 생성하는지 여부를 알아보려면 예 링크를 선택하여 해당 서비스의 서비스 연결 역할 설명서 단원을 참조하십시오.

12시간 길이 세션을 선택한 경우 역할을 수임할 수 없는 이유 (AWS CLI, AWS API)

AWS STS `AssumeRole*` API 또는 `assume-role*` CLI 작업을 사용하여 역할을 위임하는 경우 `DurationSeconds` 파라미터에 대한 값을 지정할 수 있습니다. 값을 900초(15분)에서 해당 역할에 대한 최대 세션 지속 시간 설정까지 지정할 수 있습니다. 이 설정보다 높게 값을 지정하면 작업에 실패합니다. 이 설정의 최댓값은 12시간입니다. 예를 들어 세션 기간으로 12시간을 지정했는데 관리자가 최대 세션 기간으로 6시간을 설정하면 작업에 실패합니다. 역할에 대한 최댓값을 확인하는 방법을 알아보려면 역할의 최대 세션 기간 업데이트 섹션을 참조하세요.

역할 함께 묶기(역할을 사용하여 두 번째 역할 위임)를 사용하는 경우 세션은 최대 1시간으로 제한됩니다. 그런 다음 `DurationSeconds` 파라미터를 사용하여 1시간보다 큰 값을 입력하면 이 작업에 실패합니다.

IAM 콘솔에서 역할을 전환하려고 하는데 오류가 발생합니다.

역할 전환 페이지에 입력하는 정보는 해당 역할에 대한 정보와 일치해야 합니다. 그렇지 않으면 작업이 실패하고 다음과 같은 오류가 나타납니다.

```
Invalid information in one or more fields. Check your information or
contact your administrator.
```

이 오류가 나타나면 다음 정보가 올바른지 확인하십시오.

- 계정 ID 또는 별칭 - AWS 계정 ID는 12자리 숫자입니다. 계정에는 AWS 계정 ID 대신 사용할 수 있는 회사 이름과 같이 친숙한 식별자인 별칭이 있을 수 있습니다. 이 필드에는 계정 ID 또는 별칭을 사용할 수 있습니다.
- 역할 이름 - 역할 이름은 대/소문자를 구분합니다. 계정 ID 및 역할 이름은 해당 역할에 대해 구성된 이름과 일치해야 합니다.

오류 메시지가 계속 나타나면 관리자에게 연락하여 이전 정보를 확인하십시오. 역할 신뢰 정책 또는 IAM 사용자 정책에 따라 액세스가 제한될 수 있습니다. 관리자는 이러한 정책에 대한 권한을 확인할 수 있습니다.

역할에 작업 수행을 허용하는 정책이 있지만 “액세스 거부”가 표시됩니다.

역할 세션이 세션 정책에 의해 제한되었을 수 있습니다. AWS STS 사용을 통해 프로그래밍 방식으로 임시 보안 자격 증명을 요청한 경우 인라인 또는 관리형 세션 정책을 전달할 수 있습니다. 세션 정책은 역할 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. `Policy` 파라미터를 사용하여 단일 JSON 인라인 세션 정책 문서를 전달할 수 있습니다. `PolicyArns` 파라미터를 사용하여 최대 10개까지 관리형 세션 정책을 지정할 수 있습니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 자격 증명 기반 정책의 교집합과 세션 정책입니다. 또는 관리자 또는 사용자 프로그램에서 임시 자격 증명을 제공한 경우 세션 정책에 포함되어 액세스를 제한했을 수 있습니다.

서비스에서 역할의 기본 정책 버전을 만들지 않았습니다.

서비스 역할은 서비스가 사용자를 대신하여 사용자 계정에서 작업을 수행하기 위해 수임하는 역할입니다. 일부 AWS 서비스 환경을 설정할 때, 서비스에서 맡을 역할을 정의해야 합니다. 경우에 따라 서비스는 IAM에서 서비스 역할과 해당 정책을 생성합니다. 서비스 역할 및 해당 정책을 IAM 내에서 수정하거나 삭제할 수 있지만, AWS에서는 이 옵션을 사용하지 않는 것이 좋습니다. 역할 및 정책은 해당 서비스에서만 사용할 수 있습니다. 정책을 편집하고 다른 환경을 설정하는 경우, 서비스가 동일한 역할 및 정책을 사용하려고 하면 작업이 실패할 수 있습니다.

예를 들어 AWS CodeBuild를 처음 사용하는 경우, 서비스가 `codebuild-RWBCore-service-role`라는 역할을 생성합니다 . 이 서비스 역할은 `codebuild-RWBCore-managed-policy`라는 정책을 사용합니다. 정책을 편집하면 새 버전이 생성되고 해당 버전이 기본 버전으로 저장됩니다. AWS CodeBuild에서 후속 작업을 수행하면 서비스에서 정책 업데이트를 시도할 수 있습니다. 이 경우 다음과 같은 오류가 나타납니다.

```
codebuild.amazon.com did not create the default version (V2) of the
codebuild-RWBCore-managed-policy policy that is attached to the codebuild-
RWBCore-service-role role. To continue, detach the policy from any other
identities and then delete the policy and the role.
```

이 오류가 발생하는 경우 서비스 작업을 계속하려면 먼저 IAM에서 변경을 수행해야 합니다. 먼저 기본 정책 버전을 V1로 설정하고 작업을 다시 시도하십시오. V1이 이전에 삭제되었거나 V1을 선택할 수 없는 경우, 기존 정책 및 역할을 정리하고 삭제합니다.

관리형 정책 편집에 대한 자세한 내용은 고객 관리형 정책 편집(콘솔) 단원을 참조하십시오. 정책 버전에 대한 자세한 내용은 IAM 정책 버전 관리 단원을 참조하십시오.

서비스 역할 및 해당 정책을 삭제하려면

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택합니다.
3. 정책 목록에서 삭제하려는 정책의 이름을 선택합니다.
4. 연결된 개체 탭을 선택하여 이 정책을 사용하는 IAM 사용자, 그룹 또는 역할을 확인합니다. 이러한 ID 중 하나라도 정책을 사용하는 경우 다음 작업을 완료합니다.

    a. 필요한 권한을 가진 관리형 정책을 새로 생성합니다. 작업 전후에 자격 증명에 동일한 권한이 있는지 확인하려면 기존 정책에서 JSON 정책 문서를 복사합니다. 그런 다음 관리형 정책을 새로 생성하고 JSON 편집기에서 정책 생성에 설명된 대로 JSON 문서를 붙여 넣습니다.

    b. 영향을 받는 각 ID에 대해 새 정책을 연결한 다음, 이전 정책을 분리합니다. 자세한 내용은 IAM 자격 증명 권한 추가 및 제거 단원을 참조하십시오.
5. 탐색 창에서 역할을 선택합니다.
6. 역할 목록에서 삭제하려는 역할의 이름을 선택합니다.
7. 신뢰 관계 탭을 선택하여 역할을 수임할 수 있는 엔터티를 확인합니다. 서비스 이외의 엔터티가 나열되면 다음 작업을 완료합니다.

    a. 이러한 엔터티를 신뢰하는새 역할을 생성합니다 .

    b. 이전 단계에서 생성한 정책입니다. 이 단계를 건너뛴 경우에는 지금 관리형 정책을 새로 생성합니다.

    c. 역할을 수임 중이었던 사람에게 더 이상 그렇게 할 수 없음을 알립니다. 그리고 새 역할을 수임하고 동일한 권한을 갖는 방법에 대한 정보를 제공합니다.
8. 정책을 삭제합니다.
9. 역할을 삭제합니다.

콘솔에 서비스 역할에 대한 사용 사례가 없음

일부 서비스의 경우 사용자를 대신하여 작업을 수행할 수 있는 서비스 권한을 부여하려면 서비스 역할을 수동으로 생성해야 합니다. 서비스가 IAM 콘솔에 나열되지 않는 경우에는 해당 서비스를 신뢰할 수 있는 보안 주체로 수동으로 나열해야 합니다. 사용 중인 서비스 또는 기능에 대한 설명서에 신뢰할 수 있는 보안 주체로 서비스를 나열하는 단계에 대한 지침이 없는 경우 페이지에 대한 피드백을 제공하세요.

서비스 역할을 수동으로 생성하려면 역할을 수임할 서비스의 서비스 보안 주체를 알고 있어야 합니다. 서비스 보안 주체는 서비스에 권한을 부여하는 데 사용되는 식별자입니다. 서비스 보안 주체는 서비스가 정의합니다.

다음을 확인하여 일부 서비스에 대한 서비스 주체를 찾을 수 있습니다.

1. AWS IAM으로 작업하는 서비스를 엽니다.
2. 서비스의 [서비스 연결 역할(Service-linked roles)] 열에 [예(Yes)]가 있는지 확인합니다.
3. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes(네) 링크를 선택합니다.
4. 서비스 보안 주체를 보려면 서비스 연결 역할 권한 섹션을 참조하세요.

AWS CLI 명령 또는 AWS API 작업을 사용하여 서비스 역할을 수동으로 생성할 수 있습니다. IAM 콘솔을 사용하여 서비스 역할을 수동으로 생성하려면 다음 작업을 완료합니다.

1. 계정 ID를 사용하여 IAM 역할을 생성합니다. 정책을 연결하거나 권한을 부여하지 마십시오. 세부 정보는 IAM 사용자에게 권한을 위임할 역할 생성을 참조하세요.
2. 역할을 열고 신뢰 관계를 편집합니다. 역할은 계정을 신뢰하는 것이 아니라 서비스를 신뢰해야 합니다. 예를 들어 다음 `Principal` 요소를 업데이트합니다.

   ```
   "Principal": { "AWS": "arn:aws:iam::123456789012:root" }
   ```

   보안 주체를 서비스에 대한 값(예: IAM)으로 변경합니다.

   ```
   "Principal": { "Service": "iam.amazonaws.com" }
   ```

3. 역할에 권한 정책을 연결하여 서비스에 필요한 권한을 추가합니다.
4. 사용 권한이 필요한 서비스로 돌아가서 설명서의 방법을 사용하여 서비스에 새 서비스 역할에 대해 알립니다.

IAM 및 Amazon EC2 문제 해결

다음 정보는 Amazon EC2와 관련된 IAM 문제를 해결하는 데 도움이 될 수 있습니다.

주제

- 인스턴스를 시작하려고 할 때 Amazon EC2 콘솔 IAM 역할 목록에서 역할이 보이지 않습니다.
- 제 인스턴스에 있는 자격 증명의 역할이 잘못되었습니다.

- AddRoleToInstanceProfile을 호출하려고 하면 AccessDenied 오류가 발생합니다.
- Amazon EC2 역할로 인스턴스를 시작하려고 하면 AccessDenied 오류가 발생합니다.
- 제 EC2 인스턴스의 임시 보안 자격 증명에 액세스할 수 없습니다.
- IAM 하위 트리에서 info 문서의 오류란 무엇인가요?

인스턴스를 시작하려고 할 때 Amazon EC2 콘솔 IAM 역할 목록에서 역할이 보이지 않습니다.

다음을 확인하세요.

- IAM 사용자로 로그인한 경우, `ListInstanceProfiles`를 호출할 권한이 있는지 확인하십시오. 역할 사용에 필요한 권한에 대한 자세한 내용은 Amazon EC2로 역할을 사용하는 데 필요한 권한의 내용을 참조하세요. 사용자에게 권한을 추가하는 방법에 대한 자세한 내용은 IAM 정책 관리을 참조하십시오.

  권한을 수정할 수 없는 경우, IAM을 사용할 수 있는 관리자에게 문의하여 권한을 업데이트해야 합니다.
- IAM CLI 또는 API를 사용하여 역할을 생성한 경우 다음을 확인하세요.
  - 인스턴스 프로파일을 생성하고 이 인스턴스 프로파일에 해당 역할을 추가하였습니다.
  - 역할과 인스턴스 프로파일에 같은 이름을 사용했습니다. 역할과 인스턴스 프로파일의 이름을 다르게 설정한 경우, Amazon EC2 콘솔에서 올바른 역할 이름을 볼 수 없습니다.

  Amazon EC2 콘솔의 IAM 역할 목록에는 역할 이름이 아니라 인스턴스 프로파일 이름이 나열되어 있습니다. 원하는 역할을 포함한 인스턴스 프로필 이름을 선택해야 합니다. 인스턴스 프로파일에 대한 자세한 내용은 인스턴스 프로파일 사용을 참조하십시오.

> **Note**
>
> IAM 콘솔을 사용하여 역할을 만드는 경우, 인스턴스 프로파일을 사용하지 않아도 됩니다. 인스턴스 프로파일은 IAM 콘솔에서 만드는 각 역할과 동일한 이름으로 생성되며, 역할은 해당 인스턴스 프로파일에 자동으로 추가됩니다. 하나의 인스턴스 프로파일은 하나의 IAM 역할만 포함할 수 있으며 이 제한은 늘릴 수 없습니다.

제 인스턴스에 있는 자격 증명의 역할이 잘못되었습니다.

인스턴스 프로파일의 역할이 최근에 교체되었을 수 있습니다. 그러한 경우 다음에 예정된 자동 자격 증명 교체 이후에 역할의 자격 증명을 사용할 수 있습니다.

변경을 적용하려면 인스턴스 프로파일 연결을 해제하고 나서 인스턴스 프로파일을 연결하거나, 인스턴스를 중지했다가 다시 시작합니다.

AddRoleToInstanceProfile을 호출하려고 하면 **AccessDenied** 오류가 발생합니다.

IAM 사용자로 요청을 하는 경우, 다음과 같은 권한이 있는지 확인합니다.

- 인스턴스 프로파일 ARN과 일치하는 리소스가 포함된 `iam:AddRoleToInstanceProfile`(예: `arn:aws:iam::999999999999:instance-profile/ExampleInstanceProfile`).

역할 사용에 필요한 권한에 대한 자세한 내용은 어떻게 시작할 수 있습니까?의 내용을 참조하세요. 사용자에게 권한을 추가하는 방법에 대한 자세한 내용은 IAM 정책 관리을 참조하십시오.

Amazon EC2 역할로 인스턴스를 시작하려고 하면 **AccessDenied** 오류가 발생합니다.

다음을 확인하세요.

- 인스턴스 프로필 없이 인스턴스를 시작합니다. 이를 통해 문제가 Amazon EC2 인스턴스의 IAM 역할로 제한되어 있는지 확인할 수 있습니다.
- IAM 사용자로 요청을 하는 경우, 다음과 같은 권한이 있는지 확인합니다.
  - 와일드카드 리소스("*")가 포함된 `ec2:RunInstances`
  - 역할 ARN과 일치하는 리소스가 포함된 `iam:PassRole`(예: `arn:aws:iam::999999999999:role/ExampleRoleName`)
- IAM `GetInstanceProfile` 작업을 호출하여 올바른 인스턴스 프로파일 이름 또는 올바른 인스턴스 프로파일 ARN을 사용 중인지 확인합니다. 자세한 내용은 Amazon EC2 인스턴스로 IAM 역할 사용을 참조하세요.
- IAM `GetInstanceProfile` 작업을 호출하여 인스턴스 프로필에 역할이 있는지 확인합니다. 인스턴스 프로파일이 비어 있으면 `AccessDenied` 오류가 발생합니다. 역할 만들기에 대한 자세한 내용은 IAM 역할 생성 단원을 참조하십시오.

역할 사용에 필요한 권한에 대한 자세한 내용은 어떻게 시작할 수 있습니까?의 내용을 참조하세요. 사용자에게 권한을 추가하는 방법에 대한 자세한 내용은 IAM 정책 관리을 참조하십시오.

제 EC2 인스턴스의 임시 보안 자격 증명에 액세스할 수 없습니다.

EC2 인스턴스에서 임시 보안 자격 증명에 액세스하려면 먼저 IAM 콘솔을 사용하여 역할을 생성해야 합니다. 그런 다음 해당 역할을 사용하는 EC2 인스턴스를 시작하고 실행 중인 인스턴스를 검사합니다. 자세한 내용은 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여의 어떻게 시작합니까?를 참조하십시오.

그래도 EC2 인스턴스에서 임시 보안 자격 증명에 액세스할 수 없는 경우 다음을 확인하십시오.

- 인스턴스 메타데이터 서비스(IMDS)의 다른 부분에는 액세스할 수 있습니까? 액세스할 수 없는 경우 IMDS로의 요청에 대한 액세스를 차단하는 방화벽 규칙이 없는지 확인하십시오.

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/
hostname; echo
```

- IMDS의 `iam` 하위 트리가 있습니까? 그렇지 않은 경우 EC2 `DescribeInstances` API 작업을 호출하거나 `aws ec2 describe-instances` CLI 명령을 사용하여 인스턴스에 IAM 인스턴스 프로파일이 연결되어 있는지 확인합니다.

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam;
 echo
```

- 오류가 있는지 IAM 하위 트리의 `info` 문서를 확인합니다. 오류가 있는 경우 자세한 내용은 IAM 하위 트리에서 `info` 문서의 오류란 무엇인가요?를 참조하십시오.

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam/
info; echo
```

IAM 하위 트리에서 **`info`** 문서의 오류란 무엇인가요?

`iam/info` 문서는 **`"Code":"InstanceProfileNotFound"`**를 나타냅니다.

IAM 인스턴스 프로파일이 삭제되었으므로 Amazon EC2에서 더 이상 인스턴스에 자격 증명을 제공할 수 없습니다. Amazon EC2 인스턴스에 올바른 인스턴스 프로파일을 연결해야 합니다.

해당 이름의 인스턴스 프로파일이 있는 경우, 원래 인스턴스 프로파일이 삭제되고 동일한 이름의 다른 인스턴스가 생성된 것이 아닌지 확인하십시오.

1. IAM `GetInstanceProfile` 작업을 호출하여 `InstanceProfileId`를 가져옵니다.
2. Amazon EC2 `DescribeInstances` 작업을 호출하여 인스턴스의 `IamInstanceProfileId`를 가져옵니다.
3. IAM 작업의 `InstanceProfileId`와 Amazon EC2 작업의 `IamInstanceProfileId`가 일치하는지 확인합니다.

ID가 다르면 인스턴스에 연결된 인스턴스 프로파일이 더 이상 유효하지 않습니다. 인스턴스에 올바른 인스턴스 프로파일을 연결해야 합니다.

iam/info 문서는 성공을 나타내지만 **"Message":"Instance Profile does not contain a role..."**을 나타냅니다.

역할이 IAM `RemoveRoleFromInstanceProfile` 작업에 의해 인스턴스 프로파일에서 제거되었습니다. IAM `AddRoleToInstanceProfile` 작업을 사용하여 인스턴스 프로파일에 역할을 연결할 수 있습니다. 역할의 자격 증명에 액세스하려면 다음에 예정된 새로 고침까지 기다려야 합니다.

변경을 적용하려면 인스턴스 프로파일 연결을 해제하고 나서 인스턴스 프로파일을 연결하거나, 인스턴스를 중지했다가 다시 시작합니다.

iam/security-credentials/[role-name] 문서는 **"Code":"AssumeRoleUnauthorizedAccess"**를 나타냅니다.

Amazon EC2에는 역할을 수임할 권한이 없습니다. 다음 예와 같이 역할을 수임할 권한은 해당 역할에 연결된 신뢰 정책에서 관리합니다. IAM `UpdateAssumeRolePolicy` API를 사용하여 신뢰 정책을 업데이트합니다.

```
{"Version": "2012-10-17","Statement": [{"Effect": "Allow","Principal": {"Service":
 ["ec2.amazonaws.com"]},"Action": ["sts:AssumeRole"]}]}
```

역할의 자격 증명에 액세스하려면 다음에 예정된 자동 새로 고침까지 기다려야 합니다.

변경을 적용하려면 인스턴스 프로파일 연결을 해제하고 나서 인스턴스 프로파일을 연결하거나, 인스턴스를 중지했다가 다시 시작합니다.

IAM 및 Amazon S3 문제 해결

이 문서의 정보를 사용하여 Amazon S3 및 IAM 작업 시 발생할 수 있는 문제를 해결하세요.

Amazon S3 버킷에 대한 익명 액세스 권한을 부여하는 방법은 무엇인가요?

`principal` 요소에 와일드카드(*)를 지정하는 Amazon S3 버킷 정책을 사용합니다. 이는 누구나 버킷에 액세스할 수 있다는 의미입니다. 익명 액세스를 통하면 누구나(AWS 계정이 없는 사용자 포함) 버킷에 액세스할 수 있게 됩니다. 예제 정책은 Amazon Simple Storage Service 사용 설명서의 Amazon S3 버킷 정책 예 사례를 참조하세요.

AWS 계정 루트 사용자로 로그인했습니다. 내 계정으로 Amazon S3 버킷에 액세스할 수 없는 이유는 무엇인가요?

IAM 및 Amazon S3에 대해 모든 권한을 가진 IAM 사용자가 있기도 합니다. IAM 사용자가 Amazon S3 버킷에 버킷 정책을 할당하고 루트 사용자를 보안 주체로 지정하지 않으면 루트 사용자의 버킷 액세스가 거부됩니다. 하지만 루트 사용자로 계속 버킷에 액세스할 수 있습니다. 이를 위해 버킷 정책을 수정하여 Amazon S3 콘솔 또는 AWS CLI에서 루트 사용자 액세스를 허용합니다. 다음 보안 주체를 사용하여 *123456789012*를 AWS 계정의 ID로 바꿉니다.

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

IAM과의 SAML 페더레이션 문제 해결

이 문서의 정보를 사용하여 AWS Identity and Access Management 연동 및 SAML 2.0 작업 시 발생할 수 있는 문제를 진단하고 해결할 수 있습니다.

주제

- 오류: 요청에 잘못된 SAML 응답이 포함되어 있습니다. 로그아웃하려면 여기를 클릭하십시오.
- 오류: AuthnResponse에 RoleSessionName 필요(서비스: AWSSecurityTokenService, 상태 코드: 400, 오류 코드: InvalidIdentityToken)
- 오류: sts:AssumeRoleWithSAML을 수행할 권한 없음(서비스: AWSSecurityTokenService, 상태 코드: 403, 오류 코드: AccessDenied)
- 오류: AuthnResponse의 RoleSessionName은 [a-zA-Z_0-9+=,.@-]{2,64}와 일치해야 함(서비스: AWSSecurityTokenService, 상태 코드: 400, 오류 코드: InvalidIdentityToken)

- 오류: 소스 자격 증명은 [a~zA~Z_0~9+=,.@-]{2,64}와 일치해야 하며 "aws:"로 시작할 수 없음(서비스: AWSSecurityTokenService, 상태 코드: 400, 오류 코드: InvalidIdentityToken)
- 오류: 유효하지 않은 응답 서명(서비스: AWSSecurityTokenService, 상태 코드: 400, 오류 코드: InvalidIdentityToken)
- 오류: 역할을 수임하지 못함: 지정한 공급자에 발행자가 없음(서비스: AWSOpenIdDiscoveryService, 상태 코드: 400, 오류 코드: AuthSamlInvalidSamlResponseException)
- 오류: 메타데이터를 구문 분석할 수 없습니다.
- 오류: 지정된 공급자가 존재하지 않습니다.
- 오류: 요청된 DurationSeconds가 이 역할에 대해 설정된 MaxSessionDuration을 초과합니다.
- 오류: 응답에 필수 대상이 포함되어 있지 않습니다.

오류: 요청에 잘못된 SAML 응답이 포함되어 있습니다. 로그아웃하려면 여기를 클릭하십시오.

이 오류는 자격 증명 공급자의 SAML 응답에 `Name`이 `https://aws.amazon.com/SAML/Attributes/Role`로 설정된 속성이 포함되지 않은 경우 발생할 수 있습니다. 이 속성은 하나 이상의 `AttributeValue` 요소를 포함해야 하며, 각 요소에 다음과 같은 문자열 쌍이 쉼표로 구분되어 있어야 합니다.

- 사용자를 매핑할 수 있는 역할의 ARN
- SAML 공급자의 ARN

자세한 내용은 인증 응답에 대한 SAML 어설션 구성 단원을 참조하십시오. 브라우저에서 SAML 응답을 보려면 브라우저에서 SAML 응답 보기의 단계를 따르십시오.

오류: AuthnResponse에 RoleSessionName 필요(서비스: AWSSecurityTokenService, 상태 코드: 400, 오류 코드: InvalidIdentityToken)

이 오류는 자격 증명 공급자의 SAML 응답에 `Name`이 `https://aws.amazon.com/SAML/Attributes/RoleSessionName`로 설정된 속성이 포함되지 않은 경우 발생할 수 있습니다. 속성 값은 사용자의 식별자이며, 일반적으로 사용자 ID 또는 이메일 주소입니다.

자세한 내용은 인증 응답에 대한 SAML 어설션 구성 단원을 참조하십시오. 브라우저에서 SAML 응답을 보려면 브라우저에서 SAML 응답 보기의 단계를 따르십시오.

오류: sts:AssumeRoleWithSAML을 수행할 권한 없음(서비스: AWSSecurityTokenService, 상태 코드: 403, 오류 코드: AccessDenied)

이 오류는 SAML 응답에 지정된 IAM 역할이 잘못 기재되었거나 존재하지 않는 경우 발생할 수 있습니다. 역할 이름은 대소문자를 구분하므로 역할 이름을 정확하게 사용하십시오. SAML 서비스 공급자 구성의 역할 이름을 올바르게 수정하십시오.

역할 신뢰 정책에 `sts:AssumeRoleWithSAML` 작업이 포함된 경우에만 액세스가 허용됩니다. `PrincipalTag` 속성을 사용하도록 SAML 어설션이 구성된 경우 신뢰 정책에도 `sts:TagSession` 작업이 포함되어야 합니다. 세션 태그에 대한 자세한 내용은 AWS STS에서 세션 태그 전달 섹션을 참조하세요.

이 오류는 사용자의 역할 신뢰 정책에 `sts:SetSourceIdentity` 권한이 없는 경우 발생할 수 있습니다. `SourceIdentity` 속성을 사용하도록 SAML 어설션이 구성된 경우 신뢰 정책에도 `sts:SetSourceIdentity` 작업이 포함되어야 합니다. 소스 자격 증명에 대한 자세한 내용은 위임된 역할로 수행한 작업 모니터링 및 제어 섹션을 참조하세요.

이 오류는 페더레이션 사용자가 역할을 수임할 권한이 없는 경우에도 발생할 수 있습니다. 역할에는 IAM SAML 자격 증명 공급자의 ARN을 `Principal`로 지정하는 신뢰 정책이 있어야 합니다. 또한 역할에는 어떤 사용자가 해당 역할을 수임할 수 있는지 제어하는 조건이 포함됩니다. 사용자는 조건의 요구 사항을 준수해야 합니다.

이 오류는 SAML 응답에 `Subject`가 포함된 `NameID`가 없는 경우에도 발생할 수 있습니다.

자세한 내용은 AWS에서 페더레이션 사용자에 대한 권한 수립 및 인증 응답에 대한 SAML 어설션 구성 섹션을 참조하세요. 브라우저에서 SAML 응답을 보려면 브라우저에서 SAML 응답 보기의 단계를 따르십시오.

오류: AuthnResponse의 RoleSessionName은 [a-zA-Z_0-9+=,.@-]{2,64}와 일치해야 함(서비스: AWSSecurityTokenService, 상태 코드: 400, 오류 코드: InvalidIdentityToken)

이 오류는 `RoleSessionName` 속성 값이 너무 길거나 유효하지 않은 문자가 포함된 경우 발생할 수 있습니다. 유효한 최대 길이는 64자입니다.

자세한 내용은 인증 응답에 대한 SAML 어설션 구성 단원을 참조하십시오. 브라우저에서 SAML 응답을 보려면 브라우저에서 SAML 응답 보기의 단계를 따르십시오.

오류: 소스 자격 증명은 [a~zA~Z_0~9+=,.@-]{2,64}와 일치해야 하며 "aws:"로 시작할 수 없음(서비스: AWSSecurityTokenService, 상태 코드: 400, 오류 코드: InvalidIdentityToken)

이 오류는 `sourceIdentity` 속성 값이 너무 길거나 유효하지 않은 문자가 포함된 경우 발생할 수 있습니다. 유효한 최대 길이는 64자입니다. 소스 자격 증명에 대한 자세한 내용은 위임된 역할로 수행한 작업 모니터링 및 제어 섹션을 참조하세요.

SAML 어설션을 생성하는 방법에 대한 자세한 내용은 인증 응답에 대한 SAML 어설션 구성 섹션을 참조하세요. 브라우저에서 SAML 응답을 보려면 브라우저에서 SAML 응답 보기의 단계를 따르십시오.

오류: 유효하지 않은 응답 서명(서비스: AWSSecurityTokenService, 상태 코드: 400, 오류 코드: InvalidIdentityToken)

이 오류는 자격 증명 공급자의 연동 메타데이터가 IAM 자격 증명 공급자의 메타데이터와 일치하지 않는 경우 발생할 수 있습니다. 예를 들어, 만료된 인증서를 업데이트하기 위해 자격 증명 서비스 공급자의 메타데이터 파일이 변경되었을 수 있습니다. 이 경우, 자격 증명 서비스 공급자의 업데이트된 SAML 메타데이터 파일을 다운로드합니다. 그런 다음 `aws iam update-saml-provider` 교차 플랫폼 CLI 명령 또는 `Update-IAMSAMLProvider` PowerShell cmdlet을 통해 IAM에서 정의한 AWS 자격 증명 공급자 엔터티에 이를 업데이트합니다.

오류: 역할을 수임하지 못함: 지정한 공급자에 발행자가 없음(서비스: AWSOpenIdDiscoveryService, 상태 코드: 400, 오류 코드: AuthSamlInvalidSamlResponseException)

이 오류는 업로드한 연동 메타데이터 파일에 선언되어 있는 발행자와 SAML 응답의 발행자가 일치하지 않는 경우 발생할 수 있습니다. 메타데이터 파일은 IAM에서 자격 증명 공급자를 생성할 때 AWS에 업로드되었습니다.

오류: 메타데이터를 구문 분석할 수 없습니다.

이 오류는 메타데이터 파일이 적절한 형식이 아닌 경우에 발생할 수 있습니다.

AWS Management Console에서 SAML 자격 증명 공급자를 생성하거나 관리할 때, 사용자의 자격 증명 공급자에서 SAML 메타데이터 문서를 가져와야 합니다.

이 메타데이터 파일에는 발급자 이름, 만료 정보 및 IdP에서 가져온 SAML 인증 응답(어설션)을 확인하는 데 사용할 수 있는 키가 포함되어 있습니다. 메타데이터 파일은 바이트 순서 표시(BOM)가 없는

UTF-8 형식으로 인코딩되어야 합니다. BOM을 제거하려면 Notepad++와 같은 텍스트 편집 도구를 사용해 파일을 UTF-8로 인코딩합니다.

SAML 메타데이터 문서의 일부로 포함된 x.509 인증서는 1,024비트 이상의 키를 사용해야 합니다. 또한 x.509 인증서에는 반복되는 확장이 없어야 합니다. 확장을 사용할 수 있지만 확장은 인증서에 한 번만 나타날 수 있습니다. x.509 인증서가 두 조건 중 하나를 충족하지 못하면 IdP 생성에 실패하고 “메타데이터를 구문 분석할 수 없음” 오류를 반환합니다.

SAML V2.0 Metadata Interoperability Profile Version 1.0의 정의에 따라, IAM은 메타데이터 문서의 X.509 인증서 만료를 평가하거나 이에 대해 조치를 취하지 않습니다.

오류: 지정된 공급자가 존재하지 않습니다.

이 오류는 SAML 어설션의 공급자 이름이 IAM의 공급자 이름과 일치하지 않는 경우 발생할 수 있습니다. 공급자 이름 보기에 대한 자세한 내용은 IAM에서 SAML ID 공급자 생성 단원을 참조하십시오.

오류: 요청된 DurationSeconds가 이 역할에 대해 설정된 MaxSessionDuration을 초과합니다.

이 오류는 AWS CLI 또는 API에서 역할을 위임한 경우 발생할 수 있습니다.

assume-role-with-saml CLI 또는 AssumeRoleWithSAML API 작업을 사용하여 역할을 위임하는 경우 `DurationSeconds` 파라미터의 값을 지정할 수 있습니다. 값을 900초(15분)에서 해당 역할에 대한 최대 세션 지속 시간 설정까지 지정할 수 있습니다. 이 설정보다 높게 값을 지정하면 작업에 실패합니다. 예를 들어 세션 기간으로 12시간을 지정했는데 관리자가 최대 세션 기간으로 6시간을 설정하면 작업에 실패합니다. 역할에 대한 최댓값을 확인하는 방법을 알아보려면 역할의 최대 세션 기간 업데이트 섹션을 참조하세요.

오류: 응답에 필수 대상이 포함되어 있지 않습니다.

이 오류는 SAML 구성에서 대상 URL과 자격 증명 공급자가 일치하지 않는 경우 발생할 수 있습니다. ID 제공업체(IdP) 신뢰 당사자 식별자가 SAML 구성에 제공된 대상 URL(엔터티 ID)과 정확히 일치하는지 확인하세요.

IAM과 다른 AWS 서비스와의 작동 방식

IAM은 IAM 리소스를 관리하는 데 사용하는 기본 AWS 서비스이지만 다른 모든 AWS 서비스는 IAM과 연동하여 계정의 리소스에 대한 액세스를 제어합니다.

- AWS CloudFormation

  AWS CloudFormation은 AWS CloudFormation 템플릿의 일부로 IAM 리소스를 정의하고 관리할 수 있도록 하여 IAM과 통합합니다. 프로비저닝한 다른 AWS 리소스에 필요한 IAM 권한을 지정하는 데 AWS CloudFormation을 사용할 수 있습니다. AWS CloudFormation은 또한 IAM 역할을 사용하여 AWS 인프라를 프로비저닝하고 관리하는 데 필요한 자격 증명 관리를 지원하고, 드리프트 감지 기능은 IAM 구성의 무결성 유지에 도움이 됩니다.

- AWS CloudShell

  AWS CloudShell에 액세스할 때 인증 및 권한 부여는 IAM을 통해 처리됩니다. AWS CloudShell은 사용자 또는 계정에 할당된 IAM 역할의 컨텍스트 내에서 실행됩니다. AWS CloudShell을 시작하면 할당된 IAM 역할을 기반으로 임시 보안 자격 증명이 자동으로 생성됩니다.

- AWS SDK

  AWS SDK는 IAM과 연동하여 인증 및 권한 부여 프로세스를 처리하고, AWS 자격 증명을 관리하고, IAM에 정의된 권한 및 정책을 준수하여 애플리케이션이 권한 부여된 리소스에만 액세스할 수 있도록 합니다. SDK는 임시 보안 자격 증명을 획득 및 사용하고 애플리케이션 운영에 필요한 권한을 검증하는 메커니즘을 제공합니다.

IAM과 함께 사용할 수 있는 AWS 서비스 목록과 해당 서비스가 지원하는 IAM 기능은 AWS IAM으로 작업하는 서비스 섹션을 참조하세요.

AWS CloudFormation을 사용하여 IAM 리소스 생성

AWS Identity and Access Management는 리소스 및 인프라를 생성하고 관리하는 데 소요되는 시간을 줄일 수 있도록 AWS 리소스를 모델링하고 설정하는 데 도움이 되는 서비스인 AWS CloudFormation과 통합됩니다. 원하는 모든 AWS 리소스(예: 액세스 키, 그룹, 그룹 정책, 인스턴스 프로파일, 관리형 정책, OIDC 공급자, 인라인 정책, 역할 정책, SAML 공급자, 서버 인증서, 서비스 연결 역할, 사용자 (및 그룹에 사용자 추가), 사용자 정책, 가상 MFA 디바이스)를 설명하는 템플릿을 만들면 AWS CloudFormation이 이러한 리소스를 프로비저닝하고 구성합니다.

AWS CloudFormation을 사용할 때 템플릿을 재사용하여 IAM 리소스를 일관되고 반복적으로 설정할 수 있습니다. 리소스를 한 번 설명한 후 여러 AWS 계정및 리전에서 동일한 리소스를 반복적으로 프로비저닝할 수 있습니다.

IAM 및 AWS CloudFormation 템플릿

IAM 및 관련 서비스에 대한 리소스를 프로비저닝하고 구성하려면 AWS CloudFormation 템플릿을 이해해야 합니다. 템플릿은 JSON 또는 YAML로 서식 지정된 텍스트 파일입니다. 이 템플릿은 AWS CloudFormation 스택에서 프로비저닝할 리소스에 대해 설명합니다. JSON 또는 YAML에 익숙하지 않은 경우 AWS CloudFormation Designer를 사용하면 AWS CloudFormation 템플릿을 시작하는 데 도움이 됩니다. 자세한 내용은 AWS CloudFormation 사용 설명서에서 AWS CloudFormation Designer이란 무엇입니까?를 참조하세요.

IAM은 AWS CloudFormation에서 액세스 키, 그룹, 그룹 정책, 인스턴스 프로파일, 관리형 정책, OIDC 공급자, 인라인 정책, 역할 정책, SAML 공급자, 서버 인증서, 서비스 연결 역할, 사용자 (및 그룹에 사용자 추가), 사용자 정책, 가상 MFA 디바이스를 생성하는 것을 지원합니다. IAM 리소스에 대한 JSON 및 YAML 템플릿의 예를 비롯한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS Identity and Access Management 리소스 유형 참조를 참조하세요.

역할 및 관리형 정책과 같은 관련 리소스를 생성하는 템플릿을 생성할 수도 있습니다.

AWS CloudFormation에 대해 자세히 알아보기

AWS CloudFormation에 대한 자세한 내용은 다음 리소스를 참조하세요.

- AWS CloudFormation
- AWS CloudFormation 사용 설명서
- AWS CloudFormation API 참조
- AWS CloudFormation 명령줄 인터페이스 사용 설명서

AWS CloudShell을 사용하여 AWS Identity and Access Management에서 작업

AWS CloudShell은 브라우저 기반의 사전 인증된 쉘로, AWS Management Console에서 바로 시작할 수 있습니다. 원하는 쉘(Bash, PowerShell 또는 Z 쉘)을 사용하여 AWS 서비스(AWS Identity and Access Management 포함)에 대해 AWS CLI 명령을 실행할 수 있습니다. 또한 명령줄 도구를 다운로드하거나 설치할 필요 없이 이 작업을 수행할 수 있습니다.

AWS Management Console에서 AWS CloudShell을 실행합니다. 그러면 콘솔에 로그인할 때 사용한 AWS 보안 인증을 새 셸 세션에서 자동으로 사용할 수 있습니다. 이러한 AWS CloudShell 사용자 사전 인증을 통해 AWS CLI 버전 2(셸의 컴퓨팅 환경에 사전 설치됨)를 사용하여 IAM과 같은 AWS 서비스와 상호 작용할 때 보안 인증 구성을 건너뛸 수 있습니다.

AWS CloudShell에 대한 IAM 권한 획득

관리자는 AWS Identity and Access Management에서 제공하는 액세스 관리 리소스를 사용하여 IAM 사용자에게 AWS CloudShell에 액세스하고 환경 기능을 사용할 수 있는 권한을 부여할 수 있습니다.

관리자가 사용자에게 액세스 권한을 부여하는 가장 빠른 방법은 AWS 관리형 정책을 사용하는 것입니다. AWS 관리형 정책은 AWS에서 생성 및 관리하는 독립 실행형 정책입니다. 다음과 같은 CloudShell에 대한 AWS 관리형 정책을 IAM 자격 증명에 연결할 수 있습니다.

- `AWSCloudShellFullAccess`: 모든 기능에 대한 전체 액세스 권한과 함께 AWS CloudShell을 사용할 수 있는 권한을 부여합니다.

IAM 사용자가 AWS CloudShell에서 수행할 수 있는 작업의 범위를 제한하려면 `AWSCloudShellFullAccess` 관리형 정책을 템플릿으로 사용하는 사용자 지정 정책을 생성할 수 있습니다. CloudShell에서 사용자가 사용할 수 있는 작업을 제한하는 방법에 대한 자세한 내용은 AWS CloudShell 사용 설명서의 Managing AWS CloudShell access and usage with IAM policies를 참조하세요.

IAM과의 상호 작용

AWS Management Console에서 AWS CloudShell을 실행한 후에 명령줄 인터페이스를 사용하여 즉시 IAM과의 상호 작용을 시작할 수 있습니다.

> **Note**
>
> AWS CloudShell에서 AWS CLI을 사용하는 경우 추가 리소스를 다운로드하거나 설치할 필요가 없습니다. 또한 셸 내에서 이미 인증되었기 때문에 직접 호출을 하기 전에 보안 인증을 구성하지 않아도 됩니다.

AWS CloudShell을 사용하여 IAM 그룹 생성 및 그룹에 IAM 사용자 추가

다음 예제에서는 CloudShell을 사용하여 IAM 그룹을 생성하고, 그룹에 IAM 사용자를 추가한 다음, 명령이 성공했는지 확인합니다.

1. AWS Management Console에서는 탐색 표시줄에 제공되는 다음 옵션을 선택하여 CloudShell을 실행할 수 있습니다.

   - CloudShell 아이콘을 선택합니다.
   - 검색 상자에 'cloudshell'을 입력하고 CloudShell 옵션을 선택합니다.

2. IAM 그룹을 생성하려면 CloudShell 명령줄에 다음 명령을 입력합니다. 이 예제에서는 그룹 이름을 east_coast로 지정합니다.

```
aws iam create-group --group-name east_coast
```

   직접 호출이 성공하면 명령줄에 다음 출력과 비슷한 서비스의 응답이 표시됩니다.

```
      {
         "Group": {
             "Path": "/",
             "GroupName": "east_coast",
             "GroupId": "AGPAYBDBW4JBY3EXAMPLE",
             "Arn": "arn:aws:iam::111122223333:group/east_coast",
             "CreateDate": "2023-09-11T21:02:21+00:00"
          }
      }
```

3. 생성한 그룹에 사용자를 추가하려면 다음 명령을 사용하여 그룹 이름과 사용자 이름을 지정합니다. 이 예제에서는 그룹 이름을 east_coast로, 사용자 이름을 johdoe로 지정합니다.

```
aws iam add-user-to-group --group-name east_coast --user-name johndoe
```

4. 사용자가 그룹에 속해 있는지 확인하려면 다음 명령을 사용하여 그룹 이름을 지정합니다. 이 예제에서는 east_coast 그룹을 계속 사용합니다.

```
aws iam get-group --group-name east_coast
```

   직접 호출이 성공하면 명령줄에 다음 출력과 비슷한 서비스의 응답이 표시됩니다.

```
     {
       "Users": [
         {
```

```
            "Path": "/",
            "UserName": "johndoe",
            "UserId": "AIDAYBDBW4JBXGEXAMPLE",
            "Arn": "arn:aws:iam::552108220995:user/johndoe",
            "CreateDate": "2023-09-11T20:43:14+00:00",
            "PasswordLastUsed": "2023-09-11T20:59:14+00:00"
        }
      ],
      "Group": {
            "Path": "/",
            "GroupName": "east_coast",
            "GroupId": "AGPAYBDBW4JBY3EXAMPLE",
            "Arn": "arn:aws:iam::111122223333:group/east_coast",
            "CreateDate": "2023-09-11T21:02:21+00:00"
       }
    }
```


AWS SDK와 함께 이 서비스 사용

다양한 프로그래밍 언어에 대해 AWS 소프트웨어 개발 키트(SDK)을 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예시 및 설명서를 제공합니다.

SDK 설명서	코드 예시
AWS SDK for C++	AWS SDK for C++ 코드 예시
AWS CLI	AWS CLI 코드 예시
AWS SDK for Go	AWS SDK for Go 코드 예시
AWS SDK for Java	AWS SDK for Java 코드 예시
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예시
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예시
AWS SDK for .NET	AWS SDK for .NET 코드 예시

SDK 설명서	코드 예시
AWS SDK for PHP	AWS SDK for PHP 코드 예시
AWS Tools for PowerShell	Tools for PowerShell 코드 예시
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예시
AWS SDK for Ruby	AWS SDK for Ruby 코드 예시
AWS SDK for Rust	AWS SDK for Rust 코드 예시
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예시
AWS SDK for Swift	AWS SDK for Swift 코드 예시

이 서비스 관련 예시는 AWS SDK를 사용한 IAM용 코드 예제를 참조하세요.

예제 사용 가능 여부

필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

AWS Identity and Access Management에 대한 참조 정보

이 섹션의 주제를 통해 IAM 및 AWS STS의 다양한 측면에 대한 자세한 참조 자료를 찾아보세요.

주제

Amazon 리소스 이름(ARN)으로 AWS 리소스를 식별합니다.

Amazon 리소스 이름(ARN)은 AWS 리소스를 고유하게 식별합니다. IAM 정책, Amazon Relational Database Service(RDS) 태그 및 API 호출과 같은 모든 AWS에서 리소스를 명료하게 지정해야 하는 경우 ARN이 필요합니다.

ARN 형식

다음은 ARN의 일반적인 형식입니다. 구체적인 형식은 리소스에 따라 다릅니다. ARN을 사용하려면 *###* 텍스트를 리소스별 정보로 바꿉니다. 일부 리소스의 ARN에서는 리전, 계정 ID 또는 리전과 계정 ID가 모두 생략됩니다.

```
arn:partition:service:region:account-id:resource-id
arn:partition:service:region:account-id:resource-type/resource-id
arn:partition:service:region:account-id:resource-type:resource-id
```

partition

리소스가 있는 파티션입니다. 파티션은 AWS 리전의 그룹입니다. 각 AWS 계정은 하나의 파티션으로 범위가 지정됩니다.

지원되는 파티션은 다음과 같습니다.

- aws - AWS 리전

- aws-cn - 중국 리전
- aws-us-gov - AWS GovCloud (US) 리전

service

AWS 제품을 식별하는 서비스 네임스페이스입니다.

region

리전 코드. 예를 들어 미국 동부(오하이오)의 경우 us-east-2입니다. 리전 코드 목록은 AWS 일반 참조의 리전 엔드포인트를 참조하세요.

account-id

리소스를 소유한 AWS 계정의 ID(하이픈 없음)입니다. 예: 123456789012.

resource-type

리소스 유형. Virtual Private Cloud(VPC)의 vpc를 예로 들 수 있습니다.

resource-id

리소스 식별자입니다. 리소스의 이름, 리소스의 ID 또는 리소스 경로입니다. 일부 리소스 식별자에는 상위 리소스(sub-resource-type/parent-resource/sub-resource) 또는 버전과 같은 식별자(resource-type:resource-name:qualifier)를 포함할 수 있습니다.

예

IAM 사용자

arn:aws:iam::*123456789012*:user/*johndoe*

SNS 주제

arn:aws:sns:*us-east-1*:*123456789012*:*example-sns-topic-name*

VPC

arn:aws:ec2:*us-east-1*:*123456789012*:vpc/*vpc-0e9801d129EXAMPLE*

리소스의 ARN 형식 조회

ARN의 정확한 형식은 서비스 및 리소스 유형에 따라 달라집니다. 일부 리소스 ARN에는 경로, 변수 또는 와일드카드가 포함될 수 있습니다. 특정 AWS 리소스의 ARN 형식을 조회하려면 서비스 승인 참조를 열고 해당 서비스의 페이지를 연 다음 리소스 유형 테이블로 이동합니다.

ARN의 경로

리소스 ARN에 경로를 포함할 수 있습니다. 예를 들어, Amazon S3에서 리소스 식별자는 슬래시(/)을 포함하여 경로를 구성할 수 있는 객체 이름입니다. 마찬가지로, IAM 사용자 이름과 그룹 이름에 경로를 포함할 수 있습니다. IAM 경로에는 영숫자와 슬래시(/), 더하기(+), 등호(=), 쉼표(,), 마침표(.), @(@), 밑줄 표시(_), 하이픈(-) 문자만 사용할 수 있습니다.

경로에서 와일드카드 사용

경로에는 와일드카드 문자 즉, 별표(*)를 포함할 수 있습니다. 예를 들어 IAM 정책을 작성 중인 경우 다음과 같은 와일드카드를 사용하는 product_1234 경로를 가진 모든 IAM 사용자를 지정할 수 있습니다.

```
arn:aws:iam::123456789012:user/Development/product_1234/*
```

마찬가지로 다음 예와 같이 모든 사용자를 의미하는 user/* 또는 모든 그룹을 의미하는 group/*을 지정할 수 있습니다.

```
"Resource":"arn:aws:iam::123456789012:user/*"
"Resource":"arn:aws:iam::123456789012:group/*"
```

다음 예에서는 리소스 이름에 경로를 포함하는 Amazon S3 버킷용 ARN을 보여줍니다.

```
arn:aws:s3:::my_corporate_bucket/*
arn:aws:s3:::my-corporate-bucket/Development/*
```

잘못된 와일드카드 사용

리소스 유형을 지정하는 ARN 부분에 와일드카드를 사용할 수 없습니다(예: IAM ARN의 user 용어). 예를 들어 다음은 허용되지 않습니다.

```
arn:aws:iam::123456789012:u*   <== not allowed
```

IAM 식별자

IAM은 사용자, 사용자 그룹, 역할, 정책 및 서버 인증서에 대해 몇 가지 다른 식별자를 사용합니다. 이 단원에서는 그러한 식별자와 각 식별자를 사용하는 경우를 설명합니다.

주제

표시 이름 및 경로

사용자, 역할, 사용자 그룹 또는 정책을 생성하거나 서버 인증서를 업로드할 때 표시 이름을 지정합니다. 예를 들면 Bob, TestApp1, Developers, ManageCredentialsPermissions 또는 ProdServerCert입니다.

IAM API 또는 AWS Command Line Interface(AWS CLI)를 사용하여 IAM 리소스를 생성하는 경우, 선택적 경로를 추가할 수 있습니다. 하나의 경로를 사용하거나, 하나의 폴더 구조인 것처럼 여러 경로를 중첩할 수 있습니다. 예를 들면 중첩된 경로 `/division_abc/subdivision_xyz/product_1234/engineering/`를 사용하여 귀하 회사의 조직 구조를 일치시킬 수 있습니다. 그런 다음 정책을 생성하여 그 경로의 모든 사용자가 정책 시뮬레이터 API에 액세스할 수 있도록 허용할 수 있습니다. 정책을 보려면 IAM: 사용자 경로를 바탕으로 정책 시뮬레이터 API 액세스 섹션을 참조하세요. 알기 쉬운 이름을 지정하는 방법에 대한 자세한 내용은 사용자 API 설명서를 참조하세요. 경로를 사용하는 방법의 추가 예제는 IAM ARN 섹션을 참조하세요.

AWS CloudFormation을 사용하여 리소스를 생성할 때 사용자, 사용자 그룹 및 역할과 고객 관리형 정책에 대한 경로를 지정할 수 있습니다.

동일한 경로에 사용자 및 사용자 그룹이 있는 경우 IAM은 사용자를 해당 사용자 그룹에 자동으로 배치하지 않습니다. 예를 들어, Developers 사용자 그룹을 생성하고 이 그룹의 경로를 `/division_abc/subdivision_xyz/product_1234/engineering/`(으)로 지정할 수 있습니다. Bob이라는 사용자를 생성하고 그에게 동일한 경로를 추가했다고 해서 Bob이 Developers 사용자 그룹에 자동으로 추가되지는 않습니다. IAM은 경로에 따라 사용자 또는 사용자 그룹 간에 경계를 적용하지 않습니다. 해당 리소스에 대한 권한이 있다는 가정하에, 경로가 다른 사용자가 동일한 리소스를 사용할 수 있습니다. AWS 계정의 IAM 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 IAM 및 AWS STS 할당량 섹션을 참조하세요.

IAM ARN

대부분의 리소스에는 표시 이름이 있습니다(예를 들어, Bob(이)라는 사용자 또는 `Developers`(이)라는 사용자 그룹). 그러나 권한 정책 언어에는 다음과 같은 Amazon 리소스 이름(ARN) 형식을 사용하여 하나 이상의 리소스를 지정해야 합니다.

```
arn:partition:service:region:account:resource
```

위치:

- `partition`은 리소스가 위치하는 파티션을 식별합니다. 표준 AWS 리전에서 파티션은 `aws`입니다. 리소스가 다른 파티션에 있는 경우 파티션은 `aws-`*`partitionname`*입니다. 예를 들어 중국(베이징) 리전에 있는 리소스의 파티션은 `aws-cn`입니다. 다른 파티션의 계정 간에 액세스 권한을 위임할 수 없습니다.
- `service`에서는 AWS 제품을 식별합니다. IAM 리소스는 항상 `iam`을 사용합니다.
- `region`은 리소스의 리전을 식별합니다. IAM 리소스의 경우 항상 공백입니다.
- `account`은(는) 하이픈이 없는 AWS 계정 ID를 지정합니다.
- `resource`는 특정 리소스를 이름으로 식별합니다.

다음 구문을 사용하여 IAM 및 AWS STS ARN을 지정할 수 있습니다. IAM 리소스가 글로벌이기 때문에 ARN의 리전 부분은 공백입니다.

구문:

```
arn:aws:iam::account:root
arn:aws:iam::account:user/user-name-with-path
arn:aws:iam::account:group/group-name-with-path
arn:aws:iam::account:role/role-name-with-path
arn:aws:iam::account:policy/policy-name-with-path
arn:aws:iam::account:instance-profile/instance-profile-name-with-path
arn:aws:sts::account:federated-user/user-name
arn:aws:sts::account:assumed-role/role-name/role-session-name
arn:aws:sts::account:self
arn:aws:iam::account:mfa/virtual-device-name-with-path
arn:aws:iam::account:u2f/u2f-token-id
arn:aws:iam::account:server-certificate/certificate-name-with-path
arn:aws:iam::account:saml-provider/provider-name
arn:aws:iam::account:oidc-provider/provider-name
```

다음과 같은 많은 예에는 ARN의 리소스 부분에 경로가 포함됩니다. AWS Management Console에서는 경로를 생성하거나 조작할 수 없습니다. 경로를 사용하려면 AWS API, AWS CLI 또는 Tools for Windows PowerShell을 사용하여 리소스와 함께 작업해야 합니다.

예시:

```
arn:aws:iam::123456789012:root
arn:aws:iam::123456789012:user/JohnDoe
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/JaneDoe
```

```
arn:aws:iam::123456789012:group/Developers
arn:aws:iam::123456789012:group/division_abc/subdivision_xyz/product_A/Developers
arn:aws:iam::123456789012:role/S3Access
arn:aws:iam::123456789012:role/application_abc/component_xyz/RDSAccess
arn:aws:iam::123456789012:role/aws-service-role/access-analyzer.amazonaws.com/
AWSServiceRoleForAccessAnalyzer
arn:aws:iam::123456789012:role/service-role/QuickSightAction
arn:aws:iam::123456789012:policy/UsersManageOwnCredentials
arn:aws:iam::123456789012:policy/division_abc/subdivision_xyz/UsersManageOwnCredentials
arn:aws:iam::123456789012:instance-profile/Webserver
arn:aws:sts::123456789012:federated-user/JohnDoe
arn:aws:sts::123456789012:assumed-role/Accounting-Role/JaneDoe
arn:aws:sts::123456789012:self
arn:aws:iam::123456789012:mfa/JaneDoeMFA
arn:aws:iam::123456789012:u2f/user/JohnDoe/default (U2F security key)
arn:aws:iam::123456789012:server-certificate/ProdServerCert
arn:aws:iam::123456789012:server-certificate/division_abc/subdivision_xyz/
ProdServerCert
arn:aws:iam::123456789012:saml-provider/ADFSProvider
arn:aws:iam::123456789012:oidc-provider/GoogleProvider
arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-west-2.amazonaws.com/id/
a1b2c3d4567890abcdefEXAMPLE11111
arn:aws:iam::123456789012:oidc-provider/server.example.org
```

다음 예제에서는 다양한 유형의 IAM 및 AWS STS 리소스에 대한 ARN 형식을 이해하는 데 도움이 되는 자세한 정보를 제공합니다.

- 계정의 IAM 사용자:

> **Note**
>
> 각 IAM 사용자 이름은 고유합니다. 사용자 이름은 로그인 프로세스와 같이 사용자의 경우 대소문자를 구분하지 않지만 정책에서 사용하거나 ARN의 일부로 사용하는 경우에는 대소문자를 구분합니다.

```
arn:aws:iam::123456789012:user/JohnDoe
```

- 조직 차트를 반영하는 경로를 갖는 다른 사용자:

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/JaneDoe
```

- IAM 사용자 그룹:

```
arn:aws:iam::123456789012:group/Developers
```

- 경로를 포함하는 IAM 사용자 그룹:

```
arn:aws:iam::123456789012:group/division_abc/subdivision_xyz/product_A/Developers
```

- IAM 역할:

```
arn:aws:iam::123456789012:role/S3Access
```

- 서비스 연결 역할:

```
arn:aws:iam::123456789012:role/aws-service-role/access-analyzer.amazonaws.com/
AWSServiceRoleForAccessAnalyzer
```

- 서비스 역할:

```
arn:aws:iam::123456789012:role/service-role/QuickSightAction
```

- 관리형 정책:

```
arn:aws:iam::123456789012:policy/ManageCredentialsPermissions
```

- Amazon EC2 인스턴스와 연결될 수 있는 인스턴스 프로파일:

```
arn:aws:iam::123456789012:instance-profile/Webserver
```

- IAM에서 "Paulo"로 식별되는 페더레이션 사용자:

```
arn:aws:sts::123456789012:federated-user/Paulo
```

- 역할 세션 이름 'Mary'로 역할 'Accounting-Role'을 수임하는 누군가의 활성 세션:

```
arn:aws:sts::123456789012:assumed-role/Accounting-Role/Mary
```

- 호출 세션에서 작동하는 AWS STS SetContext API와 같은 API 직접 호출에서 리소스로 사용될 때 호출자 자신의 세션을 나타냅니다.

```
arn:aws:sts::123456789012:self
```

- 사용자 이름 Jorge에 할당된 멀티 팩터 인증 디바이스:

```
arn:aws:iam::123456789012:mfa/Jorge
```

- 서버 인증서:

```
arn:aws:iam::123456789012:server-certificate/ProdServerCert
```

- 조직 차트를 반영하는 경로를 갖는 서버 인증서:

```
arn:aws:iam::123456789012:server-certificate/division_abc/subdivision_xyz/
ProdServerCert
```

- 자격 증명 공급자(SAML 및 OIDC):

```
arn:aws:iam::123456789012:saml-provider/ADFSProvider
arn:aws:iam::123456789012:oidc-provider/GoogleProvider
arn:aws:iam::123456789012:oidc-provider/server.example.org
```

- Amazon EKS OIDC ID 공급자 URL을 나타내는 경로가 있는 OIDC 자격 증명 공급자:

```
arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-west-2.amazonaws.com/id/
a1b2c3d4567890abcdefEXAMPLE11111
```

또 다른 중요한 ARN은 루트 사용자 ARN입니다. IAM 리소스는 아니지만 이 ARN의 형식을 잘 알고 있어야 합니다. 이 ARN은 종종 리소스 기반 정책의 Principal 요소에 사용됩니다.

- AWS 계정에는 다음이 표시됩니다.

```
arn:aws:iam::123456789012:root
```

다음 예에서는 Richard가 액세스 키를 관리할 수 있도록 그에게 할당할 수 있는 정책을 보여줍니다. 리소스는 IAM 사용자 Richard입니다.

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

```
            "Effect": "Allow",
            "Action": [
                "iam:*AccessKey*",
                "iam:GetUser"
            ],
            "Resource": "arn:aws:iam::*:user/division_abc/subdivision_xyz/Richard"
        },
        {
            "Sid": "ListForConsole",
            "Effect": "Allow",
            "Action": "iam:ListUsers",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> ARN을 사용하여 IAM 정책의 리소스를 식별하는 경우 정책 변수를 포함할 수 있습니다. 정책 변수에는 ARN의 일부로 런타임 정보(예: 사용자 이름)에 대한 자리표시자가 포함될 수 있습니다. 자세한 내용은 IAM 정책 요소: 변수 및 태그 섹션을 참조하세요.

ARN에서 와일드카드 및 경로 사용

ARN의 ### 부분에 와일드카드를 사용하여 여러 사용자, 사용자 그룹 또는 정책을 지정할 수 있습니다. 예를 들어, product_1234를 작업하는 모든 사용자를 지정하려면 다음을 사용합니다.

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/*
```

이름이 문자열 app_(으)로 시작하는 사용자가 있는 경우, 다음 ARN을 사용하여 모두 참조할 수 있습니다.

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/app_*
```

AWS 계정의 모든 사용자, 사용자 그룹 또는 정책을 지정하려면, ARN의 `user/`, `group/` 또는 `policy/` 부분 다음에 각각 와일드카드를 사용합니다.

```
arn:aws:iam::123456789012:user/*
arn:aws:iam::123456789012:group/*
```

```
arn:aws:iam::123456789012:policy/*
```

사용자 arn:aws:iam::111122223333:user/*에 대해 다음 ARN을 지정하는 경우 다음 두 예제 모두와 일치합니다.

```
arn:aws:iam::111122223333:user/JohnDoe
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/JaneDoe
```

그러나 사용자 arn:aws:iam::111122223333:user/division_abc*에 대해 다음 ARN을 지정하는 경우 두 번째 예제와는 일치하지만 첫 번째 예제와는 일치하지 않습니다.

```
arn:aws:iam::111122223333:user/JohnDoe
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/JaneDoe
```

ARN의 user/, group/ 또는 policy/ 부분에 와일드카드를 사용하지 마십시오. 예를 들어 IAM은 다음을 허용하지 않습니다.

```
arn:aws:iam::123456789012:u*
```

Example 프로젝트 기반 사용자 그룹의 경로 및 ARN 사용의 예

AWS Management Console에서는 경로를 생성하거나 조작할 수 없습니다. 경로를 사용하려면 AWS API, AWS CLI 또는 Tools for Windows PowerShell을 사용하여 리소스와 함께 작업해야 합니다.

이 예제에서 Marketing_Admin 사용자 그룹의 Jules가 /marketing/ 경로에 프로젝트 기반 사용자 그룹을 생성합니다. Jules는 회사의 다른 파트에 있는 사용자를 해당 사용자 그룹에 할당합니다. 이 예는 사용자의 경로가 그가 속한 사용자 그룹과 관련되지 않는다는 점을 보여줍니다.

마케팅 그룹에는 이들이 출시할 신제품이 있으므로 Jules는 /marketing/ 경로에 Widget_Launch라는 새 사용자 그룹을 생성합니다. 그런 다음 Jules는 해당 사용자 그룹에 다음과 같은 정책을 할당합니다. 이 정책은 사용자 그룹에 이 특정 출시에 지정된 example_bucket 부분의 객체에 대한 액세스 권한을 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::example_bucket/marketing/newproductlaunch/widget/*"
    },
```

```
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket*",
      "Resource": "arn:aws:s3:::example_bucket",
      "Condition": {"StringLike": {"s3:prefix": "marketing/newproductlaunch/widget/*"}}
    }
  ]
}
```

그런 다음 Jules는 해당 사용자 그룹에 이 출시와 관련된 사용자를 할당합니다. 여기에는 /marketing/ 경로의 Patricia와 Eli가 포함됩니다. 또한 /sales/ 경로의 Chris와 Chloe, /legal/ 경로의 Alice와 Jim이 포함됩니다.

고유 식별자

IAM에서 사용자, 사용자 그룹, 역할, 정책, 인스턴스 프로파일 또는 서버 인증서를 생성할 때, 각 리소스에 고유 ID를 할당합니다. 고유 ID는 다음과 같습니다.

AIDAJQABLZS4A3QDU576Q

대부분의 경우 IAM 리소스로 작업할 때 표시 이름과 ARN을 사용합니다. 이렇게 하면 특정 리소스의 고유 ID를 알아야 할 필요가 없습니다. 그러나 표시 이름을 사용하는 것이 적절치 않은 경우에는 고유 ID를 사용하는 것이 유용합니다.

한 예로써 AWS 계정에서 표시 이름을 재사용하는 경우를 살펴보겠습니다. 계정 내에서 사용자, 사용자 그룹, 역할 또는 정책의 표시 이름은 고유해야 합니다. 예를 들어, John이라는 IAM 사용자를 생성할 수 있습니다. 회사에서 Amazon S3를 사용하고 있고 각 직원에 대한 폴더가 포함된 버킷이 있습니다. IAM 사용자 John은 사용자가 버킷에서 자신의 폴더에만 액세스하도록 허용하는 권한이 있는 User-S3-Access라는 IAM 사용자 그룹의 구성원입니다. IAM 사용자가 사용자의 표시 이름을 사용하여 S3에 있는 자신의 버킷 객체에 액세스하도록 허용하는 자격 증명 기반 정책을 생성하는 방법의 예제는 Amazon S3: IAM 사용자가 프로그램 방식으로 콘솔에서 자신의 S3 홈 디렉터리에 액세스하도록 허용 섹션을 참조하세요.

John이라는 직원이 퇴사하여 John이라는 해당 IAM 사용자를 삭제한다고 가정해 봅시다. 그러나 이후 John이라는 또 다른 직원이 입사하여 John이라는 새 IAM 사용자를 생성합니다. 기존 IAM 사용자 그룹 User-S3-Access에 John이라는 새 IAM 사용자를 추가합니다. 사용자 그룹에 연결된 정책에서 IAM 사용자 표시 이름 John을 지정하는 경우 이 정책은 새 John이 이전 John이 남긴 정보에 액세스할 수 있도록 허용합니다.

일반적으로 정책의 리소스에 대해 고유한 ID 대신 ARN을 지정하는 것이 좋습니다. 그러나 이전에 삭제한 표시 이름을 재사용하는 새 IAM 사용자를 생성한다 하더라도 모든 IAM 사용자는 고유 ID를 갖습니

다. 이 예에서 이전 IAM 사용자 John과 새 IAM 사용자 John은 서로 다른 고유 ID를 갖습니다. 사용자 이름뿐만 아니라 고유 ID별로 액세스 권한을 부여하는 리소스 기반 정책을 생성할 수 있습니다. 이렇게 하면 직원에게 없어야 하는 정보에 대한 액세스 권한을 실수로 부여할 가능성이 줄어듭니다.

다음 예에서는 리소스 기반 정책의 Principal 요소에 고유 ID를 지정하는 방법을 보여줍니다.

```
"Principal": {
  "AWS": [
    "arn:aws:iam::111122223333:role/role-name",
    "AIDACKCEVSQ6C2EXAMPLE",
    "AROADBQP57FF2AEXAMPLE"
  }
```

다음 예에서는 전역 조건 키 aws:userid의 Condition 요소에 고유 ID를 지정하는 방법을 보여줍니다.

```
"Condition": {
    "StringLike": {
      "aws:userId": [
        "AIDACKCEVSQ6C2EXAMPLE",
        "AROADBQP57FF2AEXAMPLE:role-session-name",
        "AROA1234567890EXAMPLE:*",
        "111122223333"
      ]
    }
  }
```

사용자 ID가 유용한 또 다른 예는 IAM 사용자 또는 역할 정보의 데이터베이스(또는 다른 저장소)를 유지하는 경우입니다. 고유 ID는 사용자가 생성하는 각 IAM 사용자 또는 역할에 대해 고유 식별자를 제공할 수 있습니다. 이전 예제에서와 같이 시간이 지남에 따라 이름을 재사용하는 IAM 사용자 또는 역할이 있는 경우에도 마찬가지입니다.

고유 ID 접두사에 대한 이해

IAM에서는 다음과 같은 접두사를 사용해 각 고유 ID가 적용되는 리소스의 유형을 표시합니다. 접두사는 생성 시기에 따라 달라질 수 있습니다.

접두사	리소스 유형
ABIA	AWS STS 서비스 보유자 토큰

접두사	리소스 유형
ACCA	컨텍스트별 자격 증명
AGPA	사용자 그룹
AIDA	IAM 사용자
AIPA	Amazon EC2 인스턴스 프로파일
AKIA	액세스 키
ANPA	관리형 정책
ANVA	관리 정책 내 버전
APKA	퍼블릭 키
AROA	역할
ASCA	인증서
ASIA	임시(AWS STS) 액세스 키 ID는 이 접두사를 사용하지만 보안 액세스 키 및 세션 토큰과 함께만 고유합니다.

고유 식별자 가져오기

IAM 리소스의 고유 ID는 IAM 콘솔에서 제공되지 않습니다. 고유 ID를 가져오기 위해 다음과 같은 AWS CLI 명령 또는 IAM API 호출을 사용할 수 있습니다.

AWS CLI:

- get-caller-identity
- get-group
- get-role
- get-user
- get-policy
- get-instance-profile

- get-server-certificate

IAM API:

- GetCallerIdentity
- GetGroup
- GetRole
- GetUser
- GetPolicy
- GetInstanceProfile
- GetServerCertificate

IAM 및 AWS STS 할당량

AWS Identity and Access Management(IAM) 및 AWS Security Token Service(STS)에는 객체의 크기를 제한하는 할당량이 있습니다. 이러한 할당량은 객체의 이름을 지정하는 방법, 생성할 수 있는 객체 수, 객체를 전달할 때 사용할 수 있는 문자 수에 영향을 미칩니다.

> **Note**
>
> IAM 사용량 및 할당량에 관한 계정 수준 정보를 가져오려면 GetAccountSummary API 작업 또는 get-account-summary AWS CLI 명령을 사용합니다.

IAM 이름 요구 사항

IAM 이름에 대한 요구 사항 및 제한 사항은 다음과 같습니다.

- 정책 설명서에는 수평 탭(U+0009), 라인 피드(U+000A), 캐리지 리턴(U+000D), 그리고 U+0020 ~ U+00FF 범위의 문자 등 유니코드 문자만 넣을 수 있습니다.
- 사용자, 그룹, 역할, 정책, 인스턴스 프로파일, 서버 인증서 및 경로 이름은 더하기(+), 등호(=), 콤마(,), 마침표(.), at(@), 밑줄(_), 하이픈(-)을 포함하는 영숫자여야 합니다. 경로 이름은 슬래시(/)로 시작하고 끝나야 합니다.
- 사용자, 그룹, 역할 및 인스턴스 프로파일의 이름은 계정 내에서 고유해야 합니다. 대소문자는 구분하지 않습니다. 예를 들어 그룹 **`ADMINS`**와 그룹 **`admins`**를 모두 생성할 수는 없습니다.

- 타사에서 역할을 맡기 위해 사용하는 외부 ID 값은 최소 2자 이상, 최대 1,224자 이상이어야 합니다. 이 값은 공백 없이 영숫자여야 합니다. 이 값은 더하기(+), 등호(=), 쉼표(,) 마침표(.), 기호(@), 콜론(:), 슬래시(/) 및 하이픈(-)과 같은 기호도 포함할 수 있습니다. 외부 ID에 대한 자세한 내용은 타사가 소유한 AWS 계정에 액세스을 참조하세요.
- 인라인 정책의 정책 이름은 이러한 정책이 포함된 사용자, 그룹 또는 역할에 대해 고유해야 합니다. 이름에는 예약된 문자인 역슬래시(\), 슬래시(/), 별표(*), 물음표(?), 공백을 제외한 라틴어 기본(ASCII) 문자가 포함될 수 있습니다. 이러한 문자는 RFC 3986, 섹션 2.2에 따라 예약됩니다.
- 사용자 암호(로그인 프로필)에는 라틴어 기본(ASCII) 문자가 포함될 수 있습니다.
- AWS 계정 ID 별칭은 AWS 제품 전체에서 고유해야 하며, 다음 DNS 명명 규칙을 따르는 영숫자여야 합니다. 별칭은 소문자여야 하며, 하이픈으로 시작하거나 끝나면 안 되고, 2개의 하이픈이 연속으로 있으면 안 되고, 12자리 숫자는 안 됩니다.

로마자 기본(ASCII) 문자 목록을 보려면 의회 도서관 로마자 기본(ASCII) 코드 표를 확인하세요.

IAM 객체 할당량

AWS에서 제한이라고도 하는 할당량은 AWS 계정의 리소스, 작업, 항목의 최댓값입니다. Service Quotas를 사용하여 IAM 할당량을 관리합니다.

IAM 서비스 엔드포인트와 서비스 할당량의 목록은 AWS 일반 참조의 AWS Identity and Access Management endpoints and quotas를 참조하세요.

할당량 증가 요청

1. AWS 로그인 사용 설명서의 AWS에 로그인하는 방법 항목에 설명된 대로 사용자 유형에 맞는 로그인 절차에 따라 AWS Management Console에 로그인합니다.
2. Service Quotas 콘솔을 엽니다.
3. 탐색 창에서 AWS 서비스를 선택합니다.
4. 탐색 모음에서 US East (N. Virginia) 리전을 선택합니다. 그런 다음 **IAM**을 검색합니다.
5. AWS Identity and Access Management(IAM)를 선택하고 할당량을 선택한 다음, 지침에 따라 할당량 증가를 요청합니다.

자세한 내용은 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요.

Service Quotas 콘솔을 사용하여 IAM 할당량 증가를 요청하는 방법의 예를 보려면 다음 동영상을 시청하세요.

Service Quotas 콘솔을 사용하여 IAM 할당량 증가를 요청합니다.

조정 가능한 IAM 할당량에 대한 기본 할당량 증가를 요청할 수 있습니다. 최대 maximum quota개까지 요청이 자동으로 승인되고 몇 분 내에 완료됩니다.

다음 표에는 할당량 증가 영역을 자동으로 승인할 수 있는 리소스가 나와 있습니다.

Resource	기본 할당량	최대 할당량
계정별 고객 관리형 정책	1500	5000
계정당 그룹	300	500
계정당 인스턴스 프로파일 수	1000	5000
역할당 관리형 정책 수	10	20
사용자당 관리형 정책 수	10	20
역할 신뢰 정책 길이	2048자	4096자
계정당 역할 수	1000	5000
계정당 서버 인증서 수	20	1000

IAM Access Analyzer 할당량

IAM Access Analyzer 서비스 엔드포인트와 서비스 할당량의 목록은 AWS 일반 참조의 IAM Access Analyzer endpoints and quotas를 참조하세요.

IAM Roles Anywhere 할당량

IAM Roles Anywhere 서비스 엔드포인트와 서비스 할당량의 목록은 AWS 일반 참조의 AWS Identity and Access Management Roles Anywhere endpoints and quotas를 참조하세요.

IAM 및 STS 문자 제한

다음은 IAM 및 AWS STS에 대한 최대 문자 수와 크기 제한입니다. 다음 제한에 대해 증가를 요청할 수 없습니다.

설명	Limit
AWS 계정 ID에 대한 별칭	3~63자
인라인 정책:	원하는 만큼의 인라인 정책을 IAM 사용자, 역할 또는 그룹에게 추가할 수 있습니다. 단, 엔터티당 총 누적 정책 크기(모든 인라인 정책의 합)은 다음 제한을 초과할 수 없습니다. • 사용자 정책 크기는 2,048자를 초과할 수 없습니다. • 역할 정책 크기는 10,240자를 초과할 수 없습니다. • 그룹 정책 크기는 5,120자를 초과할 수 없습니다. **Note** IAM은 이러한 제한을 기준으로 정책의 크기를 계산할 때 공백을 계산하지 않습니다.
관리형 정책	• 각 관리 정책의 크기는 6,144자를 초과할 수 없습니다. **Note** IAM은 이 제한을 기준으로 정책의 크기를 계산할 때 공백을 계산하지 않습니다.
그룹 이름	128자
인스턴스 프로파일 이름	128자
로그인 프로필의 암호	1~128자

설명	Limit
경로	512자
정책 이름	128자
역할 이름	64자 **Important** 그러나 AWS Management Console에서 역할 전환 기능이 있는 역할을 사용하려면 `Path`와 `RoleName`을 합해 64자를 초과할 수 없습니다.
역할 세션 기간	12시간 AWS CLI 또는 API에서 역할을 위임할 때 `duration-seconds` CLI 파라미터 또는 `DurationSeconds` API 파라미터를 사용해 더 긴 역할 세션을 요청할 수 있습니다. 값을 900초(15분)에서 역할에 대한 최대 세션 지속 시간 설정(1~12시간)까지 지정할 수 있습니다. `DurationSeconds` 파라미터의 값을 지정하지 않으면 보안 자격 증명이 한 시간 동안 유효하게 됩니다. 콘솔에서 역할을 전환하는 IAM 사용자에게는 최대 세션 기간 또는 사용자 세션의 남은 시간 중 더 적은 시간이 부여됩니다. 최대 세션 기간 설정은 AWS 서비스에서 수임하는 세션을 제한하지 않습니다. 역할에 대한 최댓값을 확인하는 방법을 알아보려면 역할의 최대 세션 기간 업데이트 섹션을 참조하세요.
역할 세션 이름	64자

설명	Limit
역할 세션 정책	• 전달된 JSON 정책 문서의 크기와 전달된 모든 관리형 정책 ARN 문자를 합친 크기는 2,048자를 초과할 수 없습니다. • 세션을 생성할 때 최대 10개의 관리형 정책 ARN을 전달할 수 있습니다. • 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 경우 하나의 JSON 정책 문서만 전달할 수 있습니다. • 또한 AWS 변환은 전달된 세션 정책 및 세션 태그를 별도의 제한이 있는 이진 형식으로 압축합니다. `PackedPolicySize` 응답 요소는 요청에 대한 정책 및 태그가 상위 크기 제한과 얼마나 가까운지를 백분율로 나타냅니다. • AWS CLI 또는 AWS API를 사용하여 세션 정책을 전달하는 것이 좋습니다. AWS Management Console에서는 압축된 정책에 콘솔 세션 정보를 추가할 수 있습니다.
역할 세션 태그	• 세션 태그는 128자의 태그 키 제한과 256자의 태그 값 제한을 충족해야 합니다. • 최대 50개의 세션 태그를 전달할 수 있습니다. • AWS 변환은 전달된 세션 정책과 세션 태그를 별도의 제한이 있는 압축된 이진 형식으로 압축합니다. AWS CLI 또는 AWS API를 사용하여 세션 태그를 전달할 수 있습니다. `PackedPolicySize` 응답 요소는 요청에 대한 정책 및 태그가 상위 크기 제한과 얼마나 가까운지를 백분율로 나타냅니다.

설명	Limit
SAML 인증 응답 base64로 인코딩됨	100,000자 이 문자 제한은 `assume-role-with-saml` CLI 또는 `AssumeRoleWithSAML` API 작업에 적용됩니다.
태그 키	128자 이 문자 제한은 IAM 리소스의 태그와 세션 태그에 적용됩니다.
태그 값	256자 이 문자 제한은 IAM 리소스의 태그와 세션 태그에 적용됩니다. 태그 값의 길이는 0자, 즉 비어있을 수 있습니다.
IAM이(가) 생성한 고유 ID	128자. 예: • `AIDA`로 시작되는 사용자 ID • `AGPA`로 시작되는 그룹 ID • `AROA`로 시작되는 역할 ID • `ANPA`로 시작되는 관리형 정책 ID • `ASCA`로 시작되는 서버 인증서 ID **Note** 이는 포괄적인 목록을 제공하기 위한 것이 아니며 특정 유형의 ID가 지정된 문자 조합으로만 시작됨을 보장하지도 않습니다.
사용자 이름	64자

인터페이스 VPC 엔드포인트

Amazon Virtual Private Cloud(VPC)를 사용하여 AWS 리소스를 호스트하는 경우, VPC와 AWS Identity and Access Management(IAM) 또는 AWS Security Token Service(AWS STS) 간에 프라이빗 연결을 설정할 수 있습니다. 이 연결을 사용하면 IAM 또는 AWS STS가 퍼블릭 인터넷을 통하지 않고 VPC의 리소스와 통신하게 할 수 있습니다.

Amazon VPC란 사용자가 정의한 가상 네트워크에서 AWS 리소스를 시작할 때 사용할 수 있는 AWS 서비스입니다. VPC를 사용하여 IP 주소 범위, 서브넷, 라우팅 테이블, 네트워크 게이트웨이 등의 네트워크 설정을 제어할 수 있습니다. VPC를 IAM 또는 AWS STS에 연결하려면 각 서비스에 대해 인터페이스 VPC 엔드포인트를 정의하세요. 이 엔드포인트를 이용하면 인터넷 게이트웨이나 NAT(네트워크 주소 변환) 인스턴스 또는 VPN 연결 없이도 IAM 또는 AWS STS에 안정적이고 확장 가능하게 연결됩니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC란 무엇인가요?를 참조하세요.

인터페이스 VPC 엔드포인트는 프라이빗 IP 주소와 함께 탄력적 네트워크 인터페이스를 사용하여 AWS 서비스 간 프라이빗 통신을 사용할 수 있는 AWS 기술인 AWS PrivateLink에 의해 구동됩니다. 자세한 내용은 AWS 서비스를 위한 AWS PrivateLink를 참조하세요.

다음은 Amazon VPC 사용자를 위한 정보입니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC 시작하기를 참조하세요.

가용성

현재 IAM이 VPC 엔드포인트를 지원하는 리전은 다음과 같습니다.

- 미국 동부(버지니아 북부)
- 중국(베이징)

현재 AWS STS가 VPC 엔드포인트를 지원하는 리전은 다음과 같습니다.

- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오레곤)
- 아프리카(케이프타운)
- 아시아 태평양(홍콩)
- 아시아 태평양(하이데라바드)

- 아시아 태평양(자카르타)
- 아시아 태평양(멜버른)
- 아시아 태평양(뭄바이)
- 아시아 태평양(오사카)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 캐나다 서부(캘거리)
- 중국(베이징)
- 중국(닝샤)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(밀라노)
- 유럽(파리)
- 유럽(스페인)
- 유럽(스톡홀름)
- 유럽(취리히)
- 이스라엘(텔아비브)
- 중동(바레인)
- 중동(UAE)
- 남아메리카(상파울루)
- AWS GovCloud(미국 동부)
- AWS GovCloud(미국 서부)

IAM에 대한 VPC 엔드포인트 생성

VPC에서 IAM을 사용하기 시작하려면 IAM에 대한 인터페이스 VPC 엔드포인트를 생성합니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 VPC 엔드포인트를 사용하여 AWS 서비스에 액세스를 참조하세요.

IAM은 글로벌 서비스이므로 IAM용 인터페이스 VPC 엔드포인트는 IAM 컨트롤 플레인이 위치한 리전에서만 생성할 수 있습니다. 상용 AWS 리전에서 IAM 컨트롤 플레인은 미국 동부(버지니아 북부) 리전에 있습니다. IAM용 VPC 엔드포인트를 지원하는 AWS 리전의 목록은 가용성 섹션을 참조하세요. IAM 컨트롤 플레인에 대한 자세한 내용은 AWS Identity and Access Management의 복원성 섹션을 참조하세요.

VPC가 IAM 컨트롤 플레인 리전과 다른 리전에 있는 경우 다른 리전에서 IAM 인터페이스 VPC 엔드포인트에 액세스를 허용하도록 AWS Transit Gateway를 사용해야 합니다.

> **Note**
>
> VPC 피어링 연결은 피어링된 VPC 간에 트래픽을 라우팅할 수도 있지만, 이 방법은 VPC 수가 많은 경우 잘 확장되지 않습니다. VPC 피어링 대신 확장 가능한 중앙 허브를 통해 VPC 및 온프레미스 네트워크 관리를 개선하는 AWS Transit Gateway 피어링 연결을 사용하는 것이 좋습니다. VPC 피어링 연결에 대한 자세한 내용은 Amazon VPC 피어링 설명서의 Work with VPC peering connections를 참조하세요.

AWS Transit Gateway를 사용하여 다른 리전의 VPC에서 IAM 인터페이스 VPC 엔드포인트에 액세스하려면

1. 전송 게이트웨이를 생성하거나 기존 전송 게이트웨이를 사용하여 Virtual Private Cloud(VPC)와 상호 연결합니다. 각 리전에서 전송 게이트웨이는 필수입니다. 자세한 내용은 AWS Transit Gateway 가이드의 전송 게이트웨이 생성을 참조하세요.
2. 전송 게이트웨이 VPC 연결을 생성하여 각 VPC를 전송 게이트웨이에 연결합니다. 자세한 내용은 AWS Transit Gateway 가이드의 VPC에 Transit gateway Attachment 생성을 참조하세요.
3. 전송 게이트웨이 VPC 피어링 연결을 생성하여 피어링된 VPC 간에 트래픽을 라우팅합니다. 자세한 내용은 AWS Transit Gateway 가이드의 피어링 연결 생성을 참조하세요.

AWS STS에 대한 VPC 엔드포인트 생성

VPC에서 AWS STS를 사용하기 시작하려면 AWS STS에 대한 인터페이스 VPC 엔드포인트를 생성합니다. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 VPC 엔드포인트를 사용하여 AWS 서비스에 액세스를 참조하세요.

VPC 엔드포인트를 만든 후 해당 리전의 엔드포인트를 사용하여 AWS STS 요청을 보내야 합니다. AWS STS에서는 `setRegion` 및 `setEndpoint` 메서드를 모두 사용하여 리전 엔드포인트에 호출할 것을 권장합니다. 아시아 태평양(홍콩)과 같이 수동으로 활성화된 리전의 경우 `setRegion` 메서드만을 사용할 수 있습니다. 이 경우 호출은 STS 리전 엔드포인트로 전달됩니다. 리전을 수동으로 활성화하는 방법을 알아보려면 AWS 일반 참조의 AWS 리전 관리를 참조하세요. 기본적으로 활성화된 리전에 대해 `setRegion` 메서드를 사용하는 경우 호출은 `https://sts.amazonaws.com` 의 전역 엔드포인트로 전달됩니다.

리전의 엔드포인트를 사용할 경우, AWS STS는 퍼블릭 엔드포인트 또는 프라이빗 인터페이스 VPC 엔드포인트 중 사용 중인 엔드포인트를 사용하여 다른 AWS 서비스를 호출합니다. 예를 들어 AWS STS를 위한 인터페이스 VPC 엔드포인트를 만들어 VPC에 있는 리소스의 AWS STS의 임시 자격 증명을 이미 요청한 경우를 예로 들어 보겠습니다. 이 경우 이러한 자격 증명은 기본적으로 인터페이스 VPC 엔드포인트를 통합니다. AWS STS 사용을 통한 리전 요청에 대한 자세한 내용은 AWS STS에서 AWS 리전 관리 섹션을 참조하세요.

AWS IAM으로 작업하는 서비스

아래 나열된 AWS 서비스는 알파벳에 따라 그룹화되며, 지원되는 IAM 기능에 대한 정보를 포함합니다.

- 서비스 – 서비스의 이름을 선택하여 해당 서비스의 IAM 권한 부여 및 액세스에 대한 AWS 문서를 볼 수 있습니다.
- 작업 – 정책에서 개별 작업을 지정할 수 있습니다. 서비스에서 이 기능을 지원하지 않는 경우 시각적 편집기에서 모든 작업(All actions)이 선택됩니다. JSON 정책 문서의 * 요소에서 `Action`를 사용해야 합니다. 각 서비스의 작업 목록을 보려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요.
- 리소스 수준 권한 – ARN을 사용하여 정책에서 개별 리소스를 지정할 수 있습니다. 서비스에서 이 기능을 지원하지 않는 경우 정책 시각적 편집기에서 모든 리소스(All resources)가 선택됩니다. JSON 정책 문서의 * 요소에서 `Resource`를 사용해야 합니다. `List*` 작업과 같은 일부 작업은 ARN 지정을 지원하지 않습니다. 여러 리소스를 반환하기로 설계되었기 때문입니다. 서비스가 일부 리소스에 대해 이 기능을 지원하지만 다른 리소스는 지원하지 않는 경우 표에 부분(Partial)으로 표시됩니다. 자세한 정보는 서비스에 대한 문서 섹션을 참조하세요.

- 리소스 기반 정책 – 리소스 기반 정책을 서비스 내 리소스에 연결할 수 있습니다. 리소스 기반 정책은 해당 리소스에 액세스할 수 있는 IAM 자격 증명을 지정하는 `Principal` 요소를 포함합니다. 자세한 내용은 자격 증명 기반 정책 및 리소스 기반 정책 단원을 참조하세요.
- ABAC(태그 기반 권한 부여) - 태그를 기반으로 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다. 서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우 값은 부분적입니다. 태그와 같은 속성을 기반으로 권한을 정의하는 방법에 대한 자세한 내용은 ABAC 권한 부여를 통한 속성 기반 권한 정의 섹션을 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 ABAC(속성 기반 액세스 제어) 사용을 참조하세요.
- 임시 자격 증명 - IAM Identity Center를 사용하여 로그인하거나 콘솔에서 역할을 전환할 때 받거나 AWS CLI 또는 AWS API에서 AWS STS를 사용하여 생성하는 단기 자격 증명을 사용할 수 있습니다. 값이 [아니요(No)]인 서비스에는 장기 IAM 사용자 자격 증명을 사용하는 동안에만 액세스할 수 있습니다. 여기에는 사용자 이름 및 암호 또는 사용자 액세스 키가 포함됩니다. 자세한 내용은 IAM의 임시 보안 자격 증명 섹션을 참조하세요.
- 서비스 연결 역할 – 서비스 연결 역할은 사용자를 대신하여 다른 서비스의 리소스에 액세스할 수 있는 서비스 권한을 부여하는 특별한 유형의 서비스 역할입니다. 이러한 역할을 지원하는 서비스에 대한 설명서를 보려면 예 또는 일부 링크를 선택합니다. 이 열은 서비스가 표준 서비스 역할을 사용하는지 아닌지 나타내지 않습니다. 자세한 내용은 서비스 연결 역할을 참조하세요.
- 추가 정보 – 서비스가 기능을 완전히 지원하지 않는 경우 항목에 대한 각주를 검토하여 제한 사항과 관련 정보의 링크를 볼 수 있습니다.

IAM으로 작업하는 서비스

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Account Management	예	예	아니요	아니요	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Activate Console	예	아니요	아니요	아니요	예	아니요
AWS Amplify 관리자	예	예	아니요	아니요	예	아니요
AWS Amplify	예	예	아니요	부분적	예	아니요
AWS Amplify UI 빌더	예	예	아니요	예	예	아니요
Amazon MSK 클러스터용 Apache Kafka API	예	예	아니요	아니요	예	아니요
Amazon API Gateway	예	예	예	아니요	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon API Gateway Management	예	예	아니요	예	예	아니요
Amazon API Gateway Management V2	예	예	아니요	예	예	아니요
AWS 앱 스튜디오	예	아니요	아니요	아니요	예	아니요
AWS App2Container	예	아니요	아니요	아니요	예	아니요
AWS AppConfig	예	예	아니요	예	예	아니요
AWS AppFabric	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon AppFlow	예	예	아니요	예	예	아니요
Amazon AppIntegrations	예	예	아니요	예	예	예
Application Auto Scaling	예	예	아니요	예	예	예
AWS Application Cost Profiler	예	아니요	아니요	아니요	예	아니요
AWS Application Discovery Arsenal	예	아니요	아니요	아니요	예	아니요
AWS Application Discovery Service	예	아니요	아니요	아니요	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Application Migration Service	예	예	아니요	예	예	예
AWS Application Transformation Service	예	아니요	아니요	아니요	예	아니요
AWS App Mesh	예	예	아니요	예	예	예
AWS App Mesh 평가판	예	예	아니요	아니요	예	예
AWS App Runner	예	예	아니요	예	예	예
Amazon AppStream 2.0	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS AppSync	예	예	아니요	예	예	아니요
AWS Artifact	예	예	아니요	아니요	예	아니요
Amazon Athena	예	예	아니요	예	예	아니요
AWS Audit Manager	예	예	아니요	예	예	예
AWS Auto Scaling	예	아니요	아니요	아니요	예	예
AWS B2B Data Interchange	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Backup	예	예	예	예	예	예
AWS Backup 게이트웨이	예	예	아니요	예	예	아니요
AWS Backup 스토리지	예	아니요	아니요	아니요	예	아니요
AWS Batch	예	부분적	아니요	예	예	예
Amazon Bedrock	예	예	아니요	예	예	아니요
AWS Billing and Cost Management	예	아니요	아니요	아니요	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Billing and Cost Management 데이터 내보내기	예	예	아니요	예	예	아니요
AWS Billing Conductor	예	예	아니요	예	예	아니요
Amazon Braket	예	예	아니요	예	예	예
AWS Budget Service	예	예	아니요	아니요	아니요	아니요
AWS BugBust	예	예	아니요	예	예	예
AWS Certificate Manager(ACM)	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Chatbot	예	예	아니요	아니요	예	예
Amazon Chime	예	예	아니요	예	예	예
AWS Clean Rooms	예	예	아니요	예	예	아니요
AWS Clean Rooms ML	예	예	아니요	예	예	아니요
AWS Client VPN	예	예	아니요	아니요	예	예
AWS Cloud9	예	예	예	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS 클라우드 제어 API	예	아니요	아니요	아니요	예	아니요
Amazon Cloud Directory	예	예	아니요	아니요	예	아니요
AWS CloudFormation	예	예	아니요	예	예	아니요
Amazon CloudFront	예	예	아니요	부분적	예	일부(정보)
Amazon CloudFront KeyValueStore	예	예	아니요	아니요	예	아니요
AWS CloudHSM	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Cloud Map	예	예	아니요	예	예	아니요
Amazon CloudSearch	예	예	아니요	아니요	예	아니요
AWS CloudShell	예	예	아니요	아니요	예	아니요
AWS CloudTrail	예	예	일부(정보)	예	예	예
AWS CloudTrail 데이터	예	예	아니요	예	예	아니요
Amazon CloudWatch	예	예	아니요	예	예	일부(정보)

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon CloudWatch Application Insights	예	아니요	아니요	아니요	예	아니요
Amazon CloudWatch Application Signals	예	예	아니요	예	예	아니요
Amazon CloudWatch Evidently	예	예	아니요	예	예	아니요
Amazon CloudWatch Internet Monitor	예	예	아니요	예	예	아니요
Amazon CloudWatch Logs	예	예	예	부분적	예	예
Amazon CloudWatch Network Monitor	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon CloudWatch Observability Access Manager	예	예	아니요	예	예	아니요
Amazon CloudWatch RUM	예	예	아니요	예	예	아니요
Amazon CloudWatch Synthetics	예	예	아니요	예	예	아니요
AWS CodeArtifact	예	예	예	예	예	아니요
AWS CodeBuild	예	예	예 (정보)	일부(정보)	예	아니요
Amazon CodeCatalyst	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS CodeCommit	예	예	아니요	예	예	아니요
AWSCodeConnections	예	예	아니요	예	예	아니요
AWS CodeDeploy	예	예	아니요	예	예	아니요
AWS CodeDeploy 보안 호스트 명령 서비스	예	아니요	아니요	아니요	예	아니요
Amazon CodeGuru Profiler	예	예	아니요	예	예	예
Amazon CodeGuru Reviewer	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon CodeGuru Security	예	예	아니요	예	예	아니요
AWS CodePipeline	예	부분적	아니요	예	예	아니요
AWS CodeStar	예	부분적	아니요	예	예	아니요
AWS CodeStar Connections	예	예	아니요	예	예	예
AWS CodeStar 알림	예	예	아니요	예	예	예
Amazon CodeWhisperer	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Cognito	예	예	아니요	예	예	예
Amazon Cognito Sync	예	예	아니요	아니요	예	예
Amazon Cognito 사용자 풀	예	예	아니요	예	예	예
Amazon Comprehend	예	예	아니요	예	예	아니요
Amazon Comprehend Medical	예	아니요	아니요	아니요	예	아니요
AWS Compute Optimizer	예	아니요	아니요	아니요	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Config	예	일부(정보)	아니요	예	예	예
Amazon Connect	예	예	아니요	예	예	예
Amazon Connect Cases	예	예	아니요	예	예	아니요
Amazon Connect Customer Profiles	예	예	아니요	예	예	예
Amazon Connect 대용량 아웃바운드 통신	예	예	아니요	예	예	아니요
Amazon Connect Voice ID	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Console Mobile Application	예	예	아니요	아니요	예	아니요
AWS 통합 결제	예	아니요	아니요	아니요	예	아니요
AWS 제어 기능 카탈로그	예	예	아니요	아니요	예	아니요
AWS Control Tower	예	예	아니요	아니요	예	아니요
AWS Cost and Usage Report	예	예	아니요	아니요	예	아니요
AWS Cost Explorer	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Cost Optimization Hub	예	아니요	아니요	아니요	예	아니요
AWS 고객 검증 서비스	예	아니요	아니요	아니요	예	아니요
AWS Database Migration Service	예	예	아니요 (정보)	예	예	예
Database Query Metadata Service	예	아니요	아니요	아니요	예	아니요
AWS Data Exchange	예	예	아니요	예	예	아니요
Amazon Data Lifecycle Manager	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Data Pipeline	예	예	아니요	부분적	예	아니요
AWS DataSync	예	예	아니요	예	예	예
Amazon DataZone	예	아니요	아니요	아니요	예	아니요
AWSDeadline Cloud	예	예	아니요	예	예	아니요
AWS DeepComposer	예	예	아니요	예	예	아니요
AWS DeepRacer	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Detective	예	예	아니요	예	예	아니요
AWS Device Farm	예	예	아니요	예	예	예
Amazon DevOps Guru	예	예	아니요	아니요	예	예
AWS 진단 도구	예	예	아니요	예	예	아니요
AWS Direct Connect	예	예	아니요	예	예	예
AWS Directory Service	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon DocumentDB Elastic Clusters	예	예	아니요	예	예	예
Amazon DynamoDB Accelerator(DAX)	예	예	아니요	아니요	예	예
Amazon DynamoDB	예	예	예	아니요	예	아니요
Amazon Elastic Compute Cloud(Amazon EC2)	예	부분적	아니요	예	예	일부(정보)
Amazon EC2 Auto Scaling	예	예	아니요	예	예	예
EC2 Image Builder	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon EC2 Instance Connect	예	예	아니요	아니요	예	예
Amazon ElastiCache	예	예	아니요	예	예	예
AWS Elastic Beanstalk	예	부분적	아니요	예	예	예
Amazon Elastic Block Store(Amazon EBS)	예	부분적	아니요	예	예	아니요
Amazon Elastic Container Registry(Amazon ECR)	예	예	예	예	예	예
Amazon Elastic Container Registry 퍼블릭 (Amazon ECR 퍼블릭)	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Elastic Container Service(Amazon ECS)	 예	 일부(정보)	 아니요	 예	 예	 예
AWS Elastic Disaster Recovery	 예	 예	 아니요	 예	 예	 예
Amazon Elastic File System(Amazon EFS)	 예	 예	 예	 부분적	 예	 예
Amazon Elastic Inference	 예	 예	 아니요	 아니요	 예	 아니요
Amazon Elastic Kubernetes Service(Amazon EKS)	 예	 예	 아니요	 예	 예	 예
Amazon Elastic Kubernetes Service(Amazon EKS) Auth	 예	 예	 아니요	 아니요	 예	 아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Elastic Load Balancing	예	부분적	아니요	부분적	예	예
Amazon Elastic Transcoder	예	예	아니요	아니요	예	아니요
AWS Elemental Appliances and Software 활성화 서비스	예	예	아니요	예	예	아니요
AWS Elemental Appliances and Software	예	예	아니요	예	예	아니요
AWS Elemental MediaConnect	예	예	아니요	아니요	예	예
AWS Elemental MediaConvert	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Elemental MediaLive	예	예	아니요	예	예	아니요
AWS Elemental MediaPackage	예	예	아니요	예	예	일부(정보)
AWS Elemental MediaPackage V2	예	예	아니요	예	예	아니요
AWS Elemental MediaPackage VOD	예	예	아니요	예	예	일부(정보)
AWS Elemental MediaStore	예	예	예	예	예	아니요
AWS Elemental MediaTailor	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Elemental Support 사례	예	아니요	아니요	아니요	예	아니요
AWS Elemental Support 콘텐츠	예	아니요	아니요	아니요	예	아니요
Amazon EMR	예	예	아니요	예	예	예
Amazon EMR on EKS	예	예	아니요	예	예	예
Amazon EMR Serverless	예	예	아니요	예	예	예
AWS 엔터티 해상도	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon EventBridge	예	예	예	예	예	아니요
Amazon EventBridge 파이프	예	예	아니요	예	예	아니요
Amazon EventBridge 스케줄러	예	예	아니요	예	예	아니요
Amazon EventBridge 스키마	예	예	예	예	예	아니요
AWS Fault Injection Service	예	예	아니요	예	예	예
Amazon FinSpace	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon FinSpace API	예	예	아니요	아니요	예	아니요
AWS Firewall Manager	예	예	아니요	예	예	부분적
Fleet Hub for AWS IoT Device Management	예	예	아니요	예	예	아니요
Amazon Forecast	예	예	아니요	예	예	아니요
Amazon Fraud Detector	예	예	아니요	예	예	아니요
FreeRTOS	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS 프리 티어	예	아니요	아니요	아니요	예	아니요
Amazon FSx	예	예	아니요	예	예	예
Amazon GameLift	예	예	아니요	예	예	아니요
AWS Global Accelerator	예	예	아니요	예	예	예
AWS Glue	예	예	예	부분적	예	아니요
AWS Glue DataBrew	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Ground Station	예	예	아니요	예	예	예
Amazon Ground Truth Labeling	예	아니요	아니요	아니요	예	아니요
Amazon GuardDuty	예	예	아니요	예	예	예
AWS Health API 및 알림	예	예	아니요	아니요	예	아니요
AWS HealthImaging	예	예	아니요	예	예	아니요
AWS HealthLake	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS HealthOmics	예	예	아니요	예	예	아니요
AWS IAM Identity Center	예	예	아니요	부분적	예	예
IAM Identity Center 디렉터리	예	아니요	아니요	아니요	예	아니요
IAM Identity Center 아이덴티티 스토어	예	예	아니요	아니요	예	아니요
IAM Identity Center OIDC 서비스	예	예	아니요	아니요	예	아니요
AWS Identity and Access Management(IAM)	예	예	일부(정보)	일부(정보)	일부(정보)	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Identity and Access Management 액세스 분석기	예	예	아니요	예	예	부분적
AWS Identity and Access Management Roles Anywhere	예	예	아니요	예	예	예
AWS 아이덴티티 스토어 인증	예	아니요	아니요	아니요	예	아니요
AWS Identity Sync	예	예	아니요	아니요	예	아니요
AWS Import/Export	예	아니요	아니요	아니요	예	아니요
Amazon Inspector	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Inspector Classic	예	아니요	아니요	아니요	예	예
Amazon InspectorScan	예	아니요	아니요	아니요	예	아니요
Amazon Interactive Video Service	예	예	아니요	예	예	예
Amazon Interactive Video Service Chat	예	예	아니요	예	예	아니요
AWS 인보이스 발행	예	아니요	아니요	아니요	예	아니요
AWS IoT 1-Click	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS IoT Analytics	예	예	아니요	예	예	아니요
AWS IoT	예	예	일부(정보)	예	예	아니요
AWS IoT Core Device Advisor	예	예	아니요	예	예	아니요
AWS IoT Device Tester	예	아니요	아니요	아니요	예	아니요
AWS IoT Events	예	예	아니요	예	예	아니요
AWS IoT FleetWise	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS IoT Greengrass	예	예	아니요	예	예	아니요
AWS IoT Greengrass V2	예	예	아니요	부분적	예	아니요
AWS IoT 작업 데이터 영역	예	예	아니요	아니요	예	아니요
AWS IoT RoboRunner	예	예	아니요	아니요	예	아니요
AWS IoT SiteWise	예	예	아니요	예	예	예
AWS IoT TwinMaker	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS IoT Wireless	예	예	아니요	예	예	아니요
AWS IQ	예	예	아니요	아니요	예	예
AWS IQ 권한	예	예	아니요	아니요	예	아니요
Amazon Kendra	예	예	아니요	예	예	아니요
Amazon Kendra Intelligent Ranking	예	예	아니요	예	예	아니요
AWS Key Management Service (AWS KMS)	예	예	예	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Keyspaces (Apache Cassandra용)	예	예	아니요	예	예	예
Amazon Managed Service for Apache Flink	예	예	아니요	예	예	아니요
Amazon Managed Service for Apache Flink V2	예	예	아니요	예	예	아니요
Amazon Data Firehose	예	예	아니요	예	예	아니요
Amazon Kinesis Data Streams	예	예	예	아니요	예	아니요
Amazon Kinesis Video Streams	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Lake Formation	예	아니요	아니요	아니요	예	예
AWS Lambda	예	예	예	일부 (정보)	예	일부(정보)
AWS Launch Wizard	예	아니요	아니요	아니요	예	아니요
Amazon Lex	예	예	아니요	예	예	예
Amazon Lex V2	예	예	예	예	예	예
AWS License Manager	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS License Manager Linux Subscriptions Manager	예	아니요	아니요	아니요	예	아니요
AWS License Manager User Subscriptions	예	아니요	아니요	아니요	예	예
Amazon Lightsail	예	일부(정보)	아니요	일부(정보)	예	예
Amazon Location Service	예	예	아니요	예	예	아니요
Amazon Lookout for Equipment	예	예	아니요	예	예	아니요
Amazon Lookout for Metrics	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Lookout for Vision	예	예	아니요	예	예	아니요
Amazon Machine Learning	예	예	아니요	아니요	예	아니요
Amazon Macie	예	예	아니요	예	예	예
AWS Mainframe Modernization	예	예	아니요	예	예	예
AWS Mainframe Modernization 애플리케이션 테스트	예	예	아니요	예	예	아니요
Amazon Managed Blockchain	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Managed Blockchain Query	예	아니요	아니요	아니요	예	아니요
Amazon Managed Grafana	예	예	아니요	예	예	예
Amazon Managed Service for Prometheus	예	예	아니요	예	예	아니요
Amazon Managed Streaming for Apache Kafka(MSK)	예	예	일부(정보)	예	예	예
Amazon Managed Streaming for Kafka Connect	예	예	아니요	아니요	예	예
Amazon Managed Workflows for Apache Airflow	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Marketplace	예	아니요	아니요	아니요	예	예
AWS Marketplace Catalog	예	예	아니요	예	예	아니요
AWS Marketplace Commerce Analytics	예	아니요	아니요	아니요	아니요	아니요
AWS Marketplace 서비스 배포	예	예	아니요	예	예	아니요
AWS Marketplace 검색	예	아니요	아니요	아니요	예	아니요
AWS Marketplace Entitlement Service	예	아니요	아니요	아니요	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Marketplace Image Building Service	예	아니요	아니요	아니요	예	아니요
AWS Marketplace Management Portal	예	아니요	아니요	아니요	예	아니요
AWS Marketplace Metering Service	예	아니요	아니요	아니요	예	아니요
AWS Marketplace Private Marketplace	예	아니요	아니요	아니요	예	아니요
AWS Marketplace Procurement Systems Integration	예	아니요	아니요	아니요	예	아니요
AWS Marketplace Seller Reporting	예	예	아니요	아니요	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Marketplace Vendor Insights	예	예	아니요	예	예	아니요
Amazon Mechanical Turk	예	아니요	아니요	아니요	예	아니요
Amazon MediaImport	예	아니요	아니요	아니요	아니요	아니요
Amazon MemoryDB	예	예	아니요	예	예	예
Amazon 메시지 전송 서비스	예	아니요	아니요	아니요	예	아니요
Amazon Message Gateway Service	예	아니요	아니요	아니요	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Microservice Extractor for .NET	예	아니요	아니요	아니요	예	아니요
AWS 마이그레이션 가속화 프로그램 크레딧	예	예	아니요	아니요	예	아니요
AWS Migration Hub	예	예	아니요	아니요	예	예
AWS Migration Hub 오케스트레이터	예	예	아니요	예	예	예
AWS Migration Hub Refactor Spaces	예	예	예	예	예	예
AWS Migration Hub 전략 권장 사항	예	아니요	아니요	아니요	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Monitron	예	예	아니요	예	예	예
Amazon MQ	예	예	아니요	예	예	예
Amazon Neptune	예	예	아니요	아니요	예	예
Amazon Neptune Analytics	예	예	아니요	예	예	아니요
AWS Network Firewall	예	예	아니요	예	예	예
AWS Network Manager	예	예	아니요	예	예	예(정보)

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Network Manager 채팅	예	아니요	아니요	아니요	예	아니요
Amazon Nimble Studio	예	예	아니요	예	예	아니요
Amazon One Enterprise	예	예	아니요	예	예	아니요
Amazon OpenSearch Ingestion	예	예	아니요	예	예	예
Amazon OpenSearch Serverless	예	예	아니요	예	예	예
Amazon OpenSearch Service	예	예	예	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS OpsWorks	예	예	아니요	아니요	예	아니요
AWS OpsWorks 구성 관리	예	예	아니요	아니요	예	아니요
AWS Organizations	예	예	아니요	예	아니요	예
AWS Outposts	예	예	아니요	예	예	예
AWS Panorama	예	예	아니요	예	예	예
AWS Partner 중앙 계정 관리	예	아니요	아니요	아니요	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Payment Cryptography	예	예	아니요	예	예	아니요
AWS 결제	예	아니요	아니요	아니요	예	아니요
AWS 성능 개선 도우미	예	예	아니요	아니요	예	아니요
Amazon Personalize	예	예	아니요	아니요	예	아니요
Amazon Pinpoint	예	예	아니요	예	예	아니요
Amazon Pinpoint 이메일 서비스	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Pinpoint SMS 및 음성 서비스	예	아니요	아니요	아니요	예	아니요
Amazon Pinpoint SMS and Voice Service V2	예	예	아니요	예	예	아니요
Amazon Polly	예	예	아니요	아니요	예	아니요
AWS 가격표	예	아니요	아니요	아니요	예	아니요
AWS Private 5G	예	예	아니요	예	예	아니요
AWS Private CA액티브 디렉터리용 커넥터	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Private CA SCEP용 커넥터	예	예	아니요	예	예	아니요
AWS Private Certificate Authority (AWS Private CA)	예	예	예	예	예	아니요
AWS Proton	예	예	아니요	예	예	예
AWS Purchase Orders 콘솔	예	예	아니요	예	예	아니요
Amazon Q 비즈니스용	예	예	아니요	예	예	예
Amazon Q Business Q 앱	예	예	아니요	아니요	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Q 개발자	예	아니요	아니요	아니요	예	예
Amazon Q in Connect	예	예	아니요	예	예	아니요
Amazon Quantum Ledger Database(Amazon QLDB)	예	예	아니요	예	예	아니요
Amazon QuickSight	예	예	아니요	예	예	아니요
Amazon RDS Data API	예	예	아니요	아니요	예	아니요
Amazon RDS IAM 인증	예	예	아니요	아니요	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS 휴지통	예	예	아니요	예	예	아니요
Amazon Redshift	예	예	아니요	예	예	예
Amazon Redshift 데이터 API	예	예	아니요	아니요	예	아니요
Amazon Redshift Serverless	예	예	예	예	예	아니요
Amazon Rekognition	예	예	일부(정보)	예	예	아니요
Amazon Relational Database Service(Amazon RDS) (정보)	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS re:Post 프라이빗	예	예	아니요	예	예	예
AWS Resilience Hub	예	예	아니요	예	예	아니요
AWS Resource Access Manager (AWS RAM)	예	예	아니요	예	예	예
AWS 리소스 탐색기	예	예	아니요	예	예	예
AWS Resource Groups	예	예	아니요	예	일부(정보)	예
AWS Resource Groups Tagging API	예	아니요	아니요	아니요	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon RHEL Knowledgebase 포털	예	아니요	아니요	아니요	예	아니요
AWS RoboMaker	예	예	아니요	예	예	예
Amazon Route 53	예	예	아니요	아니요	예	아니요
Amazon Route 53 애플리케이션 복구 컨트롤러 - 영역 이동	예	예	아니요	아니요	예	아니요
Amazon Route 53 도메인	예	아니요	아니요	아니요	아니요	아니요
Amazon Route 53 프로필	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Route 53 Recovery Cluster	예	예	아니요	아니요	예	아니요
Amazon Route 53 Recovery Control Config	예	예	아니요	예	예	아니요
Amazon Route 53 Recovery Readiness	예	예	아니요	예	예	예
Amazon Route 53 Resolver	예	예	아니요	예	예	예
Amazon S3 Express	예	예	아니요	아니요	예	아니요
Amazon S3 Glacier	예	예	예	예	예	부분적

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon SageMaker	예	예	아니요	예	예	일부(정보)
Amazon SageMaker 지리 공간 기능	예	예	아니요	예	예	아니요
Amazon SageMaker Ground Truth Synthetic	예	아니요	아니요	아니요	예	아니요
MLflow가 탑재된 Amazon SageMaker	예	예	아니요	아니요	예	아니요
AWS 절감형 플랜	예	예	아니요	예	예	아니요
AWS Secrets Manager	예	예	예	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Security Hub	예	예	아니요	예	예	예
Amazon Security Lake	예	예	아니요	아니요	예	예
AWS Security Token Service (AWS STS)	예	일부(정보)	아니요	예	일부(정보)	아니요
AWS Serverless Application Repository	예	예	예	아니요	예	아니요
AWS Service Catalog	예	예	아니요	예	예	예
Service Quotas	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Shield	예	예	아니요	예	예	예
AWS Signer	예	예	예	예	예	아니요
AWS 로그인	예	아니요	아니요	아니요	예	아니요
Amazon SimpleDB	예	예	아니요	아니요	예	아니요
Amazon Simple Email Service - Mail Manager	예	예	아니요	예	예	예
Amazon Simple Email Service(Amazon SES) v2	예	일부(정보)	예	예	일부(정보)	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Simple Notification Service (Amazon SNS)	예	예	예	예	예	아니요
Amazon Simple Queue Service(Amazon SQS)	예	예	예	부분적	예	아니요
Amazon Simple Storage Service(S3)	예	예	예	일부(정보)	예	일부(정보)
Amazon Simple Storage Service(Amazon S3) 객체 Lambda	예	예	아니요	아니요	예	아니요
AWS Outposts의 Amazon Simple Storage Service(Amazon S3)	예	예	예	아니요	예	네
Amazon Simple Workflow Service (Amazon SWF)	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS SimSpace Weaver	예	예	아니요	예	예	아니요
AWS Site-to-Site VPN	예	예	아니요	아니요	예	예
AWS Snowball	예	아니요	아니요	아니요	예	아니요
AWS Snowball 엣지	예	아니요	아니요	아니요	예	아니요
AWS Snow Device Management	예	예	아니요	예	예	아니요
AWS SQL Workbench	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Step Functions	예	예	아니요	예	예	아니요
AWS Storage Gateway	예	예	아니요	예	예	아니요
AWS Supply Chain	예	예	아니요	예	예	아니요
AWS Support App in Slack	예	아니요	아니요	아니요	예	아니요
AWS Support	예	아니요	아니요	아니요	예	예
AWS Support 플랜	예	아니요	아니요	아니요	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Support 권장 사항	예	아니요	아니요	아니요	예	아니요
AWS 지속 가능성	예	아니요	아니요	아니요	예	아니요
AWS Systems Manager	예	예	아니요	예	예	예
AWS Systems Manager for SAP	예	예	아니요	예	예	아니요
AWS Systems ManagerGUI Connect	예	아니요	아니요	아니요	예	아니요
AWS Systems Manager Incident Manager	예	예	예	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Systems Manager Incident Manager 연락처	예	예	예	아니요	예	아니요
AWS Systems Manager 빠른 설정	예	예	아니요	예	예	아니요
태그 편집기	예	아니요	아니요	아니요	예	아니요
AWS 세금 설정	예	아니요	아니요	아니요	예	아니요
AWS 통신 네트워크 빌더	예	예	아니요	예	예	아니요
Amazon Textract	예	아니요	아니요	아니요	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Timestream	예	예	아니요	예	예	아니요
Amazon Timestream Influxdb	예	예	아니요	예	예	예
AWS Tiros API(Reachability Analyzer용)	예	아니요	아니요	아니요	아니요	아니요
Amazon Transcribe	예	예	아니요	예	예	아니요
AWS Transfer Family	예	예	아니요	예	예	아니요
Amazon Translate	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Trusted Advisor	 일부(정보)	 예	 아니요	 아니요	 부분적	 예
AWS 사용자 알림	 예	 예	 아니요	 예	 예	 예
AWS 사용자 알림 연락처	 예	 예	 아니요	 예	 예	 아니요
AWS User Subscriptions	 예	 아니요	 아니요	 아니요	 예	 아니요
AWS Verified Access	 예	 아니요	 아니요	 아니요	 예	 아니요
Amazon Verified Permissions	 예	 예	 아니요	 아니요	 예	 아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon Virtual Private Cloud(VPC)	예	일부(정보)	일부(정보)	예	예	일부(정보)
Amazon VPC Lattice	예	예	아니요	예	예	아니요
Amazon VPC Lattice Services	예	예	아니요	아니요	예	아니요
AWS WAF	예	예	아니요	예	예	예
AWS WAF 클래식	예	예	아니요	예	예	예
AWS WAF 리전	예	예	아니요	예	예	예

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
AWS Well-Architected Tool	예	예	아니요	예	예	아니요
AWS Wickr	예	예	아니요	예	예	아니요
Amazon WorkDocs	예	아니요	아니요	아니요	예	아니요
Amazon WorkMail	예	예	아니요	예	예	예
Amazon WorkMail 메시지 흐름	예	예	아니요	아니요	예	아니요
Amazon WorkSpaces	예	예	아니요	예	예	아니요

서비스	작업	리소스 수준 권한	리소스 기반 정책	ABAC	임시 자격 증명	서비스 연결 역할
Amazon WorkSpaces Secure Browser	예	예	아니요	예	예	예
Amazon WorkSpaces Thin Client	예	예	아니요	예	예	아니요
AWS X-Ray	예	일부(정보)	아니요	일부(정보)	예	아니요

추가 정보

Amazon CloudFront

CloudFront에는 서비스 연결 역할이 없지만 Lambda@Edge에는 있습니다. 자세한 내용은 Amazon CloudFront 개발자 안내서의 Lambda@Edge에 서비스 연결 역할 사용을 참조하세요.

AWS CloudTrail

CloudTrail은 AWS 외부 이벤트 소스와의 CloudTrail Lake 통합에 사용되는 CloudTrail 채널에서만 리소스 기반 정책을 지원합니다.

Amazon CloudWatch

CloudWatch 서비스 연결 역할은 AWS Management Console을 사용하여 생성할 수 없으며 경보 작업 기능만 지원합니다.

AWS CodeBuild

CodeBuild는 AWS RAM을 사용한 크로스 계정 리소스 공유를 지원합니다.

CodeBuild는 프로젝트 기반 작업에 대한 ABAC를 지원합니다.

AWS Config

AWS Config의 경우 다중 계정 다중 리전 데이터 집계 및 AWS Config 규칙에 대한 리소스 수준 권한을 지원합니다. 지원되는 리소스 목록을 보려면 AWS Config API 설명서의 Multi-Account Multi-Region Data Aggregation 섹션과 AWS Config Rules 섹션을 참조하세요.

AWS Database Migration Service

지원되는 대상 엔드포인트로 마이그레이션된 데이터를 암호화하도록 생성한 AWS KMS 암호화 키에 연결된 정책을 생성 및 수정할 수 있습니다. 지원되는 대상 엔드포인트에는 Amazon Redshift 및 Amazon S3 등이 있습니다. 자세한 내용은 AWS Database Migration Service사용 설명서의 Amazon Redshift 대상 데이터를 암호화하는 AWS KMS 키 생성 및 사용, Amazon S3 대상 객체를 암호화하는 AWS KMS 키 생성을 참조하세요.

Amazon Elastic Compute Cloud

Amazon EC2 서비스 연결 역할은 스팟 인스턴스 요청, 스팟 플릿 요청, Amazon EC2 플릿 및 Windows 인스턴스의 빠른 시작 기능에만 사용할 수 있습니다.

Amazon Elastic Container Service

일부 Amazon ECS 작업만 리소스 수준 권한을 지원합니다.

AWS Elemental MediaPackage

MediaPackage는 CloudWatch에 고객 액세스 로그 게시를 위한 서비스 연결 역할을 지원하지만 다른 API 작업을 위한 서비스 연결 역할은 지원하지 않습니다.

AWS Identity and Access Management

IAM은 역할 신뢰 정책이라고 하는 리소스 기반 정책 유형 하나만 지원하며, 이 유형은 IAM 역할에 연결됩니다. 자세한 내용은 사용자에게 역할을 전환할 권한 부여 단원을 참조하세요.

IAM에서는 대부분의 IAM 리소스에 대해서만 태그 기반 액세스 제어를 지원합니다. 자세한 내용은 AWS Identity and Access Management 리소스용 태그 단원을 참조하세요.

IAM 에 대한 일부 API 작업만 임시 보안 인증 정보로 직접적으로 호출할 수 있습니다. 자세한 내용은 API 옵션 비교 섹션을 참조하세요.

AWS IoT

AWS IoT에 연결된 디바이스는 X.509 인증서 또는 Amazon Cognito 자격 증명을 통해 인증됩니다. AWS IoT 정책을 X.509 인증서 또는 Amazon Cognito 자격 증명에 연결하여 디바이스가 어떤 작업을 수행하도록 권한 부여할 것인지 제어할 수 있습니다. 자세한 내용은 AWS IoT 개발자 안내서의 AWS IoT의 보안 및 자격 증명을 참조하세요.

AWS Lambda

Lambda는 Lambda 함수를 필수 리소스로 사용하는 API 작업에 대한 ABAC(속성 기반 액세스 제어)를 지원합니다. 레이어, 이벤트 소스 매핑 및 코드 서명 구성 리소스는 지원되지 않습니다.

Lambda에는 서비스 연결 역할이 없지만 Lambda@Edge에는 있습니다. 자세한 내용은 Amazon CloudFront 개발자 안내서의 Lambda@Edge의 서비스 연결 역할을 참조하세요.

Amazon Lightsail

Lightsail은 리소스 수준 권한 및 ABAC를 부분적으로 지원합니다. 자세한 내용은 Amazon Lightsail를 위한 작업, 리소스 및 조건 키를 참조하세요.

Amazon Managed Streaming for Apache Kafka(MSK)

다중 VPC 연결을 위해 구성된 Amazon MSK 클러스터에 클러스터 정책을 연결할 수 있습니다.

AWS Network Manager

AWS Cloud WAN에서는 서비스 연결 역할도 지원합니다. 자세한 내용은 Amazon VPC AWS 클라우드 WAN 가이드의 AWS 클라우드 WAN 서비스 연결 역할을 참조하세요.

Amazon Relational Database Service

Amazon Aurora는 MySQL 및 PostgreSQL과 호환되는 완전 관리형 관계형 데이터베이스 엔진입니다. Amazon RDS를 통해 새 데이터베이스 서버를 설정할 때 Aurora MySQL 또는 Aurora PostgreSQL을 DB 엔진 옵션으로 선택할 수 있습니다. 자세한 내용은 Amazon Aurora 사용 설명서의 Amazon Aurora ID 및 액세스 관리를 참조하세요.

Amazon Rekognition

리소스 기반 정책은 Amazon Rekognition Custom Labels 모델을 복사하는 경우에만 지원됩니다.

AWS Resource Groups

사용자는 Resource Groups 작업을 허용하는 정책이 있는 역할을 수임할 수 있습니다.

Amazon SageMaker

서비스 연결 역할은 현재 SageMaker Studio 및 SageMaker 교육 작업에 사용할 수 있습니다.

AWS Security Token Service

AWS STS는 '리소스'가 없지만 유사한 방식으로 사용자에 대한 액세스를 제한할 수 있습니다. 자세한 정보는 이름을 사용한 임시 보안 자격 증명 액세스 거부 섹션을 참조하세요.

AWS STS에 대한 일부 API 작업만 임시 보안 인증 정보를 통한 직접 호출을 지원합니다. 자세한 내용은 API 옵션 비교 섹션을 참조하세요.

Amazon Simple Email Service

`ses:SendEmail` 또는 `ses:SendRawEmail` 등과 같이 이메일 전송과 관련된 작업을 참조하는 정책 설명의 리소스 수준 권한만 사용할 수 있습니다. 기타 다른 작업을 참조하는 정책 설명의 경우 리소스 요소에 *만 포함될 수 있습니다.

Amazon SES API만 임시 보안 인증 정보를 지원합니다. Amazon SES SMTP 인터페이스는 임시 보안 자격 증명에서 파생된 SMTP 자격 증명을 지원하지 않습니다.

Amazon Simple Storage Service(S3)

Amazon S3에서는 객체 리소스에 대해서만 태그 기반 승인을 지원합니다.

Amazon S3는 Amazon S3 Storage Lens 대한 서비스 연결 역할을 지원합니다.

AWS Trusted Advisor

Trusted Advisor에 대한 API 액세스는 AWS Support API를 통해 이루어지며 AWS Support IAM 정책으로 제어합니다.

Amazon Virtual Private Cloud

IAM 사용자 정책에서는 특정 Amazon VPC 엔드포인트에 대해 권한을 제한할 수 없습니다. `Action` 또는 `ec2:*VpcEndpoint*` API 작업을 포함하는 모든 `ec2:DescribePrefixLists` 요소는 `""Resource": "*""`를 포함해야 합니다. 자세한 내용은 AWS PrivateLink 가이드의 VPC 엔드포인트 및 VPC 엔드포인트 서비스에 대한 ID 및 액세스 관리 섹션을 참조하세요.

Amazon VPC에서는 단일 리소스 정책을 VPC 엔드포인트에 연결하여 해당 엔드포인트를 통해 액세스 가능한 대상을 제한할 수 있도록 지원합니다. 특정 Amazon VPC 엔드포인트의 리소스에 대한 액세스를 제어하기 위해 리소스 기반 정책을 사용하는 방법에 대한 자세한 정보는 AWS PrivateLink 가이드의 엔드포인트 정책을 사용하여 서비스에 대한 액세스 제어 섹션을 참조하세요.

Amazon VPC에는 서비스 연결 역할이 없지만 AWS Transit Gateway에는 있습니다. 자세한 내용은 Amazon VPC AWS Transit Gateway 사용 설명서의 전송 게이트웨이에 대한 서비스 연결 역할 사용을 참조하세요.

AWS X-Ray

X-Ray는 모든 작업에 대해 리소스 수준 권한을 지원하지는 않습니다.

X-Ray는 그룹 및 샘플링 규칙에 대해 태그 기반 액세스 제어를 지원합니다.

API 요청용 AWS Signature Version 4

> **Important**
>
> AWS SDK(샘플 코드 및 라이브러리 참조) 또는 AWS Command Line Interface(AWS CLI) 도구를 사용하여 API 요청을 AWS로 전송하는 경우 SDK 및 CLI 클라이언트는 사용자 요청 인증에 사용자가 제공한 액세스 키를 사용하므로 서명 프로세스를 건너뛸 수 있습니다. 그럴 만한 이유가 없는 경우 항상 SDK 또는 CLI를 사용하는 것이 좋습니다.
> 여러 서명 버전을 지원하는 리전에서는 수동 서명을 요청하는 경우 사용할 서명 버전을 지정해야 합니다. 다중 리전 액세스 포인트로 요청을 공급하는 경우 SDK와 CLI가 자동으로 추가 구성 없이 서명 버전 4A를 사용하는 것으로 전환됩니다.

요청에서 보내는 인증 정보에는 서명이 포함되어야 합니다. AWS Signature Version 4(SigV4)는 AWS API 요청에 인증 정보를 추가하기 위한 AWS 서명 프로토콜입니다.

비밀 액세스 키를 사용하여 API 요청에 서명하지 않습니다. 대신 SigV4 서명 프로세스를 사용합니다. 서명 요청:

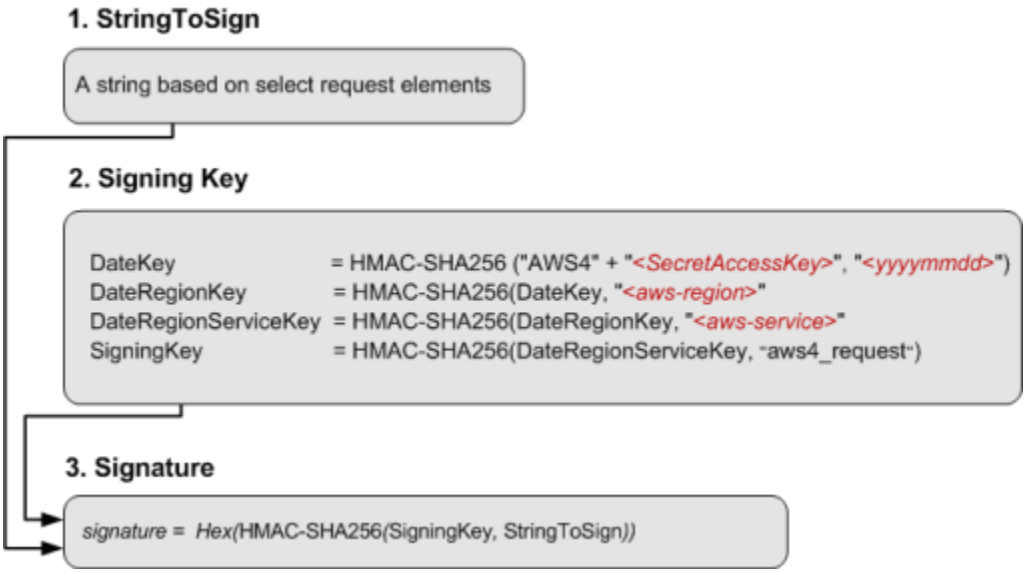
1. 요청 세부 정보를 기반으로 표준 요청 생성.
2. AWS 자격 증명을 사용하여 서명 계산.
3. 이 서명을 요청에 Authorization 헤더로 추가.

AWS에서는 이 프로세스를 복제하고 서명을 확인하여 그에 따라 액세스를 허용하거나 거부합니다.

> **Note**
>
> AWS는 다중 리전 API 요청에 대한 서명을 지원하는 확장 기능인 Signature Version 4A도 지원합니다. 자세한 정보는 GitHub에서 sigv4a-signing-examples 프로젝트를 참조하세요.

AWS SigV4 작동 방식

다음 다이어그램은 서명을 계산하는 일반적인 프로세스를 보여줍니다.



1. 서명할 문자열은 요청 유형에 따라 다릅니다. 예를 들어 HTTP Authorization 헤더 또는 쿼리 파라미터를 인증에 사용하는 경우 요청 요소 조합을 사용하여 서명할 문자열을 생성합니다. HTTP POST 요청의 경우 요청의 POST 정책은 서명하는 문자열입니다.
2. 서명 키는 각 단계의 결과를 다음 단계에 제공하는 일련의 계산입니다. 마지막 단계는 서명 키입니다.
3. AWS 서비스에서 인증된 요청을 수신하면 요청에 포함된 인증 정보를 사용하여 서명을 다시 생성합니다. 서명이 일치하는 경우 서비스에서 요청을 처리합니다. 그렇지 않으면 서비스에서 요청을 거부합니다.

자세한 내용은 AWS API 요청 서명의 요소 단원을 참조하십시오.

요청에 서명하는 경우

AWS에 API 요청을 전송하는 사용자 지정 코드를 작성할 때 요청에 서명하는 코드를 포함해야 합니다. 사용자 지정 코드를 작성하는 이유는 다음과 같습니다.

- AWS SDK가 없는 프로그래밍 언어로 작업하는 경우
- AWS에 요청을 전송하는 방법을 완벽하게 제어해야 하는 경우

API 요청은 AWS SigV4로 액세스를 인증하는 반면, AWS SDK와 AWS CLI는 사용자가 제공하는 액세스 키를 사용하여 요청을 인증합니다. AWS SDK 및 AWS CLI를 사용한 인증에 대한 자세한 내용은 추가 리소스의 내용을 참조하세요.

요청에 서명하는 이유

서명 프로세스는 다음과 같은 방법으로 요청을 보호할 수 있습니다.

- 요청자의 ID 확인

  인증된 요청에는 액세스 키(액세스 키 ID, 비밀 액세스 키)를 사용하여 만든 서명이 필요합니다. 임시 보안 인증 정보를 사용하는 경우 서명 계산에 보안 토큰도 필요합니다. 자세한 내용은 AWS 보안 자격 증명 프로그래밍 방식 액세스를 참조하세요.
- 전송 중인 데이터 보호

  요청이 전송되는 동안 훼손되는 것을 방지하기 위해 일부 요청 요소를 사용하여 요청의 해시(다이제스트)를 계산하고 결과 해시 값을 요청의 일부로 포함합니다. AWS 서비스에서는 요청이 수신되면 동일한 정보를 사용하여 해시를 계산한 후 요청에 있는 해시 값과 비교합니다. 값이 일치하지 않으면 AWS에서 요청을 거부합니다.
- 잠재적 재생 공격으로부터 보호

  대부분의 경우 요청서의 타임스탬프 시간으로부터 5분 이내에 AWS에 요청이 도착해야 합니다. 그렇지 않으면 AWS가 요청을 거부합니다.

AWS SigV4는 HTTP Authorization 헤더에서 또는 URL의 쿼리 문자열로 표현될 수 있습니다. 자세한 내용은 인증 방법 단원을 참조하십시오.

추가 리소스

- 다양한 서비스의 SigV4 서명 프로세스에 대한 자세한 내용은 서명 요청 예의 내용을 참조하세요.
- AWS CLI의 프로그래밍 액세스를 위한 자격 증명을 구성하려면 AWS 명령줄 인터페이스 사용 설명서의 인증 및 액세스 자격 증명을 참조하세요.
- AWS SDK에는 AWS API 요청에 서명하기 위한 GitHub의 소스 코드가 포함되어 있습니다. 코드 샘플은 AWS 샘플 리포지토리의 예시 프로젝트 단원을 참조하십시오.

- AWS SDK for .NET – AWS4Signer.cs
- AWS SDK for C++ – AWSAuthV4Signer.cpp
- AWS SDK for Go – v4.go
- AWS SDK for Java – BaseAws4Signer.java
- AWS SDK for JavaScript – v4.js
- AWS SDK for PHP – SignatureV4.php
- AWS SDK for Python (Boto) – signers.py
- AWS SDK for Ruby – signer.rb

AWS API 요청 서명의 요소

> **Important**
>
> AWS SDK 또는 CLI를 사용하지 않는 한 요청에 인증 정보를 제공하는 서명을 계산하는 코드를 작성해야 합니다. AWS 서명 버전 4의 서명 계산은 복잡한 작업일 수 있으므로 가능한 경우 AWS SDK 또는 CLI를 사용하는 것이 좋습니다.

서명 버전 4를 사용하는 각 HTTP/HTTPS 요청은 이러한 요소를 반드시 포함해야 합니다.

요소

- 엔드포인트 사양
- 작업
- 작업 파라미터
- 날짜
- 인증 정보

엔드포인트 사양

요청을 전송할 엔드포인트의 DNS 이름을 지정합니다. 이 이름에는 일반적으로 서비스 코드와 리전이 포함됩니다. 예를 들어 `us-east-1` 리전에서 Amazon DynamoDB의 엔드포인트는 `dynamodb.us-east-1.amazonaws.com`입니다.

HTTP/1.1 요청의 경우에는 Host 헤더를 반드시 포함해야 합니다. HTTP/2 요청의 경우에는 :authority 헤더나 Host 헤더를 포함할 수 있습니다. HTTP/2 사양의 규정 준수용으로는 :authority 헤더만 사용하십시오. 모든 서비스가 HTTP/2 요청을 지원하는 것은 아닙니다.

각 서비스에서 지원하는 엔드포인트에 대한 자세한 내용은 AWS 일반 참조의 서비스 엔드포인트 및 할당량을 참조하세요.

작업

서비스에 대한 API 작업을 지정합니다. DynamoDB CreateTable 작업 또는 Amazon EC2 DescribeInstances 작업을 예로 들 수 있습니다.

각 서비스에서 지원하는 작업은 서비스 승인 참조를 참조하세요.

작업 파라미터

요청에 지정된 작업의 파라미터를 지정합니다. 각 AWS API 작업에는 필수 및 선택적 파라미터 세트가 있습니다. API 버전은 대부분 필수 파라미터입니다.

API 작업에서 지원하는 파라미터는 해당 서비스의 API 참조를 참조하세요.

날짜

요청의 날짜 및 시간을 지정합니다. 요청에 날짜 및 시간을 포함하면 제3자가 요청을 가로채고 나중에 다시 전송하지 못하도록 차단할 수 있습니다. 보안 인증 범위에 지정하는 날짜는 요청의 날짜와 일치해야 합니다.

타임스탬프는 협정 세계시(UTC)이고 YYYYMMDDTHHMMSSZ ISO 8601 형식이어야 합니다. 예: 20220830T123600Z. 타임스탬프에 밀리초를 포함하지 마십시오.

date 또는 x-amz-date 헤더를 사용하거나 x-amz-date를 쿼리 파라미터로 포함할 수 있습니다. x-amz-date 헤더를 찾을 수 없는 경우 date 헤더를 찾습니다.

인증 정보

사용자가 전송하는 각 요청에는 다음 정보가 포함되어야 합니다. AWS는 이 정보를 사용하여 요청의 유효성 및 진위를 확인합니다.

- 알고리즘 - AWS4-HMAC-SHA256을 사용하여 HMAC-SHA256 해시 알고리즘으로 서명 버전 4를 지정합니다.

- 보안 인증 정보 - 액세스 키 ID, YYYYMMDD 형식의 날짜, 리전 코드, 서비스 코드 및 `aws4_request` 종료 문자열로 구성된 문자열로, 슬래시(/)로 구분됩니다. 리전 코드, 서비스 코드 및 종료 문자열에는 소문자를 사용해야 합니다.

  ```
  AKIAIOSFODNN7EXAMPLE/YYYYMMDD/region/service/aws4_request
  ```

- 서명된 헤더 - 서명에 포함할 HTTP 헤더로, 세미콜론(;)으로 구분됩니다. 예: `host;x-amz-date`.
- 서명 - 계산된 서명을 나타내는 16진수로 인코딩된 문자열입니다. `Algorithm` 파라미터에서 지정한 알고리즘을 사용해 서명을 계산해야 합니다.

자세한 내용은 인증 방법 섹션을 참조하세요.

인증 방법

> **Important**
>
> AWS SDK 또는 CLI를 사용하지 않는 한 요청에 인증 정보를 제공하는 서명을 계산하는 코드를 작성해야 합니다. AWS 서명 버전 4의 서명 계산은 복잡한 작업일 수 있으므로 가능한 경우 AWS SDK 또는 CLI를 사용하는 것이 좋습니다.

다음 방법 중 하나를 사용하여 인증 정보를 표현할 수 있습니다.

HTTP Authorization 헤더

HTTP `Authorization` 헤더는 요청을 인증하는 가장 일반적인 방법입니다. 모든 REST API 작업(`POST`를 사용한 브라우저 기반 업로드 제외)에는 이 헤더가 필요합니다. Authorization 헤더 값, 서명 방법과 관련 옵션에 대한 자세한 내용은 Amazon S3 API 참조의 Authenticating Requests: Using the Authorization Header(AWS Signature Version 4)를 참조하세요.

다음은 `Authorization` 헤더 값의 예시입니다. 가독성을 위해 이 예시에 줄바꿈이 추가됩니다. 코드에서는 헤더가 연속된 문자열이어야 합니다. 알고리즘과 보안 인증 정보 사이에는 쉼표가 없지만 다른 요소는 쉼표로 구분해야 합니다.

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

다음 표에서는 앞선 예제에 있는 Authorization 헤더 값의 다양한 구성 요소를 설명합니다.

구성 요소	설명
권한 부여	서명을 계산하는 데 사용된 알고리즘입니다. 인증을 위해 AWS 서명 버전 4를 사용할 때 이 값을 제공해야 합니다. 문자열은 AWS 서명 버전 4(AWS4)와 서명 알고리즘(HMAC-SHA256)을 지정합니다.
Credential	액세스 키 ID 및 범위 정보(날짜, 리전 및 서명 계산에 사용된 서비스 포함). 이 문자열의 형식은 다음과 같습니다. `<your-access-key-id>/<date>/ <aws-region>/<aws-service>/ aws4_request` 여기에서 `<date>` 값은 YYYYMMDD 형식을 사용하여 지정됩니다. `<aws-service>` 값은 Amazon S3로 요청을 보낼 때 s3입니다.
SignedHeaders	`Signature` 계산에 사용한 세미콜론으로 구분된 요청 헤더의 목록입니다. 목록에는 헤더 이름만 포함되며, 헤더 이름은 소문자여야 합니다. 예: host;range;x-amz-date
Signature	64개의 소문자 16진수 문자로 표현되는 256 비트 서명. 예:`fe5f80f77d5fa3beca 038a248ff027d0445342fe2855d dc963176630326f1024` 서명 계산은 페이로드를 전송하기 위해 선택한 옵션에 따라 달라집니다.

쿼리 문자열 파라미터

쿼리 문자열을 사용하여 전체 요청을 URL로 표현할 수 있습니다. 이 경우 쿼리 파라미터를 사용하여 인증 정보를 포함한 요청 정보를 제공합니다. 요청 서명이 URL의 일부이므로 이 URL 유형을 미리 서명된 URL이라고 합니다. 미리 서명된 URL을 사용하여 최대 7일 동안 유효할 수 있는 클릭 가능한 링크를 HTML에 포함할 수 있습니다. 자세한 내용은 Amazon S3 API 참조의 Authenticating Requests: Using Query Parameters(AWS Signature Version 4)를 참조하세요.

다음은 미리 서명된 URL의 예입니다. 가독성을 위해 이 예시에 줄바꿈이 추가됩니다.

```
https://s3.amazonaws.com/examplebucket/test.txt ?
X-Amz-Algorithm=AWS4-HMAC-SHA256 &
X-Amz-Credential=<your-access-key-id>/20130721/us-east-1/s3/aws4_request &
X-Amz-Date=20130721T201207Z &
X-Amz-Expires=86400 &
X-Amz-SignedHeaders=host &X-Amz-Signature=<signature-value>
```

> **Note**
>
> URL의 `X-Amz-Credential` 값은 가독성을 위해 '/' 문자만 표시합니다. 실제로는 %2F로 인코딩되어야 합니다. 예:
> `&X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request`

다음 표에서는 인증 정보를 제공하는 URL의 쿼리 파라미터를 설명합니다.

쿼리 문자열 파라미터 이름	설명
X-Amz-Algorithm	AWS 서명의 버전과 서명 계산에 사용한 알고리즘을 식별합니다. AWS 서명 버전 4의 경우 이 파라미터 값을 `AWS4-HMAC-SHA256` 으로 설정합니다. 이 문자열은 AWS 서명 버전 4(AWS4) 및 HMAC-SHA256 알고리즘(`HMAC-SHA256`)을 식별합니다.
X-Amz-Credential	이 파라미터는 액세스 키 ID 외에도 서명이 유효한 범위(AWS 리전 및 서비스)를 제공합니다. 이

쿼리 문자열 파라미터 이름	설명
	값은 다음 섹션의 설명과 같이 서명 계산에 사용하는 범위와 일치해야 합니다. 이 파라미터의 값의 일반적인 형식은 다음과 같습니다. `<your-access-key-id>/<date>/<AWS Region>/<AWS-service>/aws4_request` 예: `AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request` AWS 리전 문자열 목록은 AWS 일반 참조의 Regional Endpoints를 참조하세요.
X-Amz-Date	날짜 및 시간 형식은 ISO 8601 표준을 따르고, `yyyyMMddTHHmmssZ` 형식을 사용해야 합니다. 예를 들어 날짜와 시간이 '08/01/2016 15:32:41.982-700'인 경우 UTC(협정 세계시)로 변환하여 '20160801T223241Z'로 제출해야 합니다.
X-Amz-Expires	생성된 미리 서명된 URL이 유효한 기간(초)을 제공합니다. 예: 86400(24시간). 이 값은 정수입니다. 설정할 수 있는 최솟값은 1이고, 최댓값은 604800(7일)입니다. 서명 계산에 사용하는 서명 키는 최대 7일 동안 유효하므로 미리 서명된 URL은 최대 7일 동안 유효할 수 있습니다.

쿼리 문자열 파라미터 이름	설명
X-Amz-SignedHeaders	서명 계산에 사용한 헤더를 나열합니다. 서명 계산에는 다음 헤더가 필요합니다. • HTTP 호스트 헤더. • 요청에 추가하려는 모든 x-amz-* 헤더. 보안을 강화하려면 요청에 포함하려는 모든 요청 헤더에 서명해야 합니다.
X-Amz-Signature	요청을 인증하기 위한 서명을 제공합니다. 이 서명은 서비스에서 계산하는 서명과 일치해야 합니다. 그렇지 않으면 서비스가 요청을 거부합니다. 예제: `733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7` 서명 계산은 다음 섹션에서 설명합니다.
X-Amz-Security-Token	STS 서비스에서 가져온 보안 인증 정보를 사용하는 경우의 선택적 보안 자격 증명 파라미터입니다.

서명된 AWS API 요청 생성

Important

AWS SDK(샘플 코드 및 라이브러리 참조) 또는 AWS Command Line Interface(AWS CLI) 도구를 사용하여 API 요청을 AWS로 전송하는 경우 SDK 및 CLI 클라이언트는 사용자 요청 인증에 사용자가 제공한 액세스 키를 사용하므로 이 섹션을 건너뛰어도 됩니다. 그럴 만한 이유가 없는 경우 항상 SDK 또는 CLI를 사용하는 것이 좋습니다.
여러 서명 버전을 지원하는 리전에서는 수동 서명을 요청하는 경우 사용할 서명 버전을 지정해야 합니다. 다중 리전 액세스 포인트로 요청을 공급하는 경우 SDK와 CLI가 자동으로 추가 구성 없이 서명 버전 4A를 사용하는 것으로 전환됩니다.

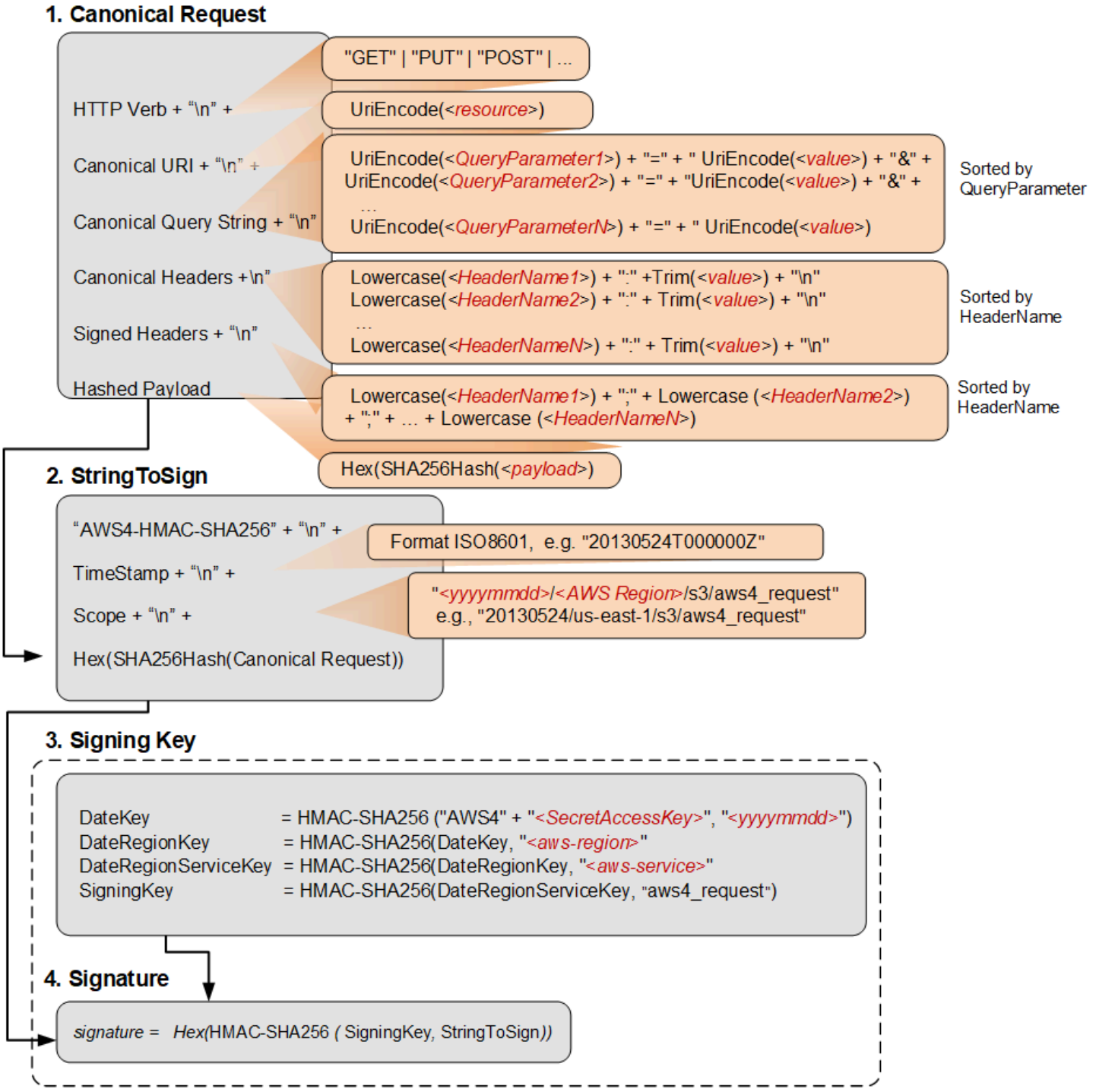
AWS SigV4 서명 프로토콜을 사용하여 AWS API 요청에 대한 서명된 요청을 생성할 수 있습니다.

1. 요청 세부 정보를 기반으로 표준 요청 생성.
2. AWS 자격 증명을 사용하여 서명 계산.
3. 이 서명을 요청에 Authorization 헤더로 추가.

AWS에서는 이 프로세스를 복제하고 서명을 확인하여 그에 따라 액세스를 허용하거나 거부합니다.

AWS SigV4를 사용하여 API 요청에 서명하는 방법은 서명 요청 예의 내용을 참조하세요.

다음 다이어그램은 서명을 위해 생성한 문자열의 다양한 구성 요소를 포함한 SigV4 서명 프로세스를 보여줍니다.



다음 표에서는 다이어그램에 표시된 함수를 설명합니다. 이러한 함수의 코드를 구현해야 합니다. 자세한 내용은 AWS SDK의 코드 예제를 참조하세요.

함수	설명
`Lowercase()`	문자열을 소문자로 변환합니다.
`Hex()`	소문자 16진법 인코딩.
`SHA256Hash()`	보안 해시 알고리즘(SHA) 암호화 해시 함수.
`HMAC-SHA256()`	제공된 서명 키를 포함한 SHA256 알고리즘을 사용하여 HMAC를 계산합니다. 이것이 최종 서명입니다.
`Trim()`	선행 또는 후행 공백을 모두 제거합니다.
`UriEncode()`	URI는 모든 바이트를 인코딩합니다. UriEncode ()는 다음 규칙을 적용해야 합니다. • URI는 예약되지 않은 문자를 제외한 모든 바이트를 인코딩합니다. 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_' 및 '~'. • 공백 문자는 예약된 문자이고 '+'가 아닌 '%20'로 인코딩되어야 합니다. • 각 URI 인코딩 바이트는 '%' 및 바이트의 두 자리 16진수 값으로 구성됩니다. • 16진수 값의 문자는 대문자여야 합니다(예: '%1A'). • 슬래시 문자('/')를 객체 키 이름을 제외한 모든 위치에 인코딩합니다. 예를 들어, 객체 키 이름이 `photos/Jan/sample.jpg` 인 경우 키 이름의 슬래시는 인코딩되지 않습니다. **Important** 개발 플랫폼에서 제공하는 표준 UriEncode 함수는 구현상의 차이와 기본 RFC의 관련 모호성으로 인해 효과적이지 않을 수 있습니다. 인코딩이 제대로

함수	설명
	작동하도록 사용자 지정 UriEncode 함수를 직접 작성하는 것이 좋습니다. Java에서 UriEncode 함수의 예를 보려면 GitHub 웹사이트에서 Java 유틸리티를 참조하세요.

> **Note**
>
> 요청에 서명할 때 AWS 서명 버전 4 또는 AWS 서명 버전 4A를 사용할 수 있습니다. 둘 사이의 주요 차이점은 서명 계산 방법에 따라 결정됩니다. AWS 서명 버전 4A의 경우 서명에 리전별 정보가 포함되지 않고 `AWS4-ECDSA-P256-SHA256` 알고리즘을 사용하여 계산됩니다.

임시 보안 자격 증명을 사용하여 요청 서명

장기 보안 인증 정보를 사용하여 요청에 서명하는 대신 AWS Security Token Service(AWS STS)에서 제공하는 임시 보안 인증 정보를 사용할 수 있습니다.

임시 보안 자격 증명을 사용하는 경우 Authorization 헤더 또는 쿼리 문자열에 `X-Amz-Security-Token`을 추가하거나 포함하여 세션 토큰을 저장해야 합니다. 일부 서비스의 경우 표준 요청에 `X-Amz-Security-Token`을 추가해야 합니다. 다른 서비스의 경우, 서명을 계산한 후에 `X-Amz-Security-Token`을 끝에 추가하기만 하면 됩니다. 구체적인 요구 사항은 각 AWS 서비스의 설명서를 참조하세요.

서명 단계 요약

표준 요청 생성:

요청 내용(호스트, 작업, 헤더 등)을 표준 형식으로 정렬합니다. 표준 요청은 서명할 문자열을 생성하는 데 사용되는 입력 중 하나입니다. 표준 요청 생성에 대한 자세한 내용은 AWS API 요청 서명의 요소의 내용을 참조하세요.

표준 요청의 해시 생성

페이로드의 해시를 생성하는 데 사용한 것과 동일한 알고리즘을 사용하여 표준 요청을 해시합니다. 표준 요청의 해시는 소문자 16진수 문자의 문자열입니다.

서명할 문자열 생성

표준 요청과 추가 정보(예: 알고리즘, 요청 날짜, 자격 증명 범위, 표준 요청의 해시)를 사용하여 서명할 문자열을 생성합니다.

서명 키 추출

AWS 비밀 액세스 키를 초기 해시 작업에 대한 키로 사용하여 요청 날짜, 리전 및 서비스에 대해 일련의 키가 추가된 해시 작업(HMAC)을 수행합니다.

서명 계산

추출된 서명 키를 해시 키로 사용하여 서명할 문자열에 대해 키가 추가된 해시 작업(HMAC)을 수행합니다.

요청에 서명 추가

HTTP 헤더 또는 요청의 쿼리 문자열에 계산된 서명을 추가합니다.

표준 요청 생성

표준 요청을 생성하려면 다음 문자열을 줄 바꿈 문자로 구분해 연결합니다. 이렇게 하면 사용자가 계산하는 서명이 AWS에서 계산하는 서명과 일치하도록 할 수 있습니다.

```
<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>
```

- *HTTPMethod* - HTTP 메서드(예:GET, PUT, HEAD 및 DELETE).
- *CanonicalUri* – 도메인 이름 뒤에 오는 / 문자로 시작하여 문자열의 끝 또는 쿼리 문자열 파라미터가 있는 경우 물음표 문자(?)까지의 절대 경로 구성 요소 URI의 URI 인코딩 버전입니다. 절대 경로가 비어있는 경우, 슬래시 문자(/)를 사용합니다. 다음 예제에서 URI /examplebucket/myphoto.jpg는 절대 경로이며 절대 경로에는 / 문자를 인코딩하지 않습니다.

  ```
  http://s3.amazonaws.com/examplebucket/myphoto.jpg
  ```

- *CanonicalQueryString* - URI 인코딩 쿼리 문자열 파라미터. 각 이름과 값을 개별적으로 URI 인코딩합니다. 또한 표준 쿼리 문자열의 파라미터를 키 이름별로 알파벳순으로 정렬해야 합니다. 정렬은 인코딩 후에 이루어집니다. 다음 URI 예제의 쿼리 문자열은 다음과 같습니다.

```
http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=2
```

표준 쿼리 문자열은 다음과 같습니다(가독성을 위해 줄바꿈 추가됨).

```
UriEncode("marker")+"="+UriEncode("someMarker")+"&"+
UriEncode("max-keys")+"="+UriEncode("20") + "&" +
UriEncode("prefix")+"="+UriEncode("somePrefix")
```

요청이 하위 리소스를 대상으로 하는 경우 해당 쿼리 파라미터 값은 빈 문자열("")이 됩니다. 예를 들어, 다음 URI는 examplebucket 버킷에서 ACL 하위 리소스를 식별합니다.

```
http://s3.amazonaws.com/examplebucket?acl
```

이 경우 CanonicalQueryString은 다음과 같습니다.

```
UriEncode("acl") + "=" + ""
```

URI에 ? 문자가 포함되지 않은 경우 요청에는 쿼리 문자열이 없으며, 표준 쿼리 문자열을 빈 문자열("")로 설정합니다. 여전히 줄 바꿈 문자("\n")를 포함해야 합니다.

- *CanonicalHeaders* - 요청 헤더와 해당 값이 포함된 목록. 개별 헤더 이름과 값 페어는 줄 바꿈 문자("\n")로 구분됩니다. 다음은 CanonicalHeader의 예제입니다.

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"
...
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

CanonicalHeaders 목록에는 다음이 포함되어야 합니다.

- HTTP host 헤더
- Content-Type 헤더가 요청에 있는 경우 이를 *CanonicalHeaders* 목록에 추가해야 합니다.

- 요청에 포함할 `x-amz-*` 헤더 또한 추가되어야 합니다. 예를 들어, 임시 보안 인증 정보를 사용하는 경우 요청에 `x-amz-security-token`을 포함해야 합니다. 이 헤더를 *CanonicalHeaders* 목록에 추가해야 합니다.

> **Note**
>
> Amazon S3 AWS 요청에는 `x-amz-content-sha256` 헤더가 필요합니다. 요청 페이로드의 해시를 제공합니다. 페이로드가 없는 경우 빈 문자열의 해시를 제공해야 합니다.

각 헤더 이름은 다음과 같아야 합니다.

- 소문자를 사용합니다.
- 알파벳 순서로 표시됩니다.
- 뒤에 콜론(`:`)이 와야 합니다.

값의 경우 다음을 수행해야 합니다.

- 선행 또는 후행 공백을 모두 잘라냅니다.
- 순차적 공백을 단일 공백으로 변환합니다.
- 쉼표를 사용하여 다중 값 헤더의 값을 구분합니다.
- host 헤더(HTTP/1.1) 또는 :authority 헤더(HTTP/2) 및 모든 `x-amz-*` 헤더를 서명에 포함해야 합니다. 서명에 content-type과 같은 다른 표준 헤더를 선택적으로 포함할 수 있습니다.

이 예제에서 사용하는 `Lowercase()` 및 `Trim()` 함수는 이전 섹션에서 설명합니다.

다음은 `CanonicalHeaders` 문자열의 예입니다. 헤더 이름은 소문자이고 정렬되어 있습니다.

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

> **Note**
>
> 승인 서명을 계산하기 위해 host 및 모든 `x-amz-*` 헤더만 필요합니다. 하지만 데이터 변조 방지를 위해 서명 계산에 모든 헤더를 포함하는 것이 좋습니다.

- *SignedHeaders* - 알파벳순으로 정렬되고 세미콜론으로 구분된 소문자 요청 헤더 이름 목록입니다. 목록의 요청 헤더는 CanonicalHeaders 문자열에 포함한 것과 동일합니다. 이전 예제에서 *SignedHeaders*의 값은 다음과 같습니다.

```
host;x-amz-content-sha256;x-amz-date
```

- *HashedPayload* - HTTP 요청 본문의 페이로드를 해시 함수의 입력으로 사용하여 생성된 문자열입니다. 이 문자열은 소문자 16진수 문자를 사용합니다.

```
Hex(SHA256Hash(<payload>))
```

요청에 페이로드가 없는 경우 빈 문자열의 해시를 계산합니다. 예를 들어, GET 요청을 사용하여 객체를 검색하는 경우 페이로드에 아무 것도 없습니다.

```
Hex(SHA256Hash(""))
```

> **Note**
>
> Amazon S3의 경우 표준 요청을 생성할 때 리터럴 문자열 UNSIGNED-PAYLOAD를 포함하고, 요청을 보낼 때는 x-amz-content-sha256 헤더 값과 동일한 값을 설정합니다. Hex(SHA256Hash("UNSIGNED-PAYLOAD"))

표준 요청의 해시 생성

페이로드의 해시를 생성하는 데 사용한 것과 동일한 알고리즘을 사용하여 표준 요청의 해시(다이제스트)를 생성합니다. 표준 요청의 해시는 소문자 16진수 문자의 문자열입니다.

서명할 문자열 생성

서명할 문자열을 생성하려면 다음 문자열을 줄 바꿈 문자로 구분하여 연결합니다. 이 문자열을 줄 바꿈 문자로 끝내지 마세요.

```
Algorithm \n
RequestDateTime \n
CredentialScope  \n
HashedCanonicalRequest
```

- *Algorithm* - 표준 요청의 해시를 생성하는 데 사용되는 알고리즘입니다. SHA-256의 경우, 알고리즘은 AWS4-HMAC-SHA256입니다.
- *RequestDateTime* - 보안 인증 범위에 사용된 날짜 및 시간입니다. 이 값은 ISO 8601 형식의 현재 UTC 시간입니다 (예: 20130524T000000Z).
- *CredentialScope* - 결과로 나오는 서명을 지정된 리전 및 서비스로 제한하는 자격 증명 범위입니다. 문자열의 형식은 *YYYYMMDD*/*region*/*service*/aws4_request입니다.
- *HashedCanonicalRequest* - 이전 단계에서 계산된 표준 요청의 해시입니다.

다음은 서명할 문자열의 예입니다.

```
"AWS4-HMAC-SHA256" + "\n" +
timeStampISO8601Format + "\n" +
<Scope> + "\n" +
Hex(SHA256Hash(<CanonicalRequest>))
```

서명 키 추출

서명 키를 추출하기 위해 AWS 비밀 액세스 키를 초기 해시 작업에 대한 키로 사용하여 요청 날짜, 리전 및 서비스에 대해 일련의 키가 추가된 해시 작업(HMAC)을 수행합니다.

각 단계마다 필요한 키와 데이터를 사용하여 해시 함수를 호출합니다. 해시 함수에 대한 각 호출의 결과는 다음 해시 함수 호출의 입력이 됩니다.

다음 예제는 이 절차의 다음 섹션에서 사용된 SigningKey 추출 방법을 보여주며, 입력이 연결되고 해시되는 순서를 보여줍니다. 보이는 것처럼 HMAC-SHA256은 데이터를 해시하는 데 사용되는 해시 함수입니다.

```
DateKey = HMAC-SHA256("AWS4"+"<SecretAccessKey>", "<YYYYMMDD>")
DateRegionKey = HMAC-SHA256(<DateKey>, "<aws-region>")
DateRegionServiceKey = HMAC-SHA256(<DateRegionKey>, "<aws-service>")
SigningKey = HMAC-SHA256(<DateRegionServiceKey>, "aws4_request")
```

필수 입력

- 비밀 액세스 키가 포함된 문자열 Key.
- 자격 증명 범위에 사용된 날짜가 YYYYMMDD 형식으로 포함된 문자열 Date.
- 리전 코드(예: us-east-1)가 포함된 문자열 Region.

리전 문자열 목록은 AWS 일반 참조의 리전 엔드포인트를 참조하세요.

- 서비스 코드(예: ec2)가 포함된 문자열 Service.
- 이전 단계에서 생성한 서명할 문자열입니다.

서명 키 추출

1. "AWS4"와 비밀 액세스 키를 연결합니다. 연결된 문자열을 키로, 날짜 문자열을 데이터로 각각 사용하여 해시 함수를 호출합니다.

```
DateKey = hash("AWS4" + Key, Date)
```

2. 이전 호출의 결과를 키로, 리전 문자열을 데이터로 각각 사용하여 해시 함수를 호출합니다.

```
DateRegionKey = hash(kDate, Region)
```

3. 이전 호출의 결과를 키로, 서비스 문자열을 데이터로 각각 사용하여 해시 함수를 호출합니다.

   서비스 코드는 서비스에 의해 정의합니다. AWS Pricing CLI의 get-products를 사용하여 서비스의 서비스 코드를 반환할 수 있습니다.

```
DateRegionServiceKey = hash(kRegion, Service)
```

4. 이전 호출의 결과를 키로, 'aws4_request'를 데이터로 각각 사용하여 해시 함수를 호출합니다.

```
SigningKey = hash(kService, "aws4_request")
```

서명 계산

서명 키를 추출한 후 서명할 문자열에 대해 키가 추가된 해시 작업을 수행하여 서명을 계산합니다. 파생된 서명 키를 이 작업에 대한 해시 키로 사용합니다.

서명을 계산하려면

1. 이전 호출의 결과를 키로, 서명할 문자열을 데이터로 각각 사용하여 해시 함수를 호출합니다. 이진수 값 형식의 서명이 결과로 반환됩니다.

```
signature = hash(SigningKey, string-to-sign)
```

2. 서명을 이진수에서 소문자의 16진수 표현으로 변환합니다.

요청에 서명 추가

계산된 서명을 요청에 추가합니다.

Example 예: 인증 헤더

다음은 `DescribeInstances` 작업의 `Authorization` 헤더를 보여 주는 예입니다. 이 예에서는 가독성을 높이기 위해 줄 바꿈을 사용했습니다. 코드에서는 이 문자열이 연속된 문자열이어야 합니다. 알고리즘과 `Credential` 사이에 쉼표는 없습니다. 단, 다른 요소는 쉼표로 구분해야 합니다.

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20220830/us-east-1/ec2/aws4_request,
SignedHeaders=host;x-amz-date,
Signature=calculated-signature
```

Example 예: 쿼리 문자열에 인증 파라미터가 있는 요청

다음 예에서는 인증 정보를 포함하는 `DescribeInstances` 작업에 대한 쿼리를 보여줍니다. 이 예에서는 가독성을 높이기 위해 줄 바꿈을 사용했으며 URL로 인코딩하지 않았습니다. 코드에서 이 쿼리 문자열은 URL로 인코딩되고 연속된 문자열이어야 합니다.

```
https://ec2.amazonaws.com/?
Action=DescribeInstances&
Version=2016-11-15&
X-Amz-Algorithm=AWS4-HMAC-SHA256&
X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20220830/us-east-1/ec2/aws4_request&
X-Amz-Date=20220830T123600Z&
X-Amz-SignedHeaders=host;x-amz-date&
X-Amz-Signature=calculated-signature
```

서명 요청 예

다음 AWS 서명 요청 예는 SigV4를 사용하여 AWS SDK 또는 AWS 명령줄 도구 없이 전송된 요청에 서명하는 방법을 보여줍니다.

HTTP POST를 사용한 브라우저 기반 Amazon S3 업로드

요청 인증: 브라우저 기반 업로드에서는 Amazon S3가 요청 수신 시 서명을 계산하는 데 사용하는 서명 및 관련 정보를 설명합니다.

예: HTTP POST를 사용한 브라우저 기반 업로드(AWS Signature Version 4 사용)에서는 샘플 POST 정책 및 파일 업로드에 사용할 수 있는 양식과 함께 자세한 정보를 제공합니다. 예제 정책과 가상 보안 인증 정보는 워크플로와 그에 따른 서명 및 정책 해시를 보여줍니다.

VPC Lattice 인증 요청

Signature Version 4(SigV4) 인증 요청의 예에서는 사용자 지정 인터셉터를 사용하거나 사용하지 않고 요청 서명을 수행하는 방법을 보여주는 Python 및 Java 예제를 제공합니다.

Amazon Translate에서 Signature Version 4 사용

Live Translations in the Metaverse에서는 실시간에 가까운 번역 솔루션을 생성하는 애플리케이션을 빌드하는 방법을 보여줍니다. 이 음성-음성 번역 솔루션은 이벤트 스트림 인코딩에서 AWS SigV4를 사용하여 실시간 트랜스크립션을 생성합니다.

Neptune에서 Signature Version 4 사용

예: Signature Version 4 서명으로 Python을 사용하여 Neptune에 연결에서는 Python을 사용하여 Neptune에 서명된 요청을 보내는 방법을 보여줍니다. 이 예에는 액세스 키 또는 임시 보안 인증 정보를 사용하기 위한 변형이 포함되어 있습니다.

S3 Glacier에 대한 HTTP 요청 서명

스트리밍 API의 서명 계산 예시는 S3 Glacier의 두 스트리밍 API 중 하나인 아카이브 업로드(POST 아카이브) 서명을 만드는 세부 사항을 안내합니다.

Amazon SWF에 대한 HTTP 요청 만들기

Amazon SWF에 대한 HTTP 요청 만들기는 Amazon SWF에 대한 JSON 요청의 헤더 내용을 보여줍니다.

Amazon OpenSearch Service의 스트리밍 API에 대한 서명 계산

PHP 버전 3용 AWS SDK로 Amazon OpenSearch Service 검색 요청 서명하기에는 Amazon OpenSearch Service에 서명된 HTTP 요청을 보내는 방법에 대한 예시가 포함되어 있습니다.

AWS 샘플 리포지토리의 예시 프로젝트

다음 예시 프로젝트에서는 Python, Node.js, Java, C#, Go 및 Rust와 같은 공통 언어를 사용하는 AWS 서비스에 대해 Rest API 요청을 만들기 위해 요청에 서명하는 방법을 보여줍니다.

Signature 버전 4a 프로젝트

sigv4-signing-examples 프로젝트는 Python, Node.js, Java, C#, Go, Rust와 같은 일반 언어를 사용하는 AWS 서비스에 대해 Rest API 요청을 만들기 위해 SigV4A로 요청에 서명하는 방법의 예시를 제공합니다.

sigv4a-signing-examples 프로젝트는 다중 리전 API 요청에 서명하는 예를 제공합니다(예: Amazon S3의 다중 리전 액세스 포인트).

AWS IoT Core에 게시

HTTPs 프로토콜을 사용하여 AWS IoT Core에 게시하는 Python 코드는 HTTPS 프로토콜 및 AWS SigV4 인증을 사용하여 AWS IoT Core에 메시지를 게시하는 방법에 대한 지침을 제공합니다. 두 가지 참조 구현이 있으며 하나는 Python, 다른 하나는 NodeJ로 되어 있습니다.

HTTPs 프로토콜을 사용하여 AWS IoT Core에 게시하는 .Net Framework 애플리케이션은 HTTPS 프로토콜 및 AWS SigV4 인증을 사용하여 AWS IoT Core에 메시지를 게시하는 방법에 대한 지침을 제공합니다. 이 프로젝트에는 .NET 코어에 상응하는 구현도 포함되어 있습니다.

AWS API 요청에 대한 Signature Version 4 서명 문제 해결

> **Important**
>
> AWS SDK 또는 CLI를 사용하지 않는 한 요청에 인증 정보를 제공하는 서명을 계산하는 코드를 작성해야 합니다. SigV4 서명 계산은 복잡한 작업일 수 있으므로 가능한 경우 AWS SDK 또는 CLI를 사용하는 것이 좋습니다.

서명된 요청을 생성하는 코드를 개발할 때, AWS 서비스에서 HTTP 403 `SignatureDoesNotMatch`가 발생할 수 있습니다. 이 오류는 AWS에 대한 HTTP 요청의 서명 값이 AWS 서비스에서 계산한 서명과 일치하지 않음을 의미합니다. 권한에서 호출자의 요청을 허용하지 않는 경우 HTTP 401 `Unauthorized` 오류가 반환됩니다.

다음과 같은 경우 API 요청에서 오류가 반환될 수 있습니다.

- API 요청이 서명되지 않았으며 API 요청이 IAM 인증을 사용합니다.
- 요청에 서명하는 데 사용된 IAM 보안 인증이 잘못되었거나 API를 간접적으로 호출할 권한이 없습니다.
- 서명된 API 요청의 서명이 AWS 서비스에서 계산한 서명과 일치하지 않습니다.
- API 요청 헤더가 잘못되었습니다.

> **Note**
>
> 다른 오류 해결 방법을 살펴보기 전에 AWS Signature Version 2(SigV2)에서 AWS Signature Version 4(SigV4)로 서명 프로토콜을 업데이트합니다. Amazon S3 등의 서비스와 리전은 더 이상 SIGv2 서명을 지원하지 않습니다.

가능한 원인

- 보안 인증 오류
- 표준 요청 및 서명 문자열 오류
- 보안 인증 범위 오류
- 키 서명 오류

보안 인증 오류

API 요청이 SigV4로 서명되었는지 확인합니다. API 요청이 서명되지 않은 경우 다음과 같은 오류가 발생할 수 있습니다. `Missing Authentication Token`. 누락된 서명을 추가하고 요청을 다시 전송합니다.

액세스 키 및 비밀 키의 보안 인증이 올바른지 확인합니다. 액세스 키가 잘못된 경우 다음과 같은 오류가 발생할 수 있습니다. `Unauthorized`. 요청에 서명하는 데 사용된 엔터티에 요청을 할 권한이 있는지 확인합니다. 세부 정보는 액세스 거부 오류 메시지 문제 해결을 참조하세요.

표준 요청 및 서명 문자열 오류

표준 요청의 해시 생성 또는 서명할 문자열 생성에서 표준 요청을 잘못 계산한 경우 서비스에서 수행되는 서명 검증 단계가 실패하고 다음 오류 메시지가 표시됩니다.

```
The request signature we calculated does not match the signature you provided
```

AWS 서비스에서 서명된 요청을 수신하면 서명을 다시 계산합니다. 값에 차이가 있으면 서명이 일치하지 않는 것입니다. 표준 요청 및 서명된 요청의 문자열을 오류 메시지의 값과 비교합니다. 차이가 있는 경우 서명 프로세스를 수정합니다.

> **Note**
>
> 헤더 또는 요청을 수정하는 프록시를 통해 요청을 전송하지 않았는지 확인할 수도 있습니다.

Example 정식 요청 예

```
GET                                                                -------- HTTP method
/                                                                  -------- Path. For API stage
 endpoint, it should be /{stage-name}/{resource-path}
                                                                   -------- Query string key-
value pair. Leave it blank if the request doesn't have a query string.
content-type:application/json                                      -------- Header key-value
 pair. One header per line.
host:0123456789.execute-api.us-east-1.amazonaws.com      -------- Host and x-amz-date
 are required headers for all signed requests.
x-amz-date:20220806T024003Z

content-type;host;x-amz-date                                       -------- A list of signed
 headers
d167e99c53f15b0c105101d468ae35a3dc9187839ca081095e340f3649a04501        -------- Hash
 of the payload
```

비밀 키가 액세스 키 ID와 일치하는지 확인하려면, 정상적으로 작동하는 것으로 알려진 구현을 사용하여 비밀 키를 테스트합니다. 예를 들어 AWS SDK 또는 AWS CLI를 사용하여 AWS에 대한 요청을 보냅니다.

API 요청 헤더

인증 헤더가 비어 있거나, 자격 증명 키 또는 서명이 없거나 올바르지 않거나, 헤더가 알고리즘 이름으로 시작하지 않거나, 키 값 쌍에 등호가 포함되지 않은 경우 다음 오류 중 하나가 발생합니다.

- 인증 헤더는 비워 둘 수 없습니다.
- 인증 헤더에는 '자격 증명' 매개변수가 필요합니다.
- 인증 헤더에는 '서명' 매개변수가 필요합니다.
- 인증 헤더에서 잘못된 키=값 쌍(등호 누락)이 서명에 포함되어 있습니다.

서명 계산에서 추가한 SigV4 인증 헤더에 올바른 자격 증명 키가 포함되어 있는지 및 HTTP 날짜 또는 `x-amz-date` 헤더를 사용하는 요청 날짜도 포함되어 있는지 확인합니다.

IncompleteSignatureException 오류가 발생했고 서명의 구성이 올바른 경우 클라이언트측 요청에서 권한 부여 헤더의 SHA-256 해시 및 B64 인코딩을 계산하여 AWS 서비스로 전송되는 동안 인증 헤더가 수정되지 않았는지 확인할 수 있습니다.

1. 요청에서 전송한 인증 헤더를 가져옵니다. 인증 헤더는 다음 예제와 유사합니다.

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=example-generated-signature
```

2. 인증 헤더의 SHA-256 해시를 계산합니다.

```
hashSHA256(rawAuthorizationHeader) = hashedAuthorizationHeader
```

3. 해시된 인증 헤더를 Base64 형식으로 인코딩합니다.

```
base64(hashedAuthorizationHeader) = encodedHashedAuthorizationHeader
```

4. 방금 계산한 해시 및 인코딩된 문자열을 오류 메시지로 받은 문자열과 비교합니다. 오류 메시지는 다음 예와 유사해야 합니다.

```
com.amazon.coral.service#IncompleteSignatureException:
The signature contains an in-valid key=value pair (missing equal-sign)
in Authorization header (hashed with SHA-256 and encoded with Base64):
'9c574f83b4b950926da4a99c2b43418b3db8d97d571b5e18dd0e4f3c3ed1ed2c'.
```

- 두 해시가 다른 경우 전송 중에 인증 헤더의 일부가 변경된 것입니다. 이는 네트워크 또는 클라이언트 핸들러가 서명된 헤더를 연결하거나 인증 헤더를 어떤 식으로든 변경했기 때문일 수 있습니다.
- 두 해시가 일치하면 요청에서 보낸 인증 헤더가 AWS에서 받은 것과 일치합니다. 수신된 오류 메시지를 검토하여 문제가 잘못된 자격 증명이나 서명으로 인한 것인지 확인합니다. 이러한 오류는 이 페이지의 다른 섹션에서 다룹니다.

보안 인증 범위 오류

서명할 문자열 생성에서 생성한 보안 인증 범위는 서명을 특정 날짜, 리전 및 서비스로 제한합니다. 이 문자열의 형식은 다음과 같습니다.

```
YYYYMMDD/region/service/aws4_request
```

> **Note**
>
> SigV4a를 사용하는 경우 해당 리전은 보안 인증 범위에 포함되지 않습니다.

날짜

보안 인증 범위에 x-amz-date 헤더와 동일한 날짜가 지정되어 있지 않은 경우, 서명 확인 단계가 실패하고 다음 오류 메시지가 표시됩니다.

```
Date in Credential scope does not match YYYYMMDD from ISO-8601 version of date from
 HTTP
```

요청에 미래의 시간이 지정되어 있는 경우, 서명 확인 단계가 실패하고 다음 오류 메시지가 표시됩니다.

```
Signature not yet current: date is still later than date
```

요청이 만료된 경우, 서명 확인 단계가 실패하고 다음 오류 메시지가 표시됩니다.

```
Signature expired: date is now earlier than date
```

리전

보안 인증 범위가 요청과 동일한 리전을 지정하지 않는 경우, 서명 확인 단계가 실패하고 다음 오류 메시지가 표시됩니다.

```
Credential should be scoped to a valid Region, not region-code
```

Service

보안 인증 범위에 host 헤더와 동일한 서비스가 지정되어 있지 않은 경우, 서명 확인 단계가 실패하고 다음 오류 메시지가 표시됩니다.

```
Credential should be scoped to correct service: 'service'
```

종료 문자열

보안 인증 범위가 aws4_request로 끝나지 않는 경우, 서명 확인 단계가 실패하고 다음 오류 메시지가 표시됩니다.

```
Credential should be scoped with a valid terminator: 'aws4_request'
```

키 서명 오류

잘못된 서명 키 추출 또는 잘못된 암호화 사용으로 인해 발생하는 오류는 해결하는 데 어려움이 있습니다. 표준 문자열과 서명할 문자열이 올바른지 확인한 후, 다음 문제 중 하나를 확인할 수도 있습니다.

- 비밀 액세스 키가 지정된 액세스 키 ID와 일치하지 않습니다.
- 키 추출 코드에 문제가 있습니다.

비밀 키가 액세스 키 ID와 일치하는지 확인하려면, 정상적으로 작동하는 것으로 알려진 구현을 사용하여 비밀 키를 테스트합니다. 예를 들어 AWS SDK 또는 AWS CLI를 사용하여 AWS에 대한 요청을 보냅니다. 예제는 서명 요청 예 단원을 참조하십시오.

IAM JSON 정책 참조

이 섹션에서는 IAM에서 JSON 정책의 자세한 구문과 설명, 요소의 예, 변수, 평가 로직을 설명합니다. 더 일반적인 내용은 JSON 정책 개요 섹션을 참조하세요.

본 참조는 다음 섹션을 포함합니다:

- IAM JSON 정책 요소 참조 - 정책을 생성할 때 사용할 수 있는 요소에 대해 자세히 알아봅니다. 더 많은 정책 예제를 보면서 조건, 지원되는 데이터 유형, 다양한 서비스에서 사용되는 방법을 살펴봅니다.
- 정책 평가 로직 - 이 섹션에서는 AWS 요청, 그 요청이 인증되는 방식 및 AWS가 정책을 사용하여 리소스에 대한 액세스를 결정하는 방식을 기술합니다.
- IAM JSON 정책 언어의 문법 - 이 섹션은 IAM에서 정책 생성에 사용되는 언어에 대한 공식 구문을 설명합니다.
- 직무에 관한 AWS 관리형 정책 - 이 단원에서는 IT 업계의 일반적인 직무와 직접 매핑되는 모든 AWS 관리형 정책이 나열됩니다. 이러한 정책을 사용하여 특정 직무 담당자에게 기대되는 작업 수행에 필요한 권한을 부여할 수 있습니다. 이러한 정책은 다수의 서비스에 대한 권한을 단일 정책으로 통합합니다.
- AWS 글로벌 조건 컨텍스트 키 - 이 섹션에는 IAM 정책에서 권한을 제한하는 데 사용할 수 있는 모든 AWS 전역 조건 키 목록이 포함되어 있습니다.
- IAM 및 AWS STS 조건 컨텍스트 키 - 이 섹션에는 IAM 정책에서 권한을 제한하는 데 사용할 수 있는 모든 IAM 및 AWS STS 조건 키 목록이 포함되어 있습니다.

- AWS 서비스에 대한 작업, 리소스 및 조건 키 - 이 섹션에는 IAM 정책에서 권한으로 사용할 수 있는 모든 AWS API 작업을 나열합니다. 또한 요청을 추가로 미세 조정하는 데 사용할 수 있는 서비스별 조건 키도 포함되어 있습니다.

IAM JSON 정책 요소 참조

JSON 정책 문서는 여러 요소로 구성됩니다. 여기에 나열되는 요소들은 정책에서 사용되는 일반적인 순서를 따릅니다. 요소 순서는 중요하지 않습니다.예를 들어 `Resource` 요소는 `Action` 요소 앞에 올 수 있습니다. 또한 정책에서 `Condition` 요소는 지정하지 않아도 됩니다. JSON 정책 문서의 일반적인 구조와 목적에 대해 자세히 알아보려면 JSON 정책 개요를 참조하세요.

일부 JSON 정책 요소는 함께 사용할 수 없습니다. 즉, 둘 다 사용하는 정책을 생성할 수 없습니다. 예를 들어 동일한 정책 문에서 `Action`과 `NotAction` 둘 다 사용할 수 없습니다. 함께 사용할 수 없는 다른 쌍에는 `Principal`/`NotPrincipal` 및 `Resource`/`NotResource`가 있습니다.

정책 세부 정보는 서비스에서 유효한 작업이나 추가되는 리소스 유형 등에 따라 각 서비스마다 차이가 있습니다. 따라서 특정 서비스에 맞는 정책을 작성할 때는 해당 서비스의 정책 예제를 살펴보는 것이 좋습니다. IAM을 지원하는 모든 서비스 목록을 비롯해 각 서비스의 IAM 및 정책 설명서 링크는 AWS IAM으로 작업하는 서비스 섹션을 참조하세요.

JSON 정책을 생성하거나 편집할 때 IAM은 효과적인 정책을 생성하는 데 도움이 되는 정책 검증을 수행할 수 있습니다. IAM은 JSON 구문 오류를 식별하는 반면, IAM Access Analyzer는 정책을 더욱 구체화하는 데 도움이 되는 권장 사항과 함께 추가 정책 검사를 제공합니다. 정책 검증에 대한 자세한 내용은 IAM 정책 검증 섹션을 참조하세요. IAM Access Analyzer 정책 검사기 및 실행 가능한 권장 사항에 대한 자세한 내용은 IAM Access Analyzer 정책 검증을 참조하세요.

주제

- IAM JSON 정책 요소: Version
- IAM JSON 정책 요소: Id
- IAM JSON 정책 요소: Statement
- IAM JSON 정책 요소: Sid
- IAM JSON 정책 요소: Effect
- AWS JSON 정책 요소: Principal
- AWS JSON 정책 요소: NotPrincipal
- IAM JSON 정책 요소: Action
- IAM JSON 정책 요소: NotAction

- IAM JSON 정책 요소: Resource
- IAM JSON 정책 요소: NotResource
- IAM JSON 정책 요소: Condition
- IAM 정책 요소: 변수 및 태그
- IAM JSON 정책 요소: 지원되는 데이터 형식

IAM JSON 정책 요소: Version

> **동음이의어 참고**
>
> Version JSON 정책 요소는 정책 버전과 다릅니다. Version 정책 요소는 정책 내에서 사용되며 정책 언어의 버전을 정의합니다. 반면에 정책 버전은 IAM에서 고객 관리형 정책을 변경할 때 생성됩니다. 변경된 정책은 기존 정책을 덮어쓰지 않습니다. 대신 IAM에서 관리형 정책의 새 버전을 생성합니다. 관리형 정책에서 사용할 수 있는 여러 버전 지원에 대한 정보를 찾고 있다면 the section called “IAM 정책 버전 관리” 섹션을 참조하세요.

Version 정책 요소는 정책의 처리에 사용할 언어 구문 규칙을 지정합니다. 사용 가능한 모든 정책 기능을 사용하려면 모든 정책의 Statement 요소 외부에 다음 Version 요소를 포함시킵니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

IAM은 다음과 같은 Version 요소 값을 지원합니다.

- 2012-10-17. 이 값은 정책 언어의 현재 버전이며, 항상 Version 요소를 포함하여 2012-10-17로 설정해야 합니다. 그렇지 않으면 이 버전에 채택되지 않은 정책 변수 등의 기능을 사용할 수 없습니다.
- 2008-10-17. 이 값은 이전 정책 언어 버전입니다. 따라서 오래된 기존 정책에서는 이 버전이 표시될 수도 있습니다. 새로운 정책에서는 또는 기존 정책을 업데이트하는 경우에는 이 버전을 사

용하지 마세요. 정책 변수와 같은 최신 기능이 사용자의 정책에서 작동하지 않습니다. 예를 들어 `${aws:username}` 같은 변수가 정책에서 변수로 인식되지 않고 리터럴 문자열로 취급됩니다.

IAM JSON 정책 요소: Id

`Id` 요소는 정책 식별자(옵션)를 지정합니다. 다른 서비스의 ID 사용 방법과는 다릅니다. ID는 리소스 기반 정책에서는 허용되지만 ID 기반 정책에서는 사용할 수 없습니다.

`ID` 요소를 설정할 수 있는 서비스의 경우에는 UUID(GUID)를 값으로 사용하거나, UUID를 ID 일부로 사용하여 고유성을 확보하는 것이 좋습니다.

```
{
  "Version": "2012-10-17",
  "Id": "cd3ad3d9-2776-4ef1-a904-4c229d1642ee",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> AWS 서비스(예: Amazon SQS 또는 Amazon SNS 등) 중에는 이 요소가 필요하거나 고유성 요건을 따로 요구하는 경우도 있습니다. 정책 작성에 대한 서비스별 정보는 이용하려는 서비스의 설명서를 참조하세요.

IAM JSON 정책 요소: Statement

`Statement` 요소는 정책의 주요 요소로서 필수입니다. `Statement` 요소는 단일 문 또는 개별 문의 배열을 포함할 수 있습니다. 각 개별 문 블록은 중괄호 `{}`로 묶어야 합니다. 여러 문의 경우 배열은 대괄호 `[]`로 묶어야 합니다.

```
"Statement": [{...},{...},{...}]
```

다음은 단일 `Statement` 요소에서 3개의 문이 하나의 배열을 이루는 정책을 나타낸 예제입니다 (이 정책에서는 Amazon S3 콘솔에서 자신의 'home 폴더'에 액세스할 수 있습니다). 정책에는

aws:username 변수가 추가되었습니다. 이 변수는 정책 평가 중 요청이 있으면 사용자 이름으로 바뀝니다. 자세한 내용은 소개 섹션을 참조하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::BUCKET-NAME",
      "Condition": {"StringLike": {"s3:prefix": [
        "",
        "home/",
        "home/${aws:username}/"
      ]}}
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}",
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}/*"
      ]
    }
  ]
}
```

IAM JSON 정책 요소: Sid

정책 문에 입력되는 식별자(선택 사항)로 Sid(문 ID)를 제공할 수 있습니다. Sid 값은 문 배열에서 각 문에 할당할 수 있습니다. Sid 값을 정책 문에 대한 설명으로 사용할 수 있습니다. SQS나 SNS처럼 ID 요소를 지정할 수 있는 서비스에서는 Sid 값이 정책 문서 ID의 하위 ID나 마찬가지입니다. IAM에서 Sid 값은 JSON 정책 내 고유성이 보장되어야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExampleStatementID",
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

Sid 요소는 ASCII 대문자(A~Z), 소문자(a~z) 및 숫자(0~9)를 지원합니다.

IAM에서 Sid는 IAM API에 노출되지 않습니다. 따라서 이 ID를 근거로 특정 문을 가져올 수는 없습니다.

> **Note**
>
> AWS 서비스(예: Amazon SQS 또는 Amazon SNS 등) 중에는 이 요소가 필요하거나 고유성 요건을 따로 요구하는 경우도 있습니다. 정책 작성에 대한 서비스별 정보는 이용하려는 서비스의 설명서를 참조하세요.

IAM JSON 정책 요소: Effect

Effect 요소는 필수로서, 문의 허용(allow) 또는 명시적 거부(explicit deny) 중 하나를 지정합니다. Effect 유효값은 Allow 및 Deny입니다. Effect 값은 대소문자를 구분합니다.

```
"Effect":"Allow"
```

기본적으로 리소스 액세스는 거부됩니다. 리소스 액세스를 허용하려면 Effect 요소를 Allow로 설정해야 합니다. 허용을 재정의하려면(예: 그 밖에 다른 방법으로 실행 중인 허용을 재정의하려면) Effect 요소를 Deny로 설정합니다. 자세한 내용은 정책 평가 로직 섹션을 참조하세요.

AWS JSON 정책 요소: Principal

리소스 기반 JSON 정책의 Principal 요소를 사용하여 리소스에 대한 액세스가 허용되거나 거부되는 보안 주체를 지정합니다.

리소스 기반 정책의 `Principal` 요소를 사용해야 합니다. IAM을 비롯한 여러 서비스가 리소스 기반 정책을 지원합니다. IAM 리소스 기반 정책 유형은 역할 신뢰 정책입니다. IAM 역할에서 역할 신뢰 정책의 `Principal` 요소를 사용하여 역할을 수임할 수 있는 사용자를 지정합니다. 크로스 계정 액세스인 경우에는 신뢰할 수 있는 계정의 12자리 식별자를 지정합니다. 해당 신뢰 영역(신뢰할 수 있는 조직 또는 계정) 외의 계정 내 보안 주체가 역할을 수임하는 권한이 있는지 자세히 알고 싶다면, IAM Access Analyzer란 무엇일까요?를 참조하세요.

> **Note**
>
> 역할을 생성한 이후 계정을 "*"로 변경하여 모두가 이 역할을 수임하도록 할 수 있습니다. 이렇게 하는 경우 다른 방법(예: 특정 IP 주소로만 액세스를 제한하는 `Condition` 요소)을 통해 역할에 액세스할 수 있는 사용자를 제한하는 것이 좋습니다. 역할을 모두 액세스할 수 있는 상태로 두지 마세요.

리소스 기반 정책을 지원하는 리소스의 다른 예에는 Amazon S3 버킷 또는 AWS KMS key(이)가 포함됩니다.

자격 증명 기반 정책에서는 `Principal` 요소를 사용할 수 없습니다. 자격 증명 기반 정책은 IAM 자격 증명(사용자, 그룹 또는 역할)에 연결할 수 있는 권한 정책입니다. 이 경우, 보안 주체는 암시적으로 정책이 연결된 자격 증명입니다.

주제

- 보안 주체 지정
- AWS 계정 보안 주체
- IAM 역할 보안 주체
- 역할 세션 보안 주체
- IAM 사용자 보안 주체
- IAM Identity Center 보안 주체
- AWS STS 페더레이션 사용자 세션 보안 주체
- AWS 서비스 보안 주체
- 옵트인 리전용 AWS 서비스 주체
- 모든 보안 주체
- 추가 정보

보안 주체 지정

보안 주체를 지원하는 리소스 기반 정책이나 조건 키의 `Principal` 요소에 있는 보안 주체를 지정합니다.

정책에서 다음 보안 주체를 지정할 수 있습니다.

- AWS 계정 및 루트 사용자
- IAM 역할
- 역할 세션
- IAM 사용자
- 페더레이션 사용자 세션
- AWS 서비스
- 모든 보안 주체

그룹은 인증이 아니라 권한과 관련이 있고 보안 주체는 인증된 IAM 엔터티이기 때문에 정책(예: 리소스 기반 정책)에서 사용자 그룹을 보안 주체로 식별할 수 없습니다.

배열을 사용하여 다음 섹션에서 각 보안 주체 유형에 대해 둘 이상의 보안 주체를 지정할 수 있습니다. 배열은 하나 이상의 값을 갖습니다. 요소에 둘 이상의 보안 주체를 지정할 때는 각 보안 주체에 권한을 부여하십시오. 한 번에 하나의 보안 주체로 인증되기 때문에 이것은 논리적 `AND`(이)며 논리적 `OR`이(가) 아닙니다. 값을 두 개 이상 포함하는 경우 배열의 각 항목을 대괄호(`[` 및 `]`) 및 쉼표로 구분합니다. 다음 예시 정책은 123456789012 계정 또는 555555555555 계정에 대한 권한을 정의합니다.

```
"Principal" : {
"AWS": [
  "123456789012",
  "555555555555"
  ]
}
```

> **Note**
>
> 보안 주체 이름이나 ARN의 일부를 나타내기 위해 와일드카드를 사용할 수 없습니다.

AWS 계정 보안 주체

보안 주체를 지원하는 리소스 기반 정책이나 조건 키의 `Principal` 요소에 있는 AWS 계정 식별자를 지정할 수 있습니다. 이렇게 하면 계정에 권한을 위임합니다. 다른 계정에 대한 액세스를 허용하면 해당 계정의 관리자는 해당 계정의 자격 증명(IAM 사용자 또는 역할)에 대한 액세스 권한을 부여해야 합니다. AWS 계정을 지정하는 경우 계정 ARN(arn:aws:iam::*account-ID*:root)을 사용하거나 `"AWS":` 접두사와 그 뒤에 계정 ID가 따라오는 약식 형태를 사용할 수 있습니다.

예를 들어, 계정 ID 123456789012의 경우 다음 방법 중 하나를 사용하여 `Principal` 요소에서 해당 계정을 지정할 수 있습니다.

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

```
"Principal": { "AWS": "123456789012" }
```

계정 ARN과 단축된 계정 ID도 같은 방식으로 작동합니다. 둘 다 계정에 권한을 위임합니다. `Principal` 요소에서 계정 ARN 사용해도 계정의 루트 사용자에게만 권한이 제한되지는 않습니다.

> **Note**
>
> 단축된 계정 ID를 포함하는 리소스 기반 정책을 저장하면 서비스에서 이를 보안 주체 ARN으로 변환할 수 있습니다. 이렇게 해도 정책의 기능은 변경되지 않습니다.

일부 AWS 서비스는 계정 보안 주체를 지정하기 위한 몇 가지 옵션을 추가로 지원합니다. 예를 들어 Amazon S3에서는 다음 형식을 사용하여 정식 사용자 ID를 지정할 수 있습니다.

```
"Principal": { "CanonicalUser":
 "79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be" }
```

또한 배열을 사용하여 둘 이상의 AWS 계정(또는 표준 사용자 ID)을 보안 주체로 지정할 수 있습니다. 예를 들어, 세 가지 방법 모두를 사용하여 보안 주체를 버킷 정책에 지정할 수 있습니다.

```
"Principal": {
  "AWS": [
    "arn:aws:iam::123456789012:root",
    "999999999999"
  ],
  "CanonicalUser": "79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be"
```

```
}
```

IAM 역할 보안 주체

보안 주체를 지원하는 리소스 기반 정책이나 조건 키의 `Principal` 요소에 있는 IAM 역할 보안 주체 ARN을 지정할 수 있습니다. IAM 역할은 자격 증명입니다. IAM에서 자격 증명은 권한을 할당할 수 있는 리소스입니다. 역할은 다른 인증된 자격 증명을 신뢰하여 해당 역할을 수임합니다. 여기에는 AWS의 보안 주체 또는 외부 자격 증명 공급자(IdP)가 포함됩니다. 보안 주체 또는 자격 증명이 역할을 맡으면 수임된 역할의 권한이 있는 임시 보안 자격 증명을 수신합니다. AWS에서 작업을 수행하기 위해 이러한 세션 자격 증명을 사용하면 역할 세션 보안 주체가 됩니다.

IAM 역할은 IAM에 존재하는 자격 증명입니다. 역할은 AWS의 보안 주체 또는 외부 ID 공급자의 사용자와 같은 다른 인증된 자격 증명을 신뢰합니다. 보안 주체 또는 자격 증명이 역할을 맡으면 임시 보안 자격 증명을 수신합니다. 그런 다음 이러한 자격 증명을 역할 세션 보안 주체로 사용하여 AWS에서 작업을 수행할 수 있습니다.

리소스 기반 정책에서 역할 보안 주체를 지정하는 경우 보안 주체에 대한 유효 사용 권한은 해당 역할에 대한 사용 권한을 제한하는 모든 정책 유형에 의해 제한됩니다. 여기에는 세션 정책과 권한 경계가 포함됩니다. 역할 세션에 대한 유효한 사용 권한을 평가하는 방법에 대한 자세한 내용은 정책 평가 로직을(를) 참조하세요.

`Principal` 요소에서 ARN 역할을 지정하려면 다음 형식을 사용하십시오.

```
"Principal": { "AWS": "arn:aws:iam::AWS-account-ID:role/role-name" }
```

> **Important**
>
> 역할 신뢰 정책의 `Principal` 요소에 특정 IAM 역할을 가리키는 ARN이 포함되어 있으면, 정책을 저장할 때 해당 ARN이 해당 역할의 고유 보안 주체 ID로 변환됩니다. 그러면 누군가가 해당 역할을 제거하고 다시 만들어 본인의 권한을 에스컬레이션할 위험을 완화할 수 있습니다. IAM은 신뢰 정책이 표시될 때 역할 ARN로 역변환을 다시 사용하기 때문에 일반적으로 콘솔에서는 이 ID가 표시되지 않습니다. 그러나 해당 역할을 삭제하면 관계가 깨집니다. 역할을 다시 만들더라도 해당 정책이 더 이상 적용되지 않습니다. 새 역할의 새 보안 주체 ID가 신뢰 정책에 저장된 ID와 일치하지 않기 때문입니다. 이 경우 AWS가 더 이상 유효한 ARN에 다시 매핑할 수 없기 때문에 보안 주체 ID가 리소스 기반 정책에 나타납니다. 결과적으로 신뢰 정책의 `Principal` 요소에서 참조된 역할을 삭제하고 다시 만드는 경우, 보안 주체 ID를 올바른 ARN으로 바꾸도록 정책에서 역할을 편집해야 합니다. 정책을 저장하면 ARN이 다시 해당 역할의 새로운 보안 주체 ID로 변환됩니다.

또는 리소스 기반 정책에서 역할 보안 주체를 보안 주체로 지정하거나 `aws:PrincipalArn` 조건 키를 사용하는 광범위한 권한 정책을 생성할 수 있습니다. 이 키를 사용하면 역할 세션 보안 주체에게 결과 세션의 ARN이 아니라 수임된 역할의 ARN을 기반으로 권한이 부여됩니다. 이는 AWS은(는) 조건 키 ARN을 ID로 변환하지 않기 때문입니다. 역할 ARN에 부여된 권한은 역할을 삭제한 다음 동일한 이름으로 새 역할을 생성하면 유지됩니다. 자격 증명 기반 정책에 명시적 거부가 포함되지 않는 한, 권한 경계 또는 세션 정책과 같은 자격 증명 기반 정책 유형은 `Principal` 요소에서 와일드카드(*)와 함께 `aws:PrincipalArn` 조건 키를 사용하여 부여한 권한을 제한하지 않습니다.

역할 세션 보안 주체

보안 주체를 지원하는 리소스 기반 정책이나 조건 키의 `Principal` 요소에 있는 역할 세션을 지정할 수 있습니다. 보안 주체 또는 자격 증명이 역할을 맡으면 수임된 역할의 권한이 있는 임시 보안 자격 증명을 수신합니다. AWS에서 작업을 수행하기 위해 이러한 세션 자격 증명을 사용하면 역할 세션 보안 주체가 됩니다.

역할 세션 보안 주체에 사용하는 형식은 역할을 수임하는 데 사용된 AWS STS 작업에 따라 다릅니다.

또한 관리자는 역할 세션이 발행되는 방식을 제어하는 프로세스를 설계할 수 있습니다. 예를 들어 예측 가능한 세션 이름을 생성하는 원클릭 솔루션을 사용자에게 제공할 수 있습니다. 관리자가 이를 수행하는 경우, 정책 또는 조건 키에서 역할 세션 보안 주체를 사용할 수 있습니다. 그렇지 않으면 역할 ARN을 `aws:PrincipalArn` 조건 키의 보안 주체로 지정할 수 있습니다. 역할을 보안 주체로 지정하는 방법에 따라 결과 세션에 대한 유효한 권한이 변경될 수 있습니다. 자세한 내용은 IAM 역할 보안 주체 단원을 참조하십시오.

수임된 역할 세션 보안 주체

수임된 역할 세션 보안 주체는 AWS STS `AssumeRole` 작업을 사용하여 발생하는 세션 보안 주체입니다. 이 작업을 사용하여 역할을 수임할 수 있는 보안 주체에 대한 자세한 내용은 AWS STS API 작업 비교을(를) 참조하세요.

`Principal` 요소에서 수임된 역할 세션 ARN을 지정하려면 다음 형식을 사용하십시오.

```
"Principal": { "AWS": "arn:aws:sts::AWS-account-ID:assumed-role/role-name/role-session-name" }
```

`Principal` 요소에 수임된 역할 세션을 지정할 때 와일드카드 “*”를 사용하여 모든 세션을 의미할 수 없습니다. 보안 주체는 항상 특정 세션의 이름을 지정해야 합니다.

OIDC 세션 보안 주체

OIDC 세션 보안 주체는 AWS STS `AssumeRoleWithWebIdentity` 작업을 사용하는 경우에 발생하는 세션 보안 주체입니다. 외부 OIDC 공급자(IdP)를 사용하여 로그인한 다음 이 작업을 사용하여 IAM 역할을 수행할 수 있습니다. 이렇게 하면 아이덴티티 페더레이션을 활용하고 역할 세션을 발행합니다. 이 작업을 사용하여 역할을 수임할 수 있는 보안 주체에 대한 자세한 내용은 AWS STS API 작업 비교을(를) 참조하세요.

OIDC 공급자로부터 역할을 발행하면 OIDC 공급자에 대한 정보가 포함된 이 특별한 유형의 세션 보안 주체를 얻게 됩니다.

정책에서 이 보안 주체 유형을 사용하여 신뢰할 수 있는 웹 자격 증명 공급자를 기반으로 액세스를 허용하거나 거부합니다. 역할 신뢰 정책의 `Principal` 요소에서 OIDC 역할 세션 ARN을 지정하려면 다음 형식을 사용합니다.

```
"Principal": { "Federated": "cognito-identity.amazonaws.com" }
```

```
"Principal": { "Federated": "www.amazon.com" }
```

```
"Principal": { "Federated": "graph.facebook.com" }
```

```
"Principal": { "Federated": "accounts.google.com" }
```

SAML 세션 보안 주체

SAML 세션 보안 주체는 AWS STS `AssumeRoleWithSAML` 작업을 사용하여 발생하는 세션 보안 주체입니다. 외부 SAML 자격 증명 공급자(IdP)를 사용하여 로그인한 다음 이 작업을 사용하여 IAM 역할을 수행할 수 있습니다. 이렇게 하면 아이덴티티 페더레이션을 활용하고 역할 세션을 발행합니다. 이 작업을 사용하여 역할을 수임할 수 있는 보안 주체에 대한 자세한 내용은 AWS STS API 작업 비교을(를) 참조하세요.

SAML 자격 증명 공급자로부터 역할을 발행하면 SAML 자격 증명 공급자에 대한 정보가 포함된 이 특별한 유형의 세션 보안 주체를 얻게 됩니다.

정책에서 이 보안 주체 유형을 사용하여 신뢰할 수 있는 SAML 자격 증명 공급자를 기반으로 액세스를 허용하거나 거부합니다. `Principal` 요소에서 SAML 아이덴티티 역할 세션 ARN을 지정하려면 다음 형식을 사용하세요.

```
"Principal": { "Federated": "arn:aws:iam::AWS-account-ID:saml-provider/provider-name" }
```

IAM 사용자 보안 주체

리소스 기반 정책의 `Principal` 요소 또는 보안 주체를 지원하는 조건 키에서 IAM 사용자를 지정할 수 있습니다.

> **Note**
>
> `Principal` 요소에서 Amazon 리소스 이름(ARN)의 사용자 이름 부분은 대/소문자를 구분합니다.

```
"Principal": { "AWS": "arn:aws:iam::AWS-account-ID:user/user-name" }
```

```
"Principal": {
  "AWS": [
    "arn:aws:iam::AWS-account-ID:user/user-name-1",
    "arn:aws:iam::AWS-account-ID:user/user-name-2"
  ]
}
```

`Principal` 요소로 사용자를 지정할 때는 "모든 사용자"의 의미로 와일드카드(*)를 사용할 수 없습니다. 보안 주체는 항상 특정 사용자가 되어야 하기 때문입니다.

> **Important**
>
> 역할 신뢰 정책의 `Principal` 요소에 특정 IAM 사용자를 가리키는 ARN이 포함되어 있으면, IAM은 정책을 저장할 때 ARN을 사용자의 고유한 보안 주체 ID로 변환합니다. 그러면 누군가가 해당 사용자를 제거하고 다시 만들어 본인의 권한을 에스컬레이션할 위험을 완화할 수 있습니다. 일반적으로 콘솔에서는 이 ID가 보이지 않습니다. 신뢰 정책이 표시될 때 해당 사용자의 ARN으로 다시 역변환되기 때문입니다. 그러나 해당 사용자를 삭제하면 관계가 깨집니다. 사용자를 다시 생성해도 정책은 더 이상 적용되지 않습니다. 새 사용자의 새 보안 주체 ID가 신뢰 정책에 저장된 ID와 일치하지 않기 때문입니다. 이 경우 AWS가 더 이상 유효한 ARN에 다시 매핑할 수 없기 때문에 보안 주체 ID가 리소스 기반 정책에 나타납니다. 결과적으로 신뢰 정책의 `Principal` 요소에서 참조된 사용자를 삭제하고 다시 만드는 경우, 역할을 편집하여 현재의 잘못된 보안 주체 ID를 올바른 ARN으로 바꿔야 합니다. IAM은 정책을 저장하면 ARN이 다시 해당 사용자의 새로운 보안 주체 ID로 변환됩니다.

IAM Identity Center 보안 주체

IAM Identity Center에서는 리소스 기반 정책의 보안 주체를 AWS 계정 보안 주체로 정의해야 합니다. 액세스를 지정하려면 조건 블록에 있는 권한 세트의 역할 ARN을 참조합니다. 자세한 내용은 IAM Identity Center 사용 설명서에서 리소스 정책, Amazon EKS 및 AWS KMS의 권한 세트 참조를 참조하세요.

AWS STS 페더레이션 사용자 세션 보안 주체

보안 주체를 지원하는 리소스 기반 정책이나 조건 키의 `Principal` 요소에 있는 페더레이션 사용자 세션을 지정할 수 있습니다.

> **Important**
>
> AWS에서는 루트 사용자 액세스 필요와 같이 필요한 경우에만 AWS STS 페더레이션 사용자를 사용하는 것이 좋습니다. 대신, 역할을 사용하여 권한 위임.

AWS STS 페더레이션 사용자 세션 보안 주체는 AWS STS `GetFederationToken` 작업을 사용하여 발생하는 세션 보안 주체입니다. 이 경우, AWS STS에서는 IAM 역할을 사용하는 대신 아이덴티티 페더레이션을 임시 액세스 토큰을 얻는 방법으로 사용합니다.

AWS에서 IAM 사용자 또는 AWS 계정 루트 사용자는 장기 액세스 키를 사용하여 인증할 수 있습니다. 이 작업을 사용하여 페더레이션할 수 있는 보안 주체에 대한 자세한 내용은 AWS STS API 작업 비교을(를) 참조하세요.

- IAM 페더레이션 사용자 - IAM 사용자는 해당 IAM 사용자에 대한 페더레이션 사용자 세션 보안 주체를 생성하는 `GetFederationToken` 작업을 사용하여 페더레이션합니다.
- 페더레이션 루트 사용자 - 루트 사용자는 해당 루트 사용자에 대한 페더레이션 사용자 세션 보안 주체를 생성하는 `GetFederationToken` 작업을 사용하여 페더레이션합니다.

IAM 사용자 또는 루트 사용자가 이 작업을 사용하여 AWS STS에서 임시 자격 증명을 요청하면 임시 페더레이션 사용자 세션이 시작됩니다. 이 세션의 ARN은 페더레이션된 원래 자격 증명을 기반으로 합니다.

`Principal` 요소에서 페더레이션 사용자 세션 ARN을 지정하려면 다음 형식을 사용하십시오.

```
"Principal": { "AWS": "arn:aws:sts::AWS-account-ID:federated-user/user-name" }
```

AWS 서비스 보안 주체

보안 주체를 지원하는 리소스 기반 정책이나 조건 키의 `Principal` 요소에 있는 AWS 서비스를 지정할 수 있습니다. 서비스 보안 주체는 서비스의 식별자입니다.

AWS 서비스에서 위임할 수 있는 IAM 역할을 서비스 역할이라고 합니다. 서비스 역할에는 신뢰 정책이 포함되어 있어야 합니다. 신뢰 정책은 역할을 수임할 수 있는 보안 주체를 정의하는 역할에 연결된 리소스 기반 정책입니다. 일부 서비스 역할에는 신뢰 정책이 미리 정의되어 있습니다. 그러나 신뢰 정책에 서비스 보안 주체를 지정해야 하는 경우도 있습니다. IAM 정책의 서비스 보안 주체는 `"Service": "*"`일 수 없습니다.

> **Important**
>
> 서비스 보안 주체의 식별자에는 서비스 이름이 포함되어 있고 일반적으로 다음과 같은 형식을 갖습니다.
>
> *service-name*.amazonaws.com

서비스 보안 주체는 서비스가 정의합니다. 일부 서비스에 대한 서비스 주체는 AWS IAM으로 작업하는 서비스을(를) 열고 Service-linked role(서비스 연결 역할) 열에서 서비스에 Yes(예)가 선택되어 있는지 확인한 다음 Yes(예) 링크를 열어 해당 서비스에 대한 서비스 연결 역할 문서를 통해 확인할 수 있습니다. 서비스 보안 주체를 보려면 서비스 연결 역할 권한 섹션을 참조하세요.

다음 예제는 서비스 역할에 연결할 수 있는 정책을 보여줍니다. 이 정책은 Amazon ECS 및 Elastic Load Balancing의 두 서비스가 역할을 수임할 수 있도록 합니다. 그러면 서비스가 해당 역할에 할당된 권한 정책에서 부여한 모든 작업을 수행할 수 있습니다(표시되지 않음). 여러 서비스 보안 주체를 지정할 때 `Service` 요소를 두 개 지정하면 안 됩니다. 하나만 지정할 수 있습니다. 대신 여러 서비스 보안 주체의 배열을 하나의 `Service` 요소의 값으로 사용합니다.

```
"Principal": {
    "Service": [
        "ecs.amazonaws.com",
        "elasticloadbalancing.amazonaws.com"
   ]
}
```

옵트인 리전용 AWS 서비스 주체

여러 AWS 리전에서 리소스를 시작할 수 있으며 이러한 리전 중 일부는 반드시 옵트인해야 합니다. 옵트인해야 하는 전체 지역 목록은 AWS 일반 참조 가이드의 AWS 리전 관리를 참조하세요.

옵트인 리전의 AWS 서비스가 동일한 리전 내에서 요청을 보내는 경우 서비스 주체 이름 형식은 해당 서비스 주체 이름의 리전별이 아닌 버전으로 식별됩니다.

service-name.amazonaws.com

옵트인 리전의 AWS 서비스가 다른 리전에 교차 리전 요청을 보내면 서비스 주체 이름 형식이 해당 서비스 주체 이름의 리전별 버전으로 식별됩니다.

service-name.{region}.amazonaws.com

예를 들어 Amazon SNS 주제가 리전 ap-southeast-1에 있고 Amazon S3 버킷이 옵트인 리전 ap-east-1에 있다고 가정해 보겠습니다. SNS 주제에 메시지를 게시하도록 S3 버킷 알림을 구성하려고 합니다. S3 서비스에서 SNS 주제에 메시지를 게시할 수 있도록 하려면 해당 주제의 리소스 기반 액세스 정책을 통해 S3 서비스 주체에게 sns:Publish 권한을 부여해야 합니다.

주제 액세스 정책에서 S3 서비스 주체 s3.amazonaws.com의 리전별이 아닌 버전을 지정하는 경우 버킷에서 주제로의 sns:Publish 요청이 실패합니다. 다음 예제에서는 SNS 주제 액세스 정책의 Principal 정책 요소에 리전별이 아닌 S3 서비스 주체를 지정합니다.

```
"Principal": { "Service": "s3.amazonaws.com" }
```

버킷은 옵트인 리전에 있고 요청은 동일한 리전 외부에서 이루어지므로 S3 서비스 주체는 리전별 서비스 주체 이름 s3.ap-east-1.amazonaws.com으로 표시됩니다. 옵트인 리전의 AWS 서비스가 다른 리전에 요청할 때는 리전별 서비스 주체 이름을 사용해야 합니다. 리전별 서비스 주체 이름을 지정한 후 버킷이 다른 리전에 있는 SNS 주제에 sns:Publish 요청을 하면 요청이 성공합니다. 다음 예제에서는 SNS 주제 액세스 정책의 Principal 정책 요소에 리전별 S3 서비스 주체를 지정합니다.

```
"Principal": { "Service": "s3.ap-east-1.amazonaws.com" }
```

옵트인 리전에서 다른 리전으로의 리전 간 요청에 대한 리소스 정책 또는 서비스 주체 기반 허용 목록은 리전별 서비스 주체 이름을 지정한 경우에만 성공합니다.

> **Note**
>
> IAM 역할 신뢰 정책의 경우 리전별이 아닌 서비스 사용자 이름을 사용하는 것이 좋습니다. IAM 리소스는 글로벌 리소스이므로 모든 리전에서 동일한 역할을 사용할 수 있습니다.

모든 보안 주체

와일드카드(*)를 사용하여 리소스 기반 정책의 Principal 요소 또는 보안 주체를 지원하는 조건 키에 모든 보안 주체를 지정할 수 있습니다. 리소스 기반 정책은 권한을 부여하고 조건 키는 정책 문의 조건을 제한하는 데 사용됩니다.

> **Important**
>
> 퍼블릭 또는 익명 액세스 권한을 부여하려는 경우가 아니면 Allow 효과가 있는 리소스 기반 정책의 Principal 요소에 와일드카드(*)를 사용하지 않는 것이 좋습니다. 그렇지 않으면 Principal 요소에서 의도한 보안 주체, 서비스 또는 AWS 계정을 지정한 다음 Condition 요소에서 액세스를 추가로 제한하세요. 이는 특히 다른 보안 주체가 계정의 보안 주체가 될 수 있도록 허용하기 때문에 IAM 역할 신뢰 정책에 특히 해당됩니다.

리소스 기반 정책의 경우 Allow 효과와 함께 와일드카드(*)를 사용하면 익명 사용자(퍼블릭 액세스)를 포함한 모든 사용자에게 액세스 권한이 부여됩니다. 계정 내의 IAM 사용자 및 역할 보안 주체의 경우 다른 권한이 필요하지 않습니다. 다른 계정의 보안 주체의 경우 계정에 리소스에 액세스할 수 있는 자격 증명 기반 권한도 있어야 합니다. 이를 크로스 계정 액세스라고 합니다.

익명 사용자의 경우 다음 요소는 동일합니다.

```
"Principal": "*"
```

```
"Principal" : { "AWS" : "*" }
```

보안 주체 이름이나 ARN의 일부를 나타내기 위해 와일드카드를 사용할 수 없습니다.

다음 예제는 Deny와 함께 NotPrincipal 지정 대신 Condition 요소에 지정된 보안 주체를 제외한 모든 보안 주체를 명시적으로 거부하는 데 사용할 수 있는 리소스 기반 정책을 보여줍니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UsePrincipalArnInsteadOfNotPrincipalWithDeny",
      "Effect": "Deny",
      "Action": "s3:*",
```

```
      "Principal": "*",
      "Resource": [
        "arn:aws:s3:::BUCKETNAME/*",
        "arn:aws:s3:::BUCKETNAME"
      ],
      "Condition": {
        "ArnNotEquals": {
          "aws:PrincipalArn": "arn:aws:iam::444455556666:user/user-name"
        }
      }
    }
  ]
}
```

추가 정보

자세한 내용은 다음 자료를 참조하십시오.

- Amazon Simple Storage Service 사용 설명서의 버킷 정책 예제
- Amazon Simple Notification Service 개발자 안내서의 Amazon SNS에 대한 정책 예제
- Amazon Simple Queue Service Developer Guide(Amazon Simple Storage Service 개발자 안내서)의 Amazon SQS policy examples(Amazon SQS 정책 예제)
- AWS Key Management Service 개발자 안내서의 키 정책
- AWS 일반 참조의 계정 식별자
- OIDC 페더레이션

AWS JSON 정책 요소: NotPrincipal

`NotPrincipal` 요소를 사용하여 IAM 사용자, 페더레이션 사용자, IAM 역할, AWS 계정, AWS 서비스 또는 `NotPrincipal` 요소에 지정된 기타 보안 주체를 제외한 모든 보안 주체에 대한 액세스를 거부할 수 있습니다.

VPC 엔드포인트를 포함한 일부 AWS 서비스에 대한 리소스 기반 정책에는 ID를 사용할 수 있습니다. 리소스 기반 정책은 리소스에 직접 삽입할 수 있는 정책입니다. IAM ID 기반 정책 또는 IAM 역할 신뢰 정책에는 `NotPrincipal` 요소를 사용할 수 없습니다.

`NotPrincipal`은 `"Effect":"Deny"`와 함께 사용해야 합니다. `"Effect":"Allow"`와 함께 사용은 지원되지 않습니다.

> **Important**
>
> NotPrincipal을 사용해야 하는 시나리오는 거의 없습니다. NotPrincipal을 사용하기 전에 다른 권한 부여 옵션을 살펴보는 것이 좋습니다. NotPrincipal을 사용할 경우 여러 정책 유형의 영향을 해결하기 어려울 수 있습니다. 대신 ARN 조건 연산자와 함께 aws:PrincipalArn 컨텍스트 키를 사용하는 것이 좋습니다. 자세한 내용은 모든 보안 주체 섹션을 참조하세요.

Deny와 함께 NotPrincipal 지정

NotPrincipal과 Deny를 함께 사용할 경우, 거부되지 않은 보안 주체의 계정 ARN도 지정해야 합니다. 지정하지 않으면 정책에서 해당 보안 주체를 포함하는 전체 계정에 대한 액세스가 거부될 수 있습니다. 정책에 포함하는 서비스에 따라 AWS에서 먼저 계정을 검증한 후 사용자를 검증할 수 있습니다. 위임된 역할 사용자(역할을 사용하는 사람)를 평가할 때 AWS는 먼저 계정을 검증한 후 위임된 역할 사용자를 평가합니다. 위임된 역할 사용자는 그 역할을 위임 받을 때 지정된 역할 세션 이름으로 식별할 수 있습니다. 따라서 사용자 계정의 ARN을 명시적으로 포함시키거나, 역할의 ARN과 해당 역할을 포함하는 계정의 ARN을 모두 포함시킬 것을 권장합니다.

> **Important**
>
> 권한 경계 정책이 연결된 IAM 사용자 또는 역할에는 Deny 효과를 포함하는 NotPrincipal 정책 요소가 있는 리소스 기반 정책 문을 사용하지 않도록 합니다. Deny 효과를 포함하는 NotPrincipal 요소는 NotPrincipal 요소에 지정된 값에 관계없이 권한 경계 정책이 연결된 IAM 보안 주체를 항상 거부합니다. 이로 인해 원래 리소스에 액세스할 수 있는 일부 IAM 사용자 또는 역할이 해당 리소스에 더 이상 액세스할 수 없습니다. NotPrincipal 요소 대신 액세스를 제한하기 위해 ArnNotEquals 조건 연산자를 aws:PrincipalArn 컨텍스트 키와 함께 사용하도록 리소스 기반 정책 문을 변경하는 것이 좋습니다. 권한 경계에 대한 자세한 내용은 IAM 엔터티의 권한 범위 섹션을 참조하세요.

> **Note**
>
> 최선의 결과를 위해 정책에 계정의 ARN을 포함시켜야 합니다. 모든 경우에 해당하는 것은 아니지만 일부 서비스에서는 계정 ARN이 필요합니다. 기존 정책은 필요한 ARN 없이 계속 적용되지만 이러한 서비스를 포함하는 새 정책은 계정 ARN을 포함시켜야 합니다. IAM은 이러한 서비스를 추적하지 않으므로 계정 ARN을 항상 포함시키는 것이 좋습니다.

다음 예제들은 같은 정책 설명에 있는 NotPrincipal과 "Effect": "Deny"를 효과적으로 사용하는 방법을 보여줍니다.

Example 동일하거나 다른 계정의 IAM 사용자의 예

다음 예제에서는 AWS 계정 444455556666에서 Bob이라는 이름의 사용자만 제외하고 모든 보안 주체의 리소스 액세스가 명시적으로 거부되었습니다. 모범 사례로서 NotPrincipal 요소는 사용자 Bob과 Bob이 속한 AWS 계정(arn:aws:iam::444455556666:root)의 ARN을 모두 포함한다는 점을 유념하세요. NotPrincipal 요소에 Bob의 ARN만 추가될 경우, 정책의 효과에 따라 사용자 Bob을 포함하는 AWS 계정에 대한 액세스가 명시적으로 거부될 수 있습니다. 경우에 따라, 사용자는 자신의 상위 계정보다 많은 권한을 가질 수 없습니다. 따라서 Bob의 계정에 대한 액세스가 명시적으로 거부되면 Bob은 리소스에도 액세스하지 못할 수도 있습니다.

이 예제는 444455556666가 아닌 같은 또는 다른 AWS 계정의 리소스에 연결된 리소스 기반 정책의 정책 설명에 포함될 때 의도한 대로 작동합니다. 이 예제 자체로는 Bob에게 액세스 권한을 부여하지 않지만 명시적으로 거부된 보안 주체 목록에서 Bob만 제외됩니다. Bob에게 리소스 액세스 권한을 허용하려면 다른 정책 문으로 "Effect": "Allow"를 작성함으로써 액세스를 명시적으로 허용해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Deny",
        "NotPrincipal": {"AWS": [
            "arn:aws:iam::444455556666:user/Bob",
            "arn:aws:iam::444455556666:root"
        ]},
        "Action": "s3:*",
        "Resource": [
            "arn:aws:s3:::BUCKETNAME",
            "arn:aws:s3:::BUCKETNAME/*"
        ]
    }]
}
```

Example 동일하거나 다른 계정의 IAM 역할의 예

다음 예제에서는 AWS 계정 444455556666에서 cross-account-audit-app이라는 이름의 위임된 역할 사용자만 제외하고 모든 보안 주체의 리소스 액세스가 명시적으로 거부되었습니다. 모범 사례로서, NotPrincipal 요소는 수임된 역할 사용자(cross-account-audit-app), 역할(cross-account-read-only-role) 및 역할이 속한 AWS 계정(444455556666)의 ARN을 포함합니다. NotPrincipal 요소에 역할의 ARN이 누락될 경우 정책 효과에 따라 역할에 대한 액세스가 명시적으로 거부될 수 있습니다. 마찬가

지로, NotPrincipal 요소에 역할이 속한 AWS 계정의 ARN이 누락될 경우에는 정책의 효과에 따라 AWS 계정과 그 계정의 모든 엔터티에 대한 액세스가 명시적으로 거부될 수 있습니다. 경우에 따라, 위임된 역할 사용자는 자신의 상위 역할보다 많은 권한을 가질 수 없고, 역할은 자신의 상위 AWS 계정보다 많은 권한을 가질 수 없습니다. 따라서 역할 또는 계정에 대한 액세스가 명시적으로 거부되면 위임된 역할 사용자는 리소스에 액세스하지 못할 수 있습니다.

이 예제는 444455556666 이 아닌 다른 AWS 계정의 리소스에 연결된 정책 문이 리소스 기반 정책의 정책 문에 포함되기 때문에 의도한 대로 효과가 나타납니다. 이 예제 자체에서는 위임된 역할 사용자인 cross-account-audit-app에게 액세스를 허용하지 않지만 명시적으로 거부된 보안 주체 목록에서 cross-account-audit-app만 제외됩니다. cross-account-audit-app에게 리소스 액세스 권한을 부여하려면 다른 정책 문으로 "Effect": "Allow"를 작성함으로써 액세스를 명시적으로 허용해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Deny",
        "NotPrincipal": {"AWS": [
            "arn:aws:sts::444455556666:assumed-role/cross-account-read-only-role/cross-
account-audit-app",
            "arn:aws:iam::444455556666:role/cross-account-read-only-role",
            "arn:aws:iam::444455556666:root"
        ]},
        "Action": "s3:*",
        "Resource": [
            "arn:aws:s3:::Bucket_AccountAudit",
            "arn:aws:s3:::Bucket_AccountAudit/*"
        ]
    }]
}
```

NotPrincipal 요소에 위임된 역할 세션을 지정할 때 와일드카드(*)를 사용하여 “모든 세션”을 의미할 수 없습니다. 보안 주체는 항상 특정 세션의 이름을 지정해야 합니다.

IAM JSON 정책 요소: Action

Action 요소는 특정 작업의 허용 또는 거부 여부를 지정합니다. 문에는 Action 또는 NotAction 요소가 반드시 추가되어야 합니다. AWS 서비스마다 실행할 수 있는 작업을 설명한 목록이 있습니다. 예를 들어 Amazon S3 작업 목록은 Amazon Simple Storage Service 사용 설명서의 정책에 권한 지정에서, Amazon EC2 작업 목록은 Amazon EC2 API 참조에서, 그리고 AWS Identity and Access Management 작업 목록은 IAM API 참조에서 확인할 수 있습니다. 그 밖에 다른 서비스의 작업 목록은 해당 서비스의 API 참조 설명서를 참조하세요.

서비스 네임스페이스를 작업 접두사(iam, ec2, sqs, sns, s3 등)로 사용하고 허용 또는 거부할 작업 이름을 사용하여 값을 지정합니다. 이름은 서비스에서 지원되는 작업과 일치해야 합니다. 접두사와 작업 이름은 대/소문자를 구분하지 않습니다. 예를 들어 iam:ListAccessKeys는 IAM:listaccesskeys와 동일합니다. 다음은 각 서비스의 Action 요소를 나타낸 예제입니다.

Amazon SQS 작업

```
"Action": "sqs:SendMessage"
```

Amazon EC2 작업

```
"Action": "ec2:StartInstances"
```

IAM 작업

```
"Action": "iam:ChangePassword"
```

Amazon S3 작업

```
"Action": "s3:GetObject"
```

Action 요소는 다수의 값을 지정할 수도 있습니다.

```
"Action": [ "sqs:SendMessage", "sqs:ReceiveMessage", "ec2:StartInstances",
 "iam:ChangePassword", "s3:GetObject" ]
```

특정 AWS 제품이 제공하는 모든 작업에 대해 액세스 권한을 부여하려면 와일드카드(*)를 사용하면 됩니다. 예를 들어, 다음 Action 요소는 모든 S3 작업에 적용됩니다.

```
"Action": "s3:*"
```

와일드카드(*)는 작업 이름에도 사용할 수 있습니다. 예를 들어 다음 Action 요소는 AccessKey, CreateAccessKey, DeleteAccessKey, ListAccessKeys 등 문자열 UpdateAccessKey를 포함하는 IAM 작업 모두에게 적용됩니다.

```
"Action": "iam:*AccessKey*"
```

일부 서비스에서는 사용 가능한 작업을 제한할 수도 있습니다. 예를 들어 Amazon SQS에서는 가능한 모든 Amazon SQS 작업의 하위 집합만 사용할 수 있습니다. 이 경우 와일드카드(*)는 대기열 전체

를 제어하지 못하고, 공유한 작업의 하위 집합만 제어가 가능합니다. 자세한 내용은 Amazon Simple Queue Service 개발자 안내서의 권한 이해를 참조하세요.

IAM JSON 정책 요소: NotAction

NotAction은 지정된 작업의 목록을 제외한 모든 작업과 명시적으로 일치하는 고급 정책 요소입니다. NotAction을 사용하면 일치하는 작업의 긴 목록을 포함하는 대신 일치하지 않는 몇몇 작업만 나열함으로써 정책을 줄일 수 있습니다. NotAction에 지정된 작업은 정책 설명의 Allow 또는 Deny 효과의 영향을 받지 않습니다. 따라서 나열되지 않은 모든 해당 작업 또는 서비스가 Allow 효과를 사용할 경우 허용되고, Deny를 사용하려는 경우에는 나열되지 않은 작업 또는 서비스가 거부됩니다. NotAction을 Resource 요소와 함께 사용할 경우 정책 범위를 제공해야 합니다. 이에 따라 AWS는 어떤 작업이나 서비스를 적용할 수 있는지 결정합니다. 자세한 내용은 다음 예제 정책을 참조하세요.

NotAction 및 Allow

설명문에서 NotAction 요소를 "Effect": "Allow"와 함께 사용하여 AWS 서비스에서 NotAction에 지정된 작업을 제외한 모든 작업에 대한 액세스 권한을 제공할 수 있습니다. 이 요소와 Resource 요소를 함께 사용하여 정책에 대한 범위를 제공하고 지정한 리소스에서 수행할 수 있는 작업으로만 작업을 제한할 수 있습니다.

다음 예제에서는 버킷 삭제를 제외하고 사용자가 S3 리소스에서 수행할 수 있는 모든 Amazon S3 작업에 액세스할 수 있도록 허용합니다. ListAllMyBuckets S3 API 작업은 "*" 리소스가 필요하기 때문에 이 작업은 사용자가 사용할 수 없습니다. 이 정책은 또한 다른 서비스에서의 작업을 허용하지 않습니다. 다른 서비스 작업은 S3 리소스에 적용되지 않기 때문입니다.

```
"Effect": "Allow",
"NotAction": "s3:DeleteBucket",
"Resource": "arn:aws:s3:::*",
```

때로는 다수의 작업에 액세스하도록 허용해야 할 수 있습니다. NotAction 요소를 사용하여 효과적으로 설명문을 반전시켜 작업 목록을 단축시킬 수 있습니다. 예를 들어 AWS 서비스는 종류가 다양하므로 사용자에게 IAM 작업에 대한 액세스를 제외한 모든 것을 허용하는 정책을 생성하기를 원할 수 있습니다.

다음 예제는 사용자가 IAM을 제외한 모든 AWS 서비스에서 모든 작업에 액세스하도록 허용합니다.

```
"Effect": "Allow",
"NotAction": "iam:*",
"Resource": "*"
```

동일한 설명문에서 또는 동일한 정책의 다른 설명문에서 NotAction 요소와 "Effect": "Allow"를 사용할 경우 주의하세요. NotAction은 명시적으로 나열되지 않거나 특정 리소스에 적용되지 않는 모든 서비스 및 작업과 일치하므로 사용자에게 의도한 것보다 많은 권한을 부여하는 결과를 가져올 수 있습니다.

NotAction 및 Deny

설명문에서 NotAction 요소를 "Effect": "Deny"와 함께 사용하여 NotAction 요소에 지정된 작업을 제외하고 모든 나열된 리소스에 대한 액세스를 거부할 수 있습니다. 이 조합은 나열된 항목을 허용하는 것이 아니라 나열되지 않은 작업을 명시적으로 거부합니다. 그러므로 허용하려는 작업은 별도로 허용해야 합니다.

다음의 조건부 예제는 사용자가 MFA를 사용하여 로그인하지 않은 경우 비 IAM 작업에 대한 액세스를 거부합니다. 사용자가 MFA를 사용하여 로그인한 경우에는 "Condition" 테스트에 실패하며 최종 "Deny" 문은 효과가 없습니다. 단, 이 정책은 사용자에게 작업에 대한 액세스 권한을 부여하는 것이 아니라 IAM 작업을 제외한 다른 모든 작업을 명시적으로 거부할 뿐입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "DenyAllUsersNotUsingMFA",
        "Effect": "Deny",
        "NotAction": "iam:*",
        "Resource": "*",
        "Condition": {"BoolIfExists": {"aws:MultiFactorAuthPresent": "false"}}
    }]
}
```

특정 서비스의 작업을 제외하고 특정 리전 외부의 작업에 대한 액세스를 거부하는 정책의 예는 AWS: 요청된 리전에 따라 AWS에 대한 액세스를 거부 섹션을 참조하세요.

IAM JSON 정책 요소: Resource

IAM 정책 문에서 Resource 요소는 문의 적용 대상인 객체를 정의합니다. 문장에는 Resource또는 NotResource요소가 반드시 추가되어야 합니다.

Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. ARN의 형식은 참조하는 AWS 서비스 및 참조하는 특정 리소스에 따라 다릅니다. ARN 형식은 다양하지만 항상 ARN을 사용하여 리소스를 식별합니다. ARL의 형식에 대한 자세한 내용은 IAM ARN 섹션을 참조하세요. 리소스 지정 방법에 대한 자세한 내용은 문을 작성하려는 리소스의 서비스 설명서를 참조하세요.

> **Note**
>
> 일부 AWS 서비스에서는 개별 리소스에 대한 작업 지정을 허용하지 않습니다. 이러한 경우 `Action` 또는 `NotAction` 요소에 나열한 모든 작업이 해당 서비스의 모든 리소스에 적용됩니다. 이 경우 `Resource` 요소에 와일드카드 문자(*)를 사용합니다.

다음은 특정 Amazon SQS 대기열을 나타낸 예시입니다.

```
"Resource": "arn:aws:sqs:us-east-2:account-ID-without-hyphens:queue1"
```

다음은 AWS 계정에서 Bob이라는 이름의 IAM 사용자를 나타내는 예제입니다.

> **Note**
>
> `Resource` 요소에서 IAM 사용자 이름은 대/소문자를 구분합니다.

```
"Resource": "arn:aws:iam::account-ID-without-hyphens:user/Bob"
```

리소스 ARN에서 와일드카드 사용

개별 ARN 세그먼트(콜론으로 구분된 부분) 내에서 와일드카드 문자(* 및 ?)를 사용하여 다음을 표현할 수 있습니다.

- 모든 문자 조합(*)
- 모든 단일 문자(?)

각 세그먼트에 여러 개의 * 또는 ? 문자를 사용할 수 있습니다. * 와일드카드가 리소스 ARN 세그먼트의 마지막 문자인 경우 콜론 경계를 넘어서 일치하도록 확장할 수 있습니다. 콜론으로 구분된 ARN 세그먼트 내에서 와일드카드(* 및 ?)를 사용하는 것이 좋습니다.

> **Note**
>
> AWS 제품을 식별하는 서비스 세그먼트에서는 와일드카드 문자를 사용할 수 없습니다. ARN 세그먼트에 대한 자세한 내용은 Amazon 리소스 이름(ARN)으로 AWS 리소스를 식별합니다. 섹션을 참조하세요.

다음은 경로가 /accounting인 IAM 사용자를 모두 나타낸 예시입니다.

```
"Resource": "arn:aws:iam::account-ID-without-hyphens:user/accounting/*"
```

다음은 특정 Amazon S3 버킷 내에 포함된 모든 항목을 나타낸 예시입니다.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
```

별표(*) 문자는 확장하여 특정 서비스 네임스페이스 내에서 구분 기호로 나타날 수 있는 슬래시(/)와 같은 문자를 포함하여 세그먼트 내의 모든 항목을 바꿀 수 있습니다. 예를 들어, 모든 서비스에 동일한 와일드카드 확장 논리가 적용되는 다음 Amazon S3 ARN을 생각해 봅시다.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*/test/*"
```

ARN의 와일드카드는 나열된 첫 번째 객체뿐만 아니라 버킷의 다음 모든 객체에 적용됩니다.

```
amzn-s3-demo-bucket/1/test/object.jpg
amzn-s3-demo-bucket/1/2/test/object.jpg
amzn-s3-demo-bucket/1/2/test/3/object.jpg
amzn-s3-demo-bucket/1/2/3/test/4/object.jpg
amzn-s3-demo-bucket/1///test///object.jpg
amzn-s3-demo-bucket/1/test/.jpg
amzn-s3-demo-bucket//test/object.jpg
amzn-s3-demo-bucket/1/test/
```

이전 목록의 마지막 두 객체를 고려하세요. Amazon S3 객체 이름은 일반적인 구분 기호 슬래시(/) 문자로 시작하거나 끝낼 수 있습니다. / 문자는 구분 기호로 작동하지만 리소스 ARN 내에서 이 문자를 사용할 때는 특별한 의미가 없습니다. 다른 유효한 문자와 동일하게 처리됩니다. ARN이 다음 객체와 일치하지 않습니다.

```
amzn-s3-demo-bucket/1-test/object.jpg
amzn-s3-demo-bucket/test/object.jpg
amzn-s3-demo-bucket/1/2/test.jpg
```

여러 리소스 지정

ARN 배열을 사용하여 Resource 요소에 여러 리소스를 지정할 수 있습니다. 다음은 DynamoDB 테이블을 두 개 나타낸 예시입니다.

```
"Resource": [
    "arn:aws:dynamodb:us-east-2:account-ID-without-hyphens:table/books_table",
    "arn:aws:dynamodb:us-east-2:account-ID-without-hyphens:table/magazines_table"
]
```

리소스 ARN에서 정책 변수 사용

Resource 요소에서 ARN의 부분에 JSON 정책 변수를 사용하여 특정 리소스를 식별할 수 있습니다(ARN의 끝 부분에 사용). 예를 들어 {aws:username} 키를 리소스 ARN에 사용하여 현재 사용자의 이름을 리소스 이름에 추가해야 한다는 것을 나타낼 수 있습니다. 다음은 {aws:username} 요소에서 Resource 키를 사용하는 방법을 나타낸 예시입니다. 이 정책에서는 현재 사용자 이름과 일치하는 Amazon DynamoDB 테이블에 대한 액세스가 허용됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "dynamodb:*",
    "Resource": "arn:aws:dynamodb:us-east-2:account-id:table/${aws:username}"
  }
}
```

JSON 정책 변수에 대한 자세한 내용은 IAM 정책 요소: 변수 및 태그 섹션을 참조하세요.

IAM JSON 정책 요소: NotResource

NotResource는 지정된 리소스를 제외한 모든 리소스와 명시적으로 일치하는 고급 정책 요소입니다. NotResource를 사용하면 일치하는 리소스의 긴 목록을 포함하는 대신 일치하지 않는 몇몇 리소스만 나열함으로써 정책을 줄일 수 있습니다. 이는 단일 AWS 서비스 내에서 적용되는 정책에 특히 유용합니다.

예를 들어 HRPayroll라는 이름의 그룹이 있다고 가정하겠습니다. 그리고 HRPayroll 멤버는 HRBucket 버킷의 Payroll 폴더를 제외하고 모든 Amazon S3 리소스에 액세스할 수 없습니다. 다음 정책은 나열된 리소스 이외의 모든 Amazon S3 리소스에 대한 액세스를 거부합니다. 단, 이 정책은 사용자에게 리소스에 대한 액세스 권한을 부여하는 것이 아닙니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
```

```
    "Effect": "Deny",
    "Action": "s3:*",
    "NotResource": [
      "arn:aws:s3:::HRBucket/Payroll",
      "arn:aws:s3:::HRBucket/Payroll/*"
    ]
  }
}
```

일반적으로 리소스에 대한 액세스를 명시적으로 거부하려면 `"Effect":"Deny"`를 사용하고 각 폴더를 개별적으로 나열하는 `Resource` 요소를 포함하는 정책을 작성합니다. 하지만 이때 사용자가 HRBucket에 폴더를 추가하거나 액세스하면 안 되는 Amazon S3에 리소스를 추가할 때마다 그 이름 역시 `Resource` 목록에 추가해야 합니다. 그렇지 않고 `NotResource` 요소를 사용할 때는 폴더 이름을 `NotResource` 요소에 추가하지 않더라도 사용자가 새 폴더에 대한 액세스 권한이 자동으로 거부됩니다.

`NotResource`를 사용할 때는 이 요소에 지정된 리소스가 제한되지 않는 유일한 리소스라는 점을 명심해야 합니다. 이렇게 하면 작업에 적용되는 모든 리소스가 제한됩니다. 위의 예에서 정책은 Amazon S3 작업에만 적용되므로 Amazon S3 리소스에만 영향을 미칩니다. 작업에도 Amazon EC2 작업이 포함되어 있으면 정책에서 EC2 리소스에 대한 액세스를 거부하지 않습니다. 리소스의 ARN을 지정할 수 있는 서비스 작업에 대해 알아보려면 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

다른 요소가 있는 NotResource

`"Effect": "Allow"`, `"Action": "*"` 및 `"NotResource": "arn:aws:s3:::HRBucket"` 요소를 함께 사용하면 안 됩니다. 이 문은 HRBucket S3 버킷을 제외한 모든 리소스에 대해 AWS의 모든 작업을 허용하므로 매우 위험합니다. 이렇게 하면 HRBucket에 액세스할 수 있는 정책을 사용자가 직접 추가할 수 있으므로 주의해야 합니다.

동일한 설명문에서 또는 동일한 정책의 다른 설명문에서 `NotResource` 요소와 `"Effect": "Allow"`를 사용할 경우 주의하세요. `NotResource`은 명시적으로 나열되지 않은 모든 서비스 및 작업을 허용하므로 사용자에게 의도한 것보다 많은 권한을 부여하는 결과를 가져올 수 있습니다. 동일한 설명문에서 `NotResource` 요소와 `"Effect": "Deny"`를 사용하면 명시적으로 나열되지 않은 서비스 및 리소스를 거부합니다.

IAM JSON 정책 요소: Condition

`Condition` 요소(또는 `Condition` 블록)를 사용하여 정책의 효력이 발생하는 시점에 대한 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. `Condition` 요소에서 조건 연산자(같음, 보다 작

음 등)를 사용하여 정책의 컨텍스트 키 및 값을 요청 컨텍스트의 키 및 값과 일치시키는 표현식을 작성합니다. 요청 컨텍스트에 대한 자세한 내용은 요청 구성 요소 섹션을 참조하세요.

```
"Condition" : { "{condition-operator}" : { "{condition-key}" : "{condition-value}" }}
```

정책 조건에서 지정하는 컨텍스트 키는 전역 조건 컨텍스트 키 또는 서비스별 컨텍스트 키일 수 있습니다. 전역 조건 컨텍스트 키에는 aws: 접두사가 있습니다. 서비스별 컨텍스트 키에는 서비스 접두사가 있습니다. 예를 들어, Amazon EC2를 사용하면 ec2:InstanceType 컨텍스트 키를 사용하여 해당 서비스에 고유한 조건을 작성할 수 있습니다. iam: 접두사가 있는 서비스별 IAM 컨텍스트 키를 보려면 IAM 및 AWS STS 조건 컨텍스트 키 섹션을 참조하세요.

컨텍스트 키 이름은 대/소문자를 구분하지 않습니다. 예를 들어 aws:SourceIP 컨텍스트 키를 포함시키는 것은 AWS:SourceIp에 대한 테스트와 동일합니다. 컨텍스트 키 값의 대/소문자 구분은 사용하는 조건 연산자에 따라 다릅니다. 예를 들어 다음 조건에는 johndoe의 요청만 일치하도록 하는 StringEquals 연산자가 포함됩니다. 이름이 JohnDoe인 사용자는 액세스가 거부됩니다.

```
"Condition" : { "StringEquals" : { "aws:username" : "johndoe" }}
```

다음 조건은 StringEqualsIgnoreCase 연산자를 사용하여 이름이 johndoe 또는 JohnDoe인 사용자와 일치합니다.

```
"Condition" : { "StringEqualsIgnoreCase" : { "aws:username" : "johndoe" }}
```

일부 컨텍스트 키는 키 이름의 특정 부분을 지정하도록 허용하는 키 값 페어를 지원합니다. aws:RequestTag/*tag-key* 컨텍스트 키, AWS KMS kms:EncryptionContext:*encryption_context_key* 및 여러 서비스에서 지원하는 ResourceTag/*tag-key* 컨텍스트 키가 그 예입니다.

- Amazon EC2와 같은 서비스에 대해 ResourceTag/*tag-key* 컨텍스트 키를 사용하는 경우 tag-key에 대한 키 이름을 지정해야 합니다.
- 키 이름은 대/소문자를 구분하지 않습니다. 따라서 정책의 조건 요소에서 "aws:ResourceTag/TagKey1": "Value1" 지정을 완료한 경우 조건은 이름이 TagKey1 또는 tagkey1인 리소스 태그 키와 일치하지만 두 가지 모두와 일치하지는 않습니다.
- 이러한 속성을 지원하는 AWS 서비스를 사용하면 대소문자만 다른 여러 키 이름을 생성할 수 있습니다. 예를 들어, ec2=test1와(과) EC2=test2을(를) 사용해 Amazon EC2 인스턴스에 태그를 지정할 수 있습니다. "aws:ResourceTag/EC2": "test1" 같은 조건을 사용하여 리소스에 대한 액세스를 허용하는 경우 키 이름은 두 태그 모두와 일치하지만, 하나의 값만 일치합니다. 이로 인해 예기치 않은 조건 실패가 발생할 수 있습니다.

> **Important**
>
> 모범 사례로서 키 값 페어 속성 이름을 지정할 때 계정의 멤버가 일관적인 명명 규칙을 따르도록 해야 합니다. 예를 들어 태그 또는 AWS KMS 암호화 컨텍스트가 여기에 해당합니다. 태그 지정에 대해 `aws:TagKeys` 컨텍스트 키를 사용하거나 AWS KMS 암호화 컨텍스트에 대해 `kms:EncryptionContextKeys` 사용을 통해 이를 적용할 수 있습니다.

- 모든 조건 연산자의 목록과 각 연산자의 작동 방식에 대한 설명을 보려면 조건 연산자를 참조하세요.
- 달리 지정하지 않는 경우 모든 컨텍스트 키는 다수의 값을 가질 수 있습니다. 다수의 값을 가진 컨텍스트 키를 취급하는 방법에 대한 설명은 다중 값 컨텍스트 키 섹션을 참조하세요.
- 전역에서 사용 가능한 모든 컨텍스트 키의 목록은 AWS 글로벌 조건 컨텍스트 키 섹션을 참조하세요.
- 각 서비스에서 정의하는 조건 컨텍스트 키는 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

요청 컨텍스트

보안 주체가 AWS에 요청하면 AWS는 요청 정보를 요청 컨텍스트로 수집합니다. 이 정보는 요청을 평가하고 승인하는 데 사용됩니다. JSON 정책의 `Condition` 요소를 사용하여 요청 컨텍스트에 대해 특정 컨텍스트 키를 테스트할 수 있습니다. 예를 들어, 사용자가 특정 날짜 범위 중에만 작업을 수행할 수 있도록 aws:CurrentTime 컨텍스트 키를 사용하는 정책을 생성할 수 있습니다.

요청이 제출되면 AWS는 정책의 각 컨텍스트 키를 평가하여 true, false, not present, 때로는 null(빈 데이터 문자열) 값을 반환합니다. 요청에 없는 컨텍스트 키는 불일치로 간주됩니다. 예를 들어, 다음 정책에서는 지난 1시간(3,600초) 동안 다중 인증(MFA)을 사용하여 로그인한 경우에만 자체 MFA 디바이스를 제거할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "AllowRemoveMfaOnlyIfRecentMfa",
        "Effect": "Allow",
        "Action": [
            "iam:DeactivateMFADevice"
        ],
        "Resource": "arn:aws:iam::*:user/${aws:username}",
        "Condition": {
            "NumericLessThanEquals": {"aws:MultiFactorAuthAge": "3600"}
```

```
        }
    }
}
```

요청 컨텍스트는 다음 값을 반환할 수 있습니다.

- True - 요청자가 지난 1시간 이내에 MFA를 사용하여 로그인한 경우 조건은 true를 반환합니다.
- False - 요청자가 1시간 이전에 MFA를 사용하여 로그인한 경우 조건은 false를 반환합니다.
- Not present - 요청자가 AWS CLI 또는 AWS API에서 IAM 사용자 액세스 키를 사용하여 요청한 경우 키가 존재하지 않습니다. 이 경우 키가 존재하지 않으므로 일치하지 않습니다.
- Null - 요청에 태그를 전달하는 등 사용자가 정의한 컨텍스트 키의 경우 빈 문자열을 포함할 수 있습니다. 이 경우 요청 컨텍스트의 값은 null입니다. 경우에 따라 null 값이 true를 반환할 수 있습니다. 예를 들어, `aws:TagKeys` 컨텍스트 키와 함께 다중 값 `ForAllValues` 조건 연산자를 사용하는 경우 요청 컨텍스트가 null을 반환하면 예기치 않은 결과가 발생할 수 있습니다. 자세한 내용은 aws:TagKeys 및 다중 값 컨텍스트 키 섹션을 참조하세요.

조건 블록

다음은 `Condition` 요소의 기본 형식을 나타낸 예제입니다.

```
"Condition": {"StringLike": {"s3:prefix": ["janedoe/*"]}}
```

요청 값은 컨텍스트 키로 표현되며, 여기에서는 `s3:prefix`가 요청 값에 해당합니다. 컨텍스트 키 값은 `janedoe/*`와(과) 같이 리터럴 값으로 지정하는 값과 비교됩니다. 비교 유형은 조건 연산자에서 지정합니다(여기서는 `StringLike`). equals, greater than 및 less than과 같은 일반적인 부울 비교를 사용하여 문자열, 날짜, 숫자 등을 비교하는 조건을 만들 수 있습니다. 문자열 연산자 또는 ARN 연산자를 사용하는 경우 컨텍스트 키 값에 정책 변수를 사용할 수도 있습니다. 다음 예에는 `aws:username` 변수가 포함되어 있습니다.

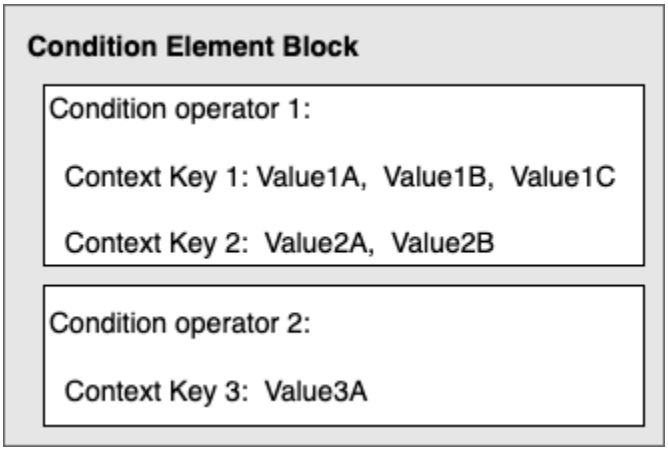
```
"Condition": {"StringLike": {"s3:prefix": ["${aws:username}/*"]}}
```

컨텍스트 키에 다수의 값을 추가할 수 있는 경우도 있습니다. 예를 들어 Amazon DynamoDB에 대한 요청에서는 다수의 테이블 속성 반환이나 업데이트를 요청할 수 있습니다. DynamoDB 테이블에 대한 액세스 정책에 따르면 `dynamodb:Attributes` 컨텍스트 키를 추가하여 요청 시 나열되는 모든 속성 저장이 가능합니다. `Condition` 요소의 설정 연산자를 사용하여 정책에 허용된 속성 목록과 요청에 포함된 속성 여러 가지를 비교함으로써 테스트할 수 있습니다. 자세한 내용은 다중 값 컨텍스트 키 단원을 참조하십시오.

요청 단계에서 정책을 평가할 때는 AWS가 키를 해당하는 요청 값으로 변환합니다. (이 예제에서는 AWS가 요청 날짜와 시간을 사용합니다). 조건 평가에 따라 true 또는 false가 반환되고, 이후 이 조건 평가 결과를 고려하여 정책 전반적인 요청 허용 또는 거부 여부를 결정합니다.

다수의 조건 값

`Condition` 요소에는 여러 조건 연산자를 추가할 수 있으며, 각 조건 연산자마다 다수의 컨텍스트 키-값 페어가 포함될 수 있습니다. 다음은 이것을 설명한 그림입니다.



자세한 내용은 다중 값 컨텍스트 키 단원을 참조하십시오.

IAM JSON 정책 요소: 조건 연산자

`Condition` 요소의 조건 연산자를 사용하여 정책의 조건 키 및 값을 요청 컨텍스트의 값과 일치시킵니다. `Condition` 요소에 대한 자세한 내용은 IAM JSON 정책 요소: Condition 섹션을 참조하세요.

정책에서 사용할 수 있는 조건 연산자는 선택한 조건 키에 따라 다릅니다. 전역 조건 키 또는 서비스별 조건 키를 선택할 수 있습니다. 전역 조건 키에 사용할 수 있는 조건 연산자를 알아보려면 AWS 글로벌 조건 컨텍스트 키 섹션을 참조하세요. 서비스별 조건 키에 사용할 수 있는 조건 연산자를 알아보려면 AWS 서비스에 대한 작업, 리소스 및 조건 키를 참조하고 보려는 서비스를 선택합니다.

> **Important**
>
> 정책 조건에서 지정한 키가 요청 컨텍스트에 없으면 값이 일치하지 않으며 조건은 false입니다. 정책 조건에 따라 키가 일치하지 않아야 하며(예: `StringNotLike` 또는 `ArnNotLike`) 올바른 키가 존재하지 않는 경우 조건은 true입니다. 이 로직은 ...IfExists 및 Null check를 제외한 모든 조건 연산자에 적용됩니다. 이 연산자는 키가 요청 컨텍스트에 존재하는지 여부를 테스트합니다.

조건 연산자는 다음 범주로 그룹화할 수 있습니다.

- 문자열
- 숫자
- 날짜 및 시간
- 부울
- 이진
- IP 주소
- Amazon 리소스 이름(ARN)(일부 서비스에서만 사용 가능.)
- ...IfExists(키 값이 다른 확인을 위해 존재하는지 여부를 확인)
- Null 확인(키 값이 단독 확인을 위해 존재하는지 여부를 확인)

문자열 조건 연산자

문자열 조건 연산자를 사용하여 키와 문자열 값을 비교한 결과에 따라 액세스를 제한하는 Condition 요소를 생성할 수 있습니다.

조건 연산자	설명
StringEquals	정확한 일치, 대소문자 구분
StringNotEquals	불일치
StringEqualsIgnoreCase	정확한 일치, 대소문자 무시
StringNotEqualsIgnoreCase	불일치, 대소문자 무시
StringLike	대소문자 구분 일치. 문자열 어디에서나 다중 문자 일치 와일드카드(*) 및 단일 문자 일치 와일드카드(?)를 값에 포함할 수 있습니다. 부분적으로 문자열이 일치하도록 하려면 와일드카드를 지정해야 합니다.

조건 연산자	설명
	Note 키에 다수의 값이 저장되는 경우에는 설정 연산자(StringLike , ForAllValues:StringLike)를 사용해 ForAnyValue:StringLike 을(를) 한정할 수 있습니다. 자세한 내용은 다중 값 컨텍스트 키 단원을 참조하십시오.
StringNotLike	대소문자 구분 불일치. 문자열 어디에서나 다중 문자 일치 와일드카드(*) 또는 단일 문자 일치 와일드카드(?)를 값에 포함할 수 있습니다.

예를 들어, 다음 문에는 aws:PrincipalTag 키를 사용하여 요청을 수행하는 보안 주체에게 iamuser-admin 작업 범주에서 태그를 지정하도록 지정하는 Condition 요소가 포함되어 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::account-id:user/*",
    "Condition": {"StringEquals": {"aws:PrincipalTag/job-category": "iamuser-admin"}}
  }
}
```

정책 조건에서 지정한 키가 요청 컨텍스트에 없으면 값이 일치하지 않습니다. 이 예제에서는 보안 주체가 태그가 연결된 IAM 사용자를 사용하는 경우 aws:PrincipalTag/job-category 키가 요청 컨텍스트에 존재합니다. 이는 태그 또는 세션 태그가 연결된 IAM 역할을 사용하는 보안 주체를 위해 포함된 것이기도 합니다. 태그가 없는 사용자가 액세스 키를 보거나 편집하려고 하면 조건이 false을 반환하고 요청이 이 문에 의해 묵시적으로 거부됩니다.

정책 변수를 String 조건 연산자와 함께 사용할 수 있습니다.

다음 예제에서는 StringLike 조건 연산자를 사용하여 정책 변수와 문자열 일치를 수행해 IAM 사용자가 Amazon S3 콘솔을 사용하여 Amazon S3 버킷에서 자신의 '홈 디렉토리'를 관리할 수 있는 정책

을 생성할 수 있습니다. 이 정책은 s3:prefix가 지정된 패턴 중 하나와 일치하는 경우 S3 버킷에서 지정된 작업을 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::BUCKET-NAME",
      "Condition": {"StringLike": {"s3:prefix": [
        "",
        "home/",
        "home/${aws:username}/"
      ]}}
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}",
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}/*"
      ]
    }
  ]
}
```

OIDC 페더레이션 시 Condition 요소를 사용하여 애플리케이션 ID와 사용자 ID에 따라 리소스 액세스를 제한하는 방법을 보여주는 정책의 예는 Amazon S3: Amazon Cognito 사용자가 버킷에 있는 객체에 액세스하도록 허용 섹션을 참조하세요.

와일드카드 매칭

문자열 조건 연산자는 사전 정의된 형식을 적용하지 않는 패턴 없는 매칭을 수행합니다. ARN 및 날짜 조건 연산자는 조건 키 값에 구조를 적용하는 문자열 연산자의 하위 집합입니다. ARN 또는 날짜의 부

분 문자열 매칭에 StringLike 또는 StringNotLike 연산자를 사용하면, 매칭 시에 구조의 어느 부분이 와일드카드인지는 무시됩니다.

예를 들어 다음 조건은 여러 조건 연산자를 사용하여 ARN의 부분 매칭을 검색합니다.

ArnLike를 사용하는 경우 ARN의 파티션, 서비스, 계정 ID, 리소스 유형 및 리소스 ID 부분이 요청 컨텍스트의 ARN과 정확히 일치해야 합니다. 리전 및 리소스 경로만 부분 매칭이 허용됩니다.

```
"Condition": {"ArnLike": {"aws:SourceArn": "arn:aws:cloudtrail:*:111122223333:trail/
*"}}
```

ArnLike 대신 StringLike를 사용하는 경우 매칭에서는 ARN 구조를 무시하고 와일드카드가 적용된 부분에 관계없이 부분 일치를 허용합니다.

```
"Condition": {"StringLike": {"aws:SourceArn": "arn:aws:cloudtrail:*:111122223333:trail/
*"}}
```

ARN	ArnLike	StringLike
arn:aws:cloudtrail:us-west-2:111122223333:trail/finance	Match	Match
arn:aws:cloudtrail:us-east-2:111122223333:trail/finance/archive	Match	Match
arn:aws:cloudtrail:us-east-2:444455556666:user/111122223333:trail/finance	일치하는 항목 없음	Match

숫자 조건 연산자

숫자 조건 연산자를 사용하여 키와 정수 또는 십진수 값을 비교한 결과에 따라 액세스를 제한하는 `Condition` 요소를 생성할 수 있습니다.

조건 연산자	설명
NumericEquals	일치
NumericNotEquals	불일치
NumericLessThan	"미만" 일치
NumericLessThanEqu als	"이하" 일치
NumericGreaterThan	"초과" 일치
NumericGreaterThan Equals	"이상" 일치

예를 들어, 다음 문에는 NumericLessThanEquals 조건 연산자에 s3:max-keys 키를 사용하여 요청자가 example_bucket에서 한 번에 최대 10개까지 객체를 나열할 수 있다고 지정하는 Condition 요소가 포함되어 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket",
    "Condition": {"NumericLessThanEquals": {"s3:max-keys": "10"}}
  }
}
```

정책 조건에서 지정한 키가 요청 컨텍스트에 없으면 값이 일치하지 않습니다. 이 예제에서 는ListBucket 작업을 수행할 때 요청에 s3:max-keys 키가 항상 존재합니다. 이 정책에서 모든 Amazon S3 작업을 허용한 경우에는 값이 10 이하인 max-keys 컨텍스트 키를 포함하는 작업만 허용됩니다.

정책 변수를 Numeric 조건 연산자와 함께 사용할 수 없습니다.

날짜 조건 연산자

날짜 조건 연산자를 사용하여 키와 날짜/시간 값을 비교한 결과에 따라 액세스를 제한하는 Condition 요소를 생성할 수 있습니다. 이러한 조건 연산자는 aws:CurrentTime 키 또는 aws:EpochTime 키와 함께 사용합니다. 날짜/시간 값은 ISO 8601 날짜 형식의 W3C 구현 값 중의 하나, 또는 epoch(UNIX) 시간으로 지정해야 합니다.

> **Note**
>
> 날짜 조건 연산자에는 와일드카드를 사용할 수 없습니다.

조건 연산자	설명
DateEquals	특정 날짜 일치
DateNotEquals	불일치
DateLessThan	특정 날짜/시간 이전에 일치
DateLessThanEquals	특정 날짜/시간 또는 이전에 일치
DateGreaterThan	특정 날짜/시간 이후에 일치
DateGreaterThanEqu als	특정 날짜/시간 또는 이후에 일치

예를 들어, 다음 문에는 aws:TokenIssueTime 키와 함께 DateGreaterThan 조건 연산자를 사용하는 Condition 요소가 포함되어 있습니다. 이 조건은 요청을 생성하는 데 사용된 임시 보안 자격 증명이 2020년에 발급되었음을 지정합니다. 계정 멤버가 새로운 자격 증명을 사용하도록 매일 프로그래밍 방식으로 이 정책을 업데이트할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::account-id:user/*",
    "Condition": {"DateGreaterThan": {"aws:TokenIssueTime": "2020-01-01T00:00:01Z"}}
```

```
  }
}
```

정책 조건에서 지정한 키가 요청 컨텍스트에 없으면 값이 일치하지 않습니다. `aws:TokenIssueTime` 키는 사용자가 임시 자격 증명을 사용하여 요청을 생성하는 경우에만 요청 컨텍스트에 존재합니다. 액세스 키를 사용하는 AWS CLI, AWS API 또는 AWS SDK 요청에는 이 키가 존재하지 않습니다. 이 예제에서 IAM 사용자가 액세스 키를 보거나 편집하려고 하면 요청이 거부됩니다.

정책 변수를 `Date` 조건 연산자와 함께 사용할 수 없습니다.

불린 조건 연산자

부울 조건을 사용하여 키를 "true" 또는 "false"와 비교하고 그에 따라 액세스를 제한하는 `Condition` 요소를 생성할 수 있습니다.

조건 연산자	설명
`Bool`	부울 일치

예를 들어 이 아이덴티티 기반 정책은 `aws:SecureTransport` 키와 함께 `Bool` 조건 연산자를 사용하여 요청이 SSL을 통해 전달되지 않는 경우 대상 버킷 및 해당 콘텐츠로의 객체 및 객체 태그 복제를 거부합니다.

> **Important**
>
> 이 정책은 어떤 작업도 허용하지 않습니다. 이 정책을 특정 작업을 허용하는 다른 정책과 함께 사용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "BooleanExample",
      "Action": "s3:ReplicateObject",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
```

```
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      }
    }
  ]
}
```

정책 조건에서 지정한 키가 요청 컨텍스트에 없으면 값이 일치하지 않습니다. `aws:SecureTransport` 요청 컨텍스트는 true 또는 false를 반환합니다.

정책 변수를 `Boolean` 조건 연산자와 함께 사용할 수 있습니다.

이진 조건 연산자

`BinaryEquals` 조건 연산자를 사용하면 이진 형식의 키 값을 테스트하는 `Condition` 요소를 생성할 수 있습니다. 지정한 키 값을 정책 내 이진 값의 base-64 인코딩 표시와 바이트 단위(byte for byte)로 비교합니다.

```
"Condition" : {
  "BinaryEquals": {
    "key" : "QmluYXJ5VmFsdWVJbkJhc2U2NA=="
  }
}
```

정책 조건에서 지정한 키가 요청 컨텍스트에 없으면 값이 일치하지 않습니다.

정책 변수를 `Binary` 조건 연산자와 함께 사용할 수 없습니다.

IP 주소 조건 연산자

IP 주소 조건 연산자를 사용하여 IPv4/IPv6 주소 또는 IP 주소 범위와 키를 비교한 결과에 따라 액세스를 제한하는 `Condition` 요소를 생성할 수 있습니다. 이 조건에는 `aws:SourceIp` 키가 사용됩니다. 값은 표준 CIDR 형식(예: 203.0.113.0/24 또는 2001:DB8:1234:5678::/64)을 따라야 합니다. 연결된 라우팅 접두사 없이 IP 주소를 지정하면 IAM은 기본 접두사 값 `/32`를 사용합니다.

일부 AWS 서비스는 0의 범위를 나타내기 위해 ::을 사용해 IPv6를 지원합니다. 서비스가 IPv6를 지원하는지 여부를 확인하려면 서비스 설명서를 참조하세요.

조건 연산자	설명
IpAddress	지정된 IP 주소 또는 범위
NotIpAddress	지정된 IP 주소 또는 범위를 제외한 모든 IP 주소

예를 들어, 다음 문은 IpAddress 조건 연산자에 aws:SourceIp 키를 사용하여 IP 범위 203.0.113.0 - 203.0.113.255에서 요청이 전송되어야 한다고 지정하고 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::account-id:user/*",
    "Condition": {"IpAddress": {"aws:SourceIp": "203.0.113.0/24"}}
  }
}
```

aws:SourceIp 조건 키는 요청이 전송되는 IP 주소를 확인합니다. 요청이 Amazon EC2 인스턴스에서 전송된 경우에는 aws:SourceIp가 인스턴스의 퍼블릭 IP 주소로 계산되어야 합니다.

정책 조건에서 지정한 키가 요청 컨텍스트에 없으면 값이 일치하지 않습니다. aws:SourceIp 키는 요청자가 VPC 엔드포인트를 사용하여 요청을 생성한 경우를 제외하고 요청 컨텍스트에 항상 존재합니다. 이 경우 조건이 false를 반환하고 요청이 이 문에 의해 묵시적으로 거부됩니다.

정책 변수를 IpAddress 조건 연산자와 함께 사용할 수 없습니다.

다음 예제에서는 IPv4와 IPv6 주소를 혼합하여 조직의 유효 IP 주소를 모두 표현하는 방법을 보여줍니다. IPv6으로 전환하는 동안 조직의 정책이 계속 적용되도록 하려면 기존의 IPv4 주소 범위에 IPv6 범위를 더하여 정책을 업그레이드하는 것이 좋습니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "someservice:*",
    "Resource": "*",
    "Condition": {
      "IpAddress": {
```

```
        "aws:SourceIp": [
          "203.0.113.0/24",
          "2001:DB8:1234:5678::/64"
        ]
      }
    }
  }
}
```

사용자 자격으로 직접 테스트한 API를 호출하는 경우 `aws:SourceIp` 조건 키는 JSON 정책에서만 작동합니다. 서비스를 사용하여 사용자를 대신해 대상 서비스를 호출하는 경우, 대상 서비스는 원래 사용자의 IP 주소 대신 호출 서비스의 IP 주소를 봅니다. 이러한 상황은 예를 들어 AWS CloudFormation을 사용하여 인스턴스를 생성하는 Amazon EC2를 호출하는 경우에 발생할 수 있습니다. 현재로서는 JSON 정책에 따라 평가하기 위해 원본 IP 주소를 호출 서비스를 통해 대상 서비스로 보낼 방법이 없습니다. 이러한 서비스 API 호출 유형의 경우 `aws:SourceIp` 조건 키를 사용하지 마십시오.

Amazon 리소스 이름(ARN) 조건 연산자

Amazon 리소스 이름(ARN) 조건 연산자를 사용하면 키와 ARN을 비교한 결과에 따라 액세스를 제한하는 `Condition` 요소를 생성할 수 있습니다. ARN은 문자열로 알려져 있습니다.

조건 연산자	설명
`ArnEquals` , `ArnLike`	ARN 대소문자 구분 일치. ARN에서 콜론으로 구분된 구성요소 6개는 각각 별도로 확인하며, 다중 문자 일치 와일드카드(*) 또는 단일 문자 일치 와일드카드(?)가 추가될 수 있습니다. `ArnEquals` 과 `ArnLike` 조건 연산자는 동일하게 동작합니다.
`ArnNotEquals` , `ArnNotLike`	ARN 불일치. `ArnNotEquals` 과 `ArnNotLike` 조건 연산자는 동일하게 동작합니다.

정책 변수를 ARN 조건 연산자와 함께 사용할 수 있습니다.

다음의 리소스 기반 정책 예는 SNS 메시지를 전송하고 싶은 Amazon SQS 대기열에 연결된 정책을 보여줍니다. 이 예제에서는 선택한 대기열로 메시지를 전송할 수 있는 권한을 Amazon SNS에 부여하고 있습니다. 단, 서비스에서 특정 Amazon SNS 주제와 관련하여 메시지를 전송하는 경우로 제한됩니다. 대기열을 `Resource` 필드에, 그리고 Amazon SNS 주제는 `SourceArn` 키 값으로 지정합니다.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "123456789012"},
    "Action": "SQS:SendMessage",
    "Resource": "arn:aws:sqs:REGION:123456789012:QUEUE-ID",
    "Condition": {"ArnEquals": {"aws:SourceArn":
 "arn:aws:sns:REGION:123456789012:TOPIC-ID"}}
  }
}
```

정책 조건에서 지정한 키가 요청 컨텍스트에 없으면 값이 일치하지 않습니다. aws:SourceArn 키는 리소스가 리소스 소유자를 대신하여 다른 서비스를 호출하도록 서비스를 트리거하는 경우에만 요청 컨텍스트에 존재합니다. IAM 사용자가 이 작업을 직접 수행하려고 하면 조건이 false을 반환하고 요청이 이 문에 의해 묵시적으로 거부됩니다.

…IfExists 조건 연산자

Null 조건을 제외하고 조건 연산자 이름 끝에 IfExists를 추가할 수 있습니다(예: StringLikeIfExists). 이렇게 하면 "요청 컨텍스트에 조건 키가 있으면 정책에 지정된 대로 키를 처리합니다. 키가 없으면 조건 요소를 true로 평가합니다." 문의 다른 조건 요소는 여전히 불일치한 결과를 발생시킬 수 있지만 ...IfExists로 확인하면 누락되는 키는 없습니다. StringNotEqualsIfExists와 같은 부정 조건 연산자와 함께 "Effect": "Deny" 요소를 사용하는 경우 조건 키가 제공되지 않아도 요청이 계속 거부됩니다.

IfExists 사용 예제

대부분 조건 키는 특정 형식의 리소스 정보를 의미하기 때문에 해당 형식의 리소스에 액세스할 때만 존재합니다. 이러한 조건 키는 다른 형식의 리소스에는 표시되지 않습니다. 그렇다고 정책 문이 한 가지 형식의 리소스에만 적용된다고 해서 문제가 되지는 않습니다. 하지만 정책 문이 여러 서비스의 작업을 참조하는 경우나, 혹은 한 가지 서비스 내에서 임의의 작업이 동일한 서비스에서 여러 가지 다른 리소스 형식에 액세스하는 경우처럼 단일 문이 여러 유형의 리소스에 적용될 수 있는 경우도 있습니다. 이런 경우 오직 한 가지 리소스에만 적용되는 조건 키를 정책 문에 추가하면 정책 문의 Condition 요소를 충족하지 못하고 결국 "Effect"가 적용되지 않습니다.

예를 들어 다음과 같은 정책 예제를 살펴보십시오.

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

```
    "Effect": "Allow",
    "Action": "ec2:RunInstances",
    "Resource": "*",
    "Condition": {"StringLike": {"ec2:InstanceType": [
      "t1.*",
      "t2.*",
      "m3.*"
    ]}}
  }
}
```

위 정책의 의도는 사용자가 t1, t2 또는 m3 형식의 인스턴스를 모두 실행할 수 있도록 하는 것이 목적입니다. 하지만 인스턴스를 실행하려면 인스턴스 외에도 이미지, 키 페어, 보안 그룹 등 다양한 리소스에 액세스해야 합니다. 전체 문은 인스턴스를 실행하는 데 필요한 모든 리소스와 비교하여 평가됩니다. 하지만 이러한 추가 리소스에는 ec2:InstanceType 조건 키가 없기 때문에 StringLike 검사는 fail로 끝나고 사용자에게 권한이 부여되지 않아 어떤 인스턴스 유형도 실행하지 못합니다.

이 문제를 해결하려면 그 대신 StringLikeIfExists 조건 연산자를 사용해야 합니다. 이렇게 하면 조건 키가 존재하는 경우에만 테스트가 실행됩니다. 그 결과 다음 정책은 이렇게 해석할 수 있습니다. “검사 대상 리소스에 'ec2:InstanceType' 조건 키가 있으면 키 값이 t1., t2. 또는 m3.로 시작할 때에만 작업을 허용합니다. 검사 대상 리소스에 조건 키가 없으면 그냥 둡니다." 조건 키 값에 별표(*)를 StringLikeIfExists 조건 연산자와 함께 사용하면 와일드카드로 해석되어 문자열이 부분적으로 일치합니다. DescribeActions 문에는 콘솔에서 해당 인스턴스를 보는 데 필요한 작업이 포함되어 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RunInstance",
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "*",
            "Condition": {
                "StringLikeIfExists": {
                    "ec2:InstanceType": [
                        "t1.*",
                        "t2.*",
                        "m3.*"
             ]}}
        },
```

```
        {
            "Sid": "DescribeActions",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeImages",
                "ec2:DescribeInstances",
                "ec2:DescribeVpcs",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        }]
}
```

조건 키의 존재를 확인하는 조건 연산자

`Null` 조건 연산자를 사용하여 권한을 부여하는 시점에 조건 키가 없는지 확인합니다. 정책 문에서는 `true`(키가 부재하며 값이 null임) 또는 `false`(키가 존재하며 값이 null이 아님)를 사용합니다.

정책 변수를 `Null` 조건 연산자와 함께 사용할 수 없습니다.

예를 들어, 이 조건 연산자를 사용하여 작업 시 사용자가 자신의 자격 증명을 사용하는지, 혹은 임시 자격 증명을 사용하는지 알 수 있습니다. 사용자가 임시 자격 증명을 사용하는 경우에는 `aws:TokenIssueTime` 키가 존재하며, 값을 갖고 있습니다. 다음은 Amazon EC2 API 사용자의 경우 임시 자격 증명의 사용이 제한된다는 것(키가 존재해서는 안 됨)을 명시하는 조건을 나타낸 예제입니다.

```
{
  "Version": "2012-10-17",
  "Statement":{
      "Action":"ec2:*",
      "Effect":"Allow",
      "Resource":"*",
      "Condition":{"Null":{"aws:TokenIssueTime":"true"}}
  }
}
```

다수의 컨텍스트 키 또는 값을 사용하는 조건

정책의 `Condition` 요소를 사용하여 요청에서 다수의 컨텍스트 키, 혹은 단일 컨텍스트 키에 대한 다수의 값을 테스트할 수 있습니다. 프로그래밍 방식이든, AWS Management Console을 사용하든 상관

없이 AWS에게 요청할 경우 요청에 보안 주체, 작업, 태그 등에 대한 정보가 포함됩니다. 컨텍스트 키를 사용하여 정책 조건에서 컨텍스트 키가 지정된 요청에서 일치하는 컨텍스트 키의 값을 테스트할 수 있습니다. 요청에 포함되는 정보 및 데이터에 대한 자세한 내용은 요청 컨텍스트 섹션을 참조하세요.

주제

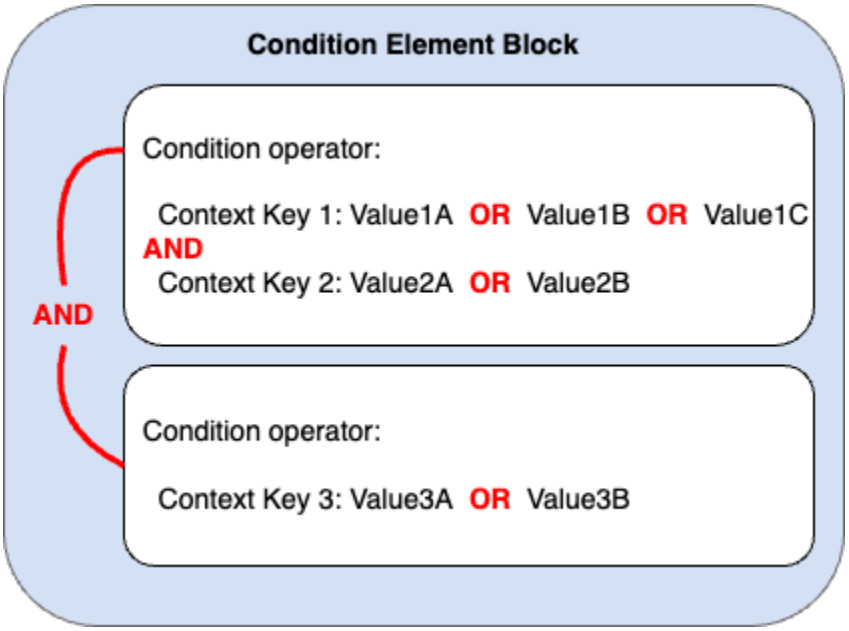
- 다수의 컨텍스트 키 또는 값에 대한 평가 로직
- 부정 일치 조건 연산자에 대한 평가 로직

다수의 컨텍스트 키 또는 값에 대한 평가 로직

Condition 요소에는 여러 조건 연산자를 추가할 수 있으며, 각 조건 연산자마다 다수의 컨텍스트 키-값 페어가 포함될 수 있습니다. 대부분의 컨텍스트 키에는 별도로 지정되지 않는 한 여러 개의 값을 사용할 수 있습니다.

- 정책 문에 여러 개의 조건 연산자가 있는 경우 조건 연산자는 논리 연산자 AND를 사용하여 평가됩니다.
- 정책 문에서 다수의 컨텍스트 키가 단일 조건 연산자에 추가되어 있는 경우 논리 연산자 AND를 사용하여 컨텍스트 키가 평가됩니다.
- 단일 조건 연산자에 하나의 컨텍스트 키에 대한 다수의 값이 포함된 경우 이러한 값은 논리 연산자 OR을 사용하여 평가됩니다.
- 단일 부정 일치 조건 연산자에 하나의 컨텍스트 키에 대한 다수의 값이 포함된 경우 이러한 값은 논리 연산자 NOR을 사용하여 평가됩니다.

조건 요소 블록의 모든 컨텍스트 키가 true로 확인되어야 원하는 Allow 또는 Deny 효과를 간접적으로 호출할 수 있습니다. 다음 그림은 여러 조건 연산자와 컨텍스트 키-값 쌍이 있는 조건에 대한 평가 로직을 보여줍니다.



예를 들어, 다음 S3 버킷 정책은 이전 그림이 정책에 어떻게 표시되는지 보여줍니다. 조건 블록에는 조건 연산자 `StringEquals` 및 `ArnLike`와, 컨텍스트 키 `aws:PrincipalTag` 및 `aws:PrincipalArn`이 포함됩니다. 원하는 `Allow` 또는 `Deny` 효과를 간접적으로 호출하기 위해 조건 블록의 모든 컨텍스트 키가 true로 확인되어야 합니다. 요청하는 사용자는 정책에 지정된 태그 키 값 중 하나를 포함하는 두 개의 보안 주체 태그 키(department와 role)를 모두 가지고 있어야 합니다. 또한 요청하는 사용자의 보안 주체 ARN이 정책에서 지정된 `aws:PrincipalArn` 값 중 하나와 일치해야 true로 평가됩니다.

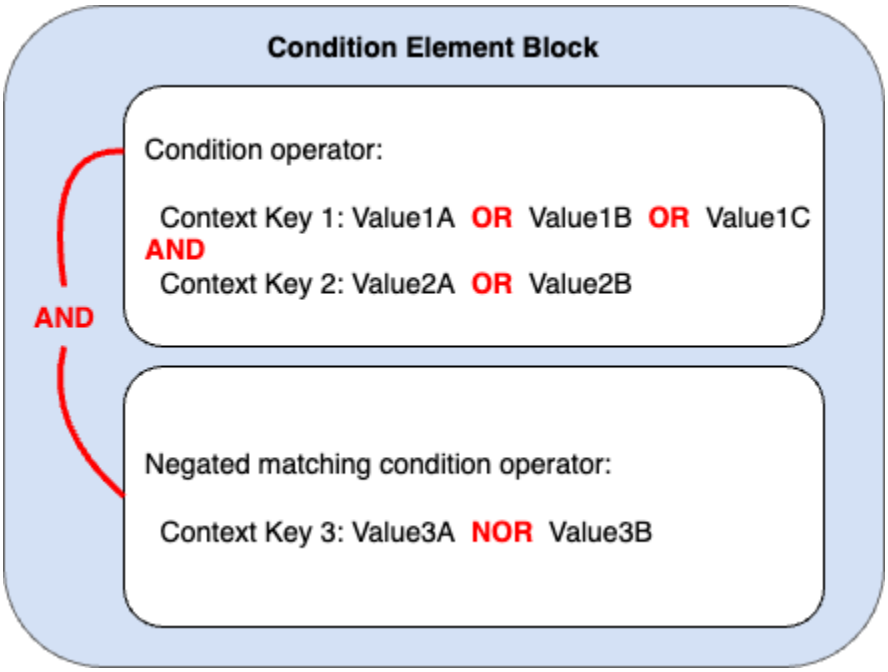
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::222222222222:root"
      },
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/department": [
            "finance",
            "hr",
            "legal"
          ],
```

```
          "aws:PrincipalTag/role": [
            "audit",
            "security"
          ]
        },
        "ArnLike": {
          "aws:PrincipalArn": [
            "arn:aws:iam::222222222222:user/Ana",
            "arn:aws:iam::222222222222:user/Mary"
          ]
        }
      }
    }
  ]
}
```

부정 일치 조건 연산자에 대한 평가 로직

StringNotEquals 또는 ArnNotLike와 같은 일부 조건 연산자는 부정 일치를 사용하여 정책의 컨텍스트 키-값 쌍과 요청의 컨텍스트 키-값 쌍을 비교합니다. 부정 일치 조건 연산자가 포함된 정책의 단일 컨텍스트 키에 대해 다수의 값이 지정되면 유효한 권한은 논리 연산자 NOR와 같이 작동합니다. 부정 일치에서 논리 연산자 NOR 또는 NOT OR은 모든 값이 false로 평가되는 경우에만 true를 반환합니다.

다음 그림은 여러 조건 연산자와 컨텍스트 키-값 쌍이 있는 조건에 대한 평가 로직을 보여줍니다. 그림에는 컨텍스트 키 3에 대한 부정 일치 조건 연산자가 포함되어 있습니다.



예를 들어, 다음 S3 버킷 정책은 이전 그림이 정책에 어떻게 표시되는지 보여줍니다. 조건 블록에는 조건 연산자 `StringEquals` 및 `ArnNotLike`와, 컨텍스트 키 `aws:PrincipalTag` 및 `aws:PrincipalArn`이 포함됩니다. 원하는 `Allow` 또는 `Deny` 효과를 간접적으로 호출하기 위해 조건 블록의 모든 컨텍스트 키가 true로 확인되어야 합니다. 요청하는 사용자는 정책에 지정된 태그 키 값 중 하나를 포함하는 두 개의 보안 주체 태그 키(department와 role)를 모두 가지고 있어야 합니다. `ArnNotLike` 조건 연산자가 부정 일치를 사용하므로 요청하는 사용자의 보안 주체 ARN이 정책에서 지정된 `aws:PrincipalArn` 값 중에서 일치하는 게 없어야 true로 평가됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::222222222222:root"
      },
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/department": [
            "finance",
            "hr",
            "legal"
          ],
          "aws:PrincipalTag/role": [
            "audit",
            "security"
          ]
        },
        "ArnNotLike": {
          "aws:PrincipalArn": [
            "arn:aws:iam::222222222222:user/Ana",
            "arn:aws:iam::222222222222:user/Mary"
          ]
        }
      }
    }
  ]
}
```

단일 값 vs. 다중 값 컨텍스트 키

단일 값 컨텍스트 키와 다중 값 컨텍스트 키의 차이는 정책 조건의 값 수가 아니라 요청 컨텍스트의 값 수에 따라 달라집니다.

- 단일 값 조건 컨텍스트 키는 요청 컨텍스트에서 최대 하나의 값을 갖습니다. 예를 들어, AWS에서 리소스에 태그를 지정할 수 있습니다. 리소스 태그는 태그 키-값 페어로 저장됩니다. 리소스 태그 키는 단일 태그 값을 가질 수 있습니다. 따라서 the section called “ResourceTag”는 단일 값 컨텍스트 키입니다. 단일 값 컨텍스트 키에는 조건 집합 연산자를 사용하지 마세요.
- 다중 값 조건 컨텍스트 키는 요청 컨텍스트에서 여러 값을 가질 수 있습니다. 예를 들어, AWS의 리소스에 태그를 지정하고 요청에 여러 태그 키-값 페어를 포함할 수 있습니다. 따라서 the section called “TagKeys”는 다중 값 컨텍스트 키입니다. 다중 값 컨텍스트 키에는 조건 집합 연산자가 필요합니다.

> **Important**
>
> 다중 값 컨텍스트 키에는 조건 집합 연산자가 필요합니다. 단일 값 컨텍스트 키에는 조건 집합 연산자 `ForAllValues` 또는 `ForAnyValue`를 사용하지 마세요. 조건 집합 연산자에 대한 자세한 내용은 다중 값 컨텍스트 키 섹션을 참조하세요.

단일 값 및 다중 값 분류는 AWS 글로벌 조건 컨텍스트 키 주제의 값 유형으로 각 조건 컨텍스트 키의 설명에 포함됩니다. 서비스 승인 참조는 `ArrayOf` 접두사 뒤에 조건 연산자 범주 유형이 붙는 형식으로 다중 값 컨텍스트 키에 대해 서로 다른 값 유형 분류를 사용합니다. 예: `ArrayOfString` 또는 `ArrayOfARN`.

예를 들어, 요청은 최대 하나의 VPC 엔드포인트에서 시작될 수 있으므로 the section called “SourceVpce”의 경우 단일 값 컨텍스트 키입니다. 서비스에는 서비스에 속한 서비스 보안 주체 이름이 두 개 이상 있을 수 있으므로 aws:PrincipalServiceNamesList의 경우 다중 값 컨텍스트 키입니다.

사용 가능한 단일 값 컨텍스트 키를 정책 변수로 사용할 수 있습니다. 다중 값 컨텍스트 키는 정책 변수로 사용할 수 없습니다. 정책 변수에 대한 자세한 내용은 IAM 정책 요소: 변수 및 태그 섹션을 참조하세요.

다중 값 컨텍스트 키에는 조건 집합 연산자 `ForAllValues` 또는 `ForAnyValue`가 필요합니다. the section called “RequestTag” 및 the section called “ResourceTag”와 같은 키-값 쌍을 포함하는 컨텍스트 키는 여러 *`tag-key`* 값이 있을 수 있으므로 혼란을 야기할 수 있습니다. 하지만 각 *`tag-key`*의 경우 하나의 값만 가질 수 있으므로 `aws:RequestTag` 및 `aws:ResourceTag`는 둘 다 단일 값 컨텍스

트 키입니다. 단일 값 컨텍스트 키와 함께 조건 집합 연산자를 사용하면 정책이 과도하게 허용될 수 있습니다.

다중 값 컨텍스트 키

조건 컨텍스트 키를 여러 키 값이 있는 요청 컨텍스트와 비교하려면 `ForAllValues` 또는 `ForAnyValue` 집합 연산자를 사용해야 합니다. 이러한 집합 연산자는 요청의 태그 집합 및 정책 조건의 태그 집합과 같이 두 값 집합을 비교하는 데 사용됩니다.

`ForAllValues` 및 `ForAnyValue` 한정자는 조건 연산자에 집합 연산 기능을 추가하므로 여러 값이 포함된 요청 컨텍스트 키를 정책 조건의 여러 컨텍스트 키 값에 대해 테스트할 수 있습니다. 또한 와일드카드 또는 변수를 사용하여 정책에 다중 값 문자열 컨텍스트 키를 포함하는 경우 `StringLike` 조건 연산자를 사용해야 합니다. 여러 조건 키 값은 배열처럼 대괄호로 묶어야 합니다. 예: `"Key2": ["Value2A", "Value2B"]`.

- `ForAllValues` - 이 한정자는 요청 집합의 모든 멤버 값이 조건 컨텍스트 키 집합의 하위 집합인지 여부를 테스트합니다. 요청의 모든 컨텍스트 키 값이 정책에 있는 하나 이상의 컨텍스트 키 값과 일치하면 조건이 true를 반환합니다. 요청에 컨텍스트 키가 없거나 컨텍스트 키 값이 빈 문자열과 같은 null 데이터 세트로 확인되는 경우에도 true를 반환합니다. 누락된 컨텍스트 키나 값이 비어 있는 컨텍스트 키가 true로 평가되지 않도록 하려면 정책의 `Null` 조건 연산자에 false 값을 포함하여 컨텍스트 키가 존재하고 그 값이 null이 아닌지 확인할 수 있습니다.

  > **Important**
  >
  > `Allow` 효과에 `ForAllValues`를 사용하는 경우 요청 컨텍스트에 컨텍스트 키가 누락되거나 컨텍스트 키의 값이 비어 있는 것을 예상할 수 없으면 지나치게 허용적일 수 있으므로 주의하세요. 정책의 `Null` 조건 연산자에 false 값을 포함하여 컨텍스트 키가 존재하는지와 그 값이 null이 아닌지 여부를 확인할 수 있습니다. 예시는 태그 키를 기반으로 액세스 제어에서 확인하세요.

- `ForAnyValue` - 이 한정자는 요청 컨텍스트 키 값 집합에서 하나 이상의 멤버가 정책 조건의 컨텍스트 키 값 집합에서 하나 이상의 멤버와 일치하는지 테스트합니다. 요청의 컨텍스트 키 값 중 하나가 정책의 컨텍스트 키 값 중 하나와 일치하면 컨텍스트 키는 true를 반환합니다. 일치하는 컨텍스트 키가 없거나 null 데이터 세트의 경우 조건에서 false를 반환합니다.

> **Note**
>
> 단일 값 컨텍스트 키와 다중 값 컨텍스트 키의 차이는 정책 조건의 값 수가 아니라 요청 컨텍스트의 값 수에 따라 달라집니다.

조건 정책 예제

IAM 정책에서 요청 컨텍스트와 비교하기 위해 단일 값 및 다중 값 컨텍스트 키 모두에 대해 여러 값을 지정할 수 있습니다. 다음은 다수의 컨텍스트 키와 값을 포함한 정책 조건을 보여주는 일련의 정책 예제입니다.

> **Note**
>
> 이 참조 설명에 포함시킬 정책을 제출하고자 하는 경우 이 페이지의 하단에 있는 의견 버튼을 사용합니다. IAM 자격 증명 기반 정책의 예는 IAM 자격 증명 기반 정책의 예 섹션을 참조하세요.

조건 정책 예: 단일 값 컨텍스트 키

- 단일 값 컨텍스트 키가 있는 다수의 조건 블록. (이 예제 보기.)
- 다수의 단일 값 컨텍스트 키와 값이 있는 하나의 조건 블록. (이 예제 보기.)

조건 정책 예: 다중 값 컨텍스트 키

- 조건 집합 연산자 `ForAllValues`를 포함하는 거부 정책 (이 예제 보기.)
- 조건 집합 연산자 `ForAnyValue`를 포함하는 거부 정책 (이 예제 보기.)

다중 값 컨텍스트 키 예제

다음은 다중 값 컨텍스트 키로 정책 조건을 만드는 방법을 보여주는 일련의 정책 예제입니다.

예: 조건 집합 연산자 ForAllValues를 포함한 거부 정책

다음 예제 자격 증명 기반 정책은 요청에 특정 태그 키 접두사가 포함된 경우 IAM 태깅 작업의 사용을 거부합니다. 컨텍스트 키 `aws:TagKeys`의 각 값에는 부분 문자열 일치를 위한 와일드카드(*)가 포함됩니다. 정책에는 컨텍스트 키 `aws:TagKeys`가 포함된 `ForAllValues` 집합 연산자가 포함되는데, 요

청 컨텍스트 키에 여러 값이 포함될 수 있기 때문입니다. 컨텍스트 키 aws:TagKeys에서 true를 반환하려면 요청의 모든 값이 정책에서 하나 이상의 값과 일치해야 합니다.

요청에 컨텍스트 키가 없거나 컨텍스트 키 값이 빈 문자열과 같은 null 데이터 세트로 확인되는 경우에도 ForAllValues 집합 연산자에서 true를 반환합니다. 누락된 컨텍스트 키나 값이 비어 있는 컨텍스트 키가 true로 평가되지 않도록 하려면 정책의 Null 조건 연산자에 false 값을 포함하여 요청에 컨텍스트 키가 존재하고 그 값이 null이 아닌지 확인할 수 있습니다.

> **Important**
>
> 이 정책은 어떤 작업도 허용하지 않습니다. 이 정책을 특정 작업을 허용하는 다른 정책과 함께 사용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyRestrictedTags",
      "Effect": "Deny",
      "Action": [
        "iam:Tag*",
        "iam:UnTag*"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "Null": {
          "aws:TagKeys": "false"
        },
        "ForAllValues:StringLike": {
          "aws:TagKeys": [
            "key1*",
            "key2*",
            "key3*"
          ]
        }
      }
    }
  ]
```

```
}
```

예: 조건 집합 연산자 ForAnyValue를 포함한 거부 정책

다음 자격 증명 기반 정책 예제는 정책에 지정된 태그 키(environment 또는 webserver) 중 하나로 태그가 지정된 스냅샷이 있는 경우 EC2 인스턴스 볼륨의 스냅샷 생성을 거부합니다. 정책에는 컨텍스트 키 aws:TagKeys가 포함된 ForAnyValue 집합 연산자가 포함되는데, 요청 컨텍스트 키에 여러 값이 포함될 수 있기 때문입니다. 정책에 지정된 태그 키 값 중 하나가 태깅 요청에 포함된 경우 aws:TagKeys 컨텍스트 키는 true를 반환하여 거부 정책 효과를 간접적으로 호출합니다.

> **Important**
>
> 이 정책은 어떤 작업도 허용하지 않습니다. 이 정책을 특정 작업을 허용하는 다른 정책과 함께 사용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ec2:CreateSnapshot",
                "ec2:CreateSnapshots"
            ],
            "Resource": "arn:aws:ec2:us-west-2::snapshot/*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": ["environment", "webserver"]
                }
            }
        }
    ]
}
```

단일 값 컨텍스트 키 정책 예제

다음은 단일 값 컨텍스트 키로 정책 조건을 만드는 방법을 보여주는 일련의 정책 예제입니다.

예: 단일 값 컨텍스트 키가 있는 다수의 조건 블록

조건 블록에 각각 단일 컨텍스트 키가 있는 다수의 조건이 있는 경우 모든 컨텍스트 키가 true로 확인되어야 원하는 Allow 또는 Deny 효과가 간접적으로 호출됩니다. 부정 일치 조건 연산자를 사용하면 조건 값의 평가 로직이 반전됩니다.

다음 예제에서는 사용자가 EC2 볼륨을 생성하고 볼륨 생성 도중 볼륨에 태그를 적용합니다. 요청 컨텍스트에는 컨텍스트 키 aws:RequestTag/project의 값이 포함되어야 하고, 컨텍스트 키 aws:ResourceTag/environment의 값은 프로덕션 이외의 모두 가능합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:CreateVolume",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateTags",
      "Resource": "arn:aws:ec2:::volume/*",
      "Condition": {
        "StringLike": {
          "aws:RequestTag/project": "*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateTags",
      "Resource": "arn:aws:ec2:region:account:*/*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceTag/environment": "production"
        }
      }
    }
  ]
}
```

요청 컨텍스트에는 프로젝트 태그-값이 포함되어야 하고 프로덕션 리소스가 Allow 효과를 간접적으로 호출하도록 생성할 수 없습니다. 프로젝트 이름이 Feature3이고 QA 리소스 태그가 있으므로 다음 EC2 볼륨이 생성됩니다.

```
aws ec2 create-volume \
    --availability-zone us-east-1a \
    --volume-type gp2 \
    --size 80 \
    --tag-specifications 'ResourceType=volume,Tags=[{Key=project,Value=Feature3},
{Key=environment,Value=QA}]'
```

예: 다수의 단일 값 컨텍스트 키와 값이 있는 하나의 조건 블록

조건 블록에 다수의 컨텍스트 키가 있고 각 컨텍스트 키에 다수의 값이 있는 경우 각 컨텍스트 키가 true로 확인되어야 원하는 Allow 또는 Deny 효과에서 하나 이상의 키 값이 간접적으로 호출됩니다. 부정 일치 조건 연산자를 사용하면 컨텍스트 키 값의 평가 로직이 반전됩니다.

다음 예제를 통해 사용자는 Amazon Elastic Container Service 클러스터에서 작업을 시작하고 실행할 수 있습니다.

- 요청 컨텍스트에는 aws:RequestTag/environment 컨텍스트 키 AND에 대해 production 또는 pre-prod가 포함되어야 합니다.
- ecs:cluster 컨텍스트 키를 통해 작업이 default1 또는 default2 ARN ECS 클러스터터에서 실행됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:RunTask",
        "ecs:StartTask"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/environment": [
```

```
            "production",
            "prod-backup"
          ]
        },
        "ArnEquals": {
          "ecs:cluster": [
            "arn:aws:ecs:us-east-1:111122223333:cluster/default1",
            "arn:aws:ecs:us-east-1:111122223333:cluster/default2"
          ]
        }
      }
    }
  ]
}
```

IAM 정책 요소: 변수 및 태그

정책 작성 시 리소스나 조건 키의 정확한 값을 모를 경우 AWS Identity and Access Management(IAM) 정책 변수를 자리 표시자로 사용하세요.

> **Note**
>
> AWS에서 변수를 확인할 수 없는 경우 전체 문이 잘못되는 문제가 발생할 수 있습니다. 예를 들어 `aws:TokenIssueTime` 번수를 사용하는 경우 변수는 요청자가 임시 자격 증명을 사용하여 인증된 경우(IAM 역할)에만 값을 확인합니다. 잘못된 문을 유발하는 변수를 방지하려면 ...IfExists 조건 연산자를 사용하세요.

주제

소개

IAM 정책에서는 다양한 작업을 통해 액세스를 제어하려는 특정 리소스에 이름을 지정할 수 있습니다. 예를 들어, 다음 정책은 사용자가 marketing 프로젝트를 위해 S3 버킷 DOC-EXAMPLE-BUCKET의 객체를 나열하고, 읽고, 쓸 수 있도록 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
      "Condition": {"StringLike": {"s3:prefix": ["marketing/*"]}}
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/marketing/*"]
    }
  ]
}
```

정책을 작성하다 보면 정확한 리소스 이름을 모를 때도 있습니다. 사용자마다 고유한 정책 사본을 만들 필요 없이 여러 사용자에게 작용하도록 정책을 일반화해야 할 수 있습니다. 각 사용자별로 별도의 정책을 생성하는 대신, 해당 그룹의 모든 사용자에게 적용되는 단일 그룹 정책을 생성하는 것이 좋습니다.

정책에 변수 사용

정책에서 자리 표시자를 설정하는 정책 변수를 사용하여 정책 내에서 동적 값을 정의할 수 있습니다.

변수는 요청에 있는 값의 변수 이름을 감싸는 중괄호 한 쌍(**{ }**)을 사용하여 **$** 접두사 뒤에 표시됩니다.

정책을 평가할 때는 이 정책 변수가 요청에서 전달되는 조건 컨텍스트 키에서 온 값으로 바뀝니다. 변수는 아이덴티티 기반 정책, 리소스 정책, 서비스 제어 정책, 세션 정책 및 VPC 엔드포인트 정책에 사용할 수 있습니다. 권한 경계로 사용되는 아이덴티티 기반 정책은 정책 변수도 지원합니다.

글로벌 조건 컨텍스트 키는 AWS 서비스 전반의 요청에서 변수로 사용할 수 있습니다. 서비스별 조건 키는 AWS 리소스와 상호 작용할 때 변수로도 사용할 수 있지만, 이러한 조건 키를 지원하는 리소스에

대한 요청을 할 때만 사용할 수 있습니다. 각 AWS 서비스 및 리소스에 사용할 수 있는 컨텍스트 키 목록은 서비스 승인 참조를 참조하세요. 특정 상황에서는 글로벌 조건 컨텍스트 키에 값을 채울 수 없습니다. 각 키에 대한 자세한 내용은 AWS 글로벌 조건 컨텍스트 키를 참조하세요.

> **Important**
>
> - 키 이름은 대/소문자를 구분하지 않습니다. 예를 들어, `aws:CurrentTime`은 `AWS:currenttime`과 같습니다.
> - 단일 값 조건 키를 변수로 사용할 수 있습니다. 다중 값 조건 키는 변수로 사용할 수 없습니다.

다음 예에서는 특정 리소스 이름을 정책 변수로 대체하는 IAM 역할 또는 사용자에 대한 정책을 보여 줍니다. `aws:PrincipalTag` 조건 키를 활용하여 이 정책을 재사용할 수 있습니다. 이 정책을 평가할 때 `${aws:PrincipalTag/team}`은 버킷 이름이 `team` 보안 주체 태그의 팀 이름으로 끝나는 경우에만 작업을 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
      "Condition": {"StringLike": {"s3:prefix": ["${aws:PrincipalTag/team}/*"]}}
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/${aws:PrincipalTag/team}/*"]
    }
  ]
}
```

변수는 $ 접두사 뒤에 중괄호({ })를 사용하여 표시합니다. ${ } 문자 안에는 정책에서 사용할 요청 값의 이름을 추가할 수 있습니다. 사용할 수 있는 값은 이 페이지 후반에서 다루겠습니다.

이 글로벌 조건 키에 대한 자세한 내용은 글로벌 조건 키 목록의 aws:PrincipalTag/tag-key 섹션을 참조하세요.

> **Note**
>
> 정책 변수를 사용하려면 Version 요소를 문에 추가해야 하며, 이때 버전은 정책 변수를 지원하는 버전으로 설정해야 합니다. 변수는 버전 2012-10-17에서 도입되었습니다. 정책 언어의 조기 버전은 정책 변수를 지원하지 않기 때문입니다. Version 요소를 추가하지 않고 해당 버전 날짜로 설정하면 ${aws:username} 같은 변수가 정책에서 리터럴 문자열로 처리됩니다. Version 정책 요소는 정책 버전과 다릅니다. Version 정책 요소는 정책 내에서 사용되며 정책 언어의 버전을 정의합니다. 반면에 정책 버전은 IAM에서 고객 관리형 정책을 변경할 때 생성됩니다. 변경된 정책은 기존 정책을 덮어쓰지 않습니다. 대신 IAM에서 관리형 정책의 새 버전을 생성합니다. Version 정책 요소에 대한 자세한 정보는 the section called “Version”을 참조하세요. 정책 버전에 대한 자세한 정보는 the section called “IAM 정책 버전 관리” 섹션을 참조하세요.

보안 주체가 S3 버킷의 /David 경로에서 객체를 가져오도록 허용하는 정책은 다음과 같습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:GetObject"],
    "Resource": ["arn:aws:s3::DOC-EXAMPLE-BUCKET/David/*"]
  }]
}
```

이 정책이 사용자 David에게 연결된 경우 해당 사용자는 자신의 S3 버킷에서 객체를 가져오지만 각 사용자에 대해 사용자 이름을 포함하는 별도의 정책을 생성해야 합니다. 그런 다음 각 정책을 개별 사용자에 연결합니다.

정책 변수를 사용하여 재사용할 수 있는 정책을 생성할 수 있습니다. 다음 정책은 aws:PrincipalTag의 tag-key 값이 요청에서 전달된 tag-key owner 값과 일치하는 경우 사용자가 Amazon S3 버킷에서 객체를 가져올 수 있도록 허용합니다.

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

```
    "Sid": "AllowUnlessOwnedBySomeoneElse",
    "Effect": "Allow",
    "Action": ["s3:GetObject"],
    "Resource": ["*"],
    "Condition": {
        "StringEquals": {
          "s3:ExistingObjectTag/owner": "${aws:PrincipalTag/owner}"
        }
      }
    }
  ]
}
```

이와 같이 사용자 대신 정책 변수를 사용하는 경우 개별 사용자마다 별도의 정책이 있을 필요가 없습니다. 다음 예시에서는 임시 보안 자격 증명을 사용하여 제품 관리자가 수임하는 IAM 역할에 정책이 연결됩니다. 사용자가 Amazon S3 객체 추가를 요청하면 IAM은 현재 요청의 `dept` 태그 값을 `${aws:PrincipalTag}` 변수로 대체하고 정책을 평가합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowOnlyDeptS3Prefix",
    "Effect": "Allow",
    "Action": ["s3:GetObject"],
    "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/${aws:PrincipalTag/dept}/*"],
    }
  ]
}
```

정책 변수로서의 태그

일부 AWS 서비스에서는 사용자 지정 속성을 해당 서비스가 생성한 리소스에 연결할 수 있습니다. 예를 들어 Amazon S3 버킷 또는 IAM 사용자에 태그를 적용할 수 있습니다. 태그는 키 값 페어입니다. 태그 키 이름을 정의하고, 해당 키 이름에 연결된 값을 정의합니다. 예를 들어 **`department`** 키와 **`Human Resources`** 값으로 태그를 만들 수 있습니다. IAM 엔터티 태그 지정에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요. 다른 AWS 서비스에서 생성한 리소스에 대한 태그 지정 정보는 해당 서비스의 문서 섹션을 참조하세요. Tag Editor에 대한 자세한 내용은 AWS Management Console 사용 설명서의 Tag Editor 작업 섹션을 참조하세요.

IAM 리소스에 태그를 추가하면 IAM 리소스를 쉽게 찾고, 구성하고, 추적할 수 있습니다. 또한 IAM 자격 증명에 태그를 지정하여 리소스에 대한 액세스를 제어하거나 자체 태그를 지정할 수 있습니다. 태그

를 사용하여 액세스를 제어하는 방법에 대한 자세한 내용은 태그를 사용하여 IAM 사용자 및 역할에 대한 액세스 제어 섹션을 참조하세요.

정책 변수를 사용할 수 있는 경우

정책 변수는 Resource 요소를 비롯해 Condition 요소의 문자열 비교에 사용할 수 있습니다.

리소스 요소

Resource 요소에서 정책 변수를 사용할 수 있지만, ARN의 리소스 부분에서만 사용할 수 있습니다. ARN의 이 부분은 5번째 콜론(:) 뒤에 나타납니다. 변수를 사용하여 서비스나 계정과 같이 5번째 콜론 앞의 ARN 부분을 바꿀 수 없습니다. ARN 형식에 대한 자세한 내용은 IAM ARN 섹션을 참조하세요.

ARN의 일부를 태그 값으로 바꾸려면 접두사와 키 이름을 ${ }로 묶습니다. 예를 들어 다음 Resource 요소는 요청한 사용자의 department 태그 값과 동일한 이름의 버킷만 참조합니다.

```
"Resource": ["arn:aws::s3:::bucket/${aws:PrincipalTag/department}"]
```

많은 AWS 리소스에서 사용자가 생성한 이름이 포함된 ARN을 사용합니다. 다음 IAM 정책은 access-project, access-application 및 access-environment 태그 값이 일치하는 의도된 사용자만 리소스를 수정할 수 있도록 합니다. 또한 * 와일드카드 일치로 사용자 지정 리소스 이름 접미사를 허용할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessBasedOnArnMatching",
      "Effect": "Allow",
      "Action": [
        "sns:CreateTopic",
        "sns:DeleteTopic"],
      "Resource": ["arn:aws:sns:*:*:${aws:PrincipalTag/access-project}-
${aws:PrincipalTag/access-application}-${aws:PrincipalTag/access-environment}-*"
      ]
    }
  ]
}
```

조건 요소

문자열 연산자 또는 ARN 연산자와 관련된 모든 조건의 Condition 값에 정책 변수를 사용할 수 있습니다. 문자열 연산자는 StringEquals, StringLike 및 StringNotLike를 포함합니다. ARN 연산

자에는 ArnEquals 및 ArnLike가 포함됩니다. Numeric, Date, Boolean, Binary, IP Address 또는 Null 연산자와 같은 다른 연산자와 함께 정책 변수를 사용할 수 없습니다. 조건 연산자에 대한 자세한 내용은 IAM JSON 정책 요소: 조건 연산자 섹션을 참조하세요.

Condition 요소 표현식에서 태그를 참조할 때는 관련 접두사와 키 이름을 조건 키로 사용하세요. 그런 다음 조건 값에서 테스트할 값을 사용합니다.

예를 들어, 다음 정책 예에서는 costCenter 태그가 리소스에 연결된 경우에만 사용자에 대한 모든 액세스를 허용합니다. 태그의 값은 12345 또는 67890이어야 합니다. 태그에 값이 없거나 다른 값이 있으면 요청이 실패합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "iam:*user*"
       ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iam:ResourceTag/costCenter": [ "12345", "67890" ]
        }
      }
    }
  ]
}
```

값이 없는 정책 변수

값이 없거나 요청의 승인 컨텍스트에 없는 조건 컨텍스트 키를 정책 변수에서 참조하는 경우, 해당 값은 사실상 null입니다. 같거나 비슷한 값이 없습니다. 다음과 같은 경우 승인 컨텍스트에 조건 컨텍스트 키가 없을 수 있습니다.

- 해당 조건 키를 지원하지 않는 리소스에 대한 요청에서 서비스별 조건 컨텍스트 키를 사용하는 경우
- IAM 보안 주체, 세션, 리소스 또는 요청의 태그가 없는 경우
- AWS 글로벌 조건 컨텍스트 키에 각 글로벌 조건 컨텍스트 키별로 나열된 기타 상황

IAM 정책의 조건 요소에 값이 없는 변수를 사용하는 경우, StringEquals 또는 StringLike와 같은 IAM JSON 정책 요소: 조건 연산자이(가) 일치하지 않고 정책 문이 적용되지 않습니다.

StringNotEquals 또는 StringNotLike와 같은 역 조건 연산자는 테스트 대상 조건 키의 값이 사실상 null 값과 같거나 비슷하지 않기 때문에 null 값에 대해 일치하지 않습니다.

다음 예에서는 액세스를 허용하려면 aws:principaltag/Team이 s3:ExistingObjectTag/Team과 같아야 합니다. aws:principaltag/Team이 설정되지 않은 경우 액세스가 명시적으로 거부됩니다. 승인 컨텍스트에 값이 없는 변수를 정책의 Resource 또는 NotResource 요소의 일부로 사용하는 경우, 값이 없는 정책 변수를 포함하는 리소스는 어떤 리소스와도 일치하지 않습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
   {
    "Effect": "Deny",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::/example-bucket/*",
    "Condition": {
      "StringNotEquals": {
        "s3:ExistingObjectTag/Team": "${aws:PrincipalTag/Team}"
       }
      }
    }
  ]
}
```

정책 변수로 사용할 수 있는 요청 정보

JSON 정책의 Condition 요소를 사용하여 요청 컨텍스트의 키를 정책에서 지정된 키 값과 비교할 수 있습니다. 정책 변수를 사용하는 경우 AWS은 정책의 변수 대신 요청 컨텍스트 키의 값을 대체합니다.

보안 주체 키 값

aws:username, aws:userid 및 aws:PrincipalType 값은 요청을 시작한 보안 주체 유형에 따라 다릅니다. 예를 들어 요청은 IAM 사용자, IAM 역할 또는 AWS 계정 루트 사용자의 자격 증명을 사용하여 가능합니다. 다음은 다른 유형의 보안 주체에 사용되는 키 값을 나타낸 목록입니다.

- AWS 계정 루트 사용자
  - aws:username: (없음)
  - aws:userid: AWS 계정 ID

  - `aws:PrincipalType`: `Account`
- IAM 사용자
  - `aws:username`: *`IAM-user-name`*
  - `aws:userid`: 고유 ID
  - `aws:PrincipalType`: `User`
- 페더레이션 사용자
  - `aws:username`: (없음)
  - `aws:userid`: *`account`*:*`caller-specified-name`*
  - `aws:PrincipalType`: `FederatedUser`
- 웹 페더레이션 사용자 및 SAML 페더레이션 사용자

> **Note**
>
> OIDC 페더레이션을 사용할 때 사용 가능한 정책 키에 대한 내용은 OIDC 페더레이션 섹션을 참조하세요.

  - `aws:username`: (없음)
  - `aws:userid`: (없음)
  - `aws:PrincipalType`: `AssumedRole`
- 위임된 역할
  - `aws:username`: (없음)
  - `aws:userid`: *`role-id`*:*`caller-specified-role-name`*
  - `aws:PrincipalType`: `Assumed role`
- Amazon EC2 인스턴스에 할당된 역할
  - `aws:username`: (없음)
  - `aws:userid`: *`role-id`*:*`ec2-instance-id`*
  - `aws:PrincipalType`: `Assumed role`
- 익명 호출자(Amazon SQS Amazon SNS 및 Amazon S3만 해당)
  - `aws:username`: (없음)
  - `aws:userid`: (없음)
  - `aws:PrincipalType`: `Anonymous`

이 목록에 있는 항목의 경우 다음을 참고하세요.

- 없음이란 현재 요청 정보에 값이 없다는 의미이며, 이때 일치시키려고 하면 실패하고 문이 잘못됩니다.
- *role-id*는 각 역할 생성 시 할당되는 고유 식별자입니다. AWS CLI 명령 `aws iam get-role --role-name` *rolename*으로 역할 ID를 표시할 수 있습니다.
- *caller-specified-name* 및 *caller-specified-role-name*은 임시 자격 증명을 가져오기 위해 호출할 때 호출 프로세스(예: 애플리케이션 또는 서비스 등)에서 전달되는 이름입니다.
- *ec2-instance-id*는 실행 시 인스턴스에 할당되는 값으로서 Amazon EC2 콘솔의 인스턴스 페이지에 표시됩니다. 또한 AWS CLI 명령 `aws ec2 describe-instances`를 실행하여 인스턴스 ID를 표시할 수도 있습니다.

페더레이션 사용자 요청에 사용할 수 있는 정보

페더레이션 사용자란 IAM 외에 다른 시스템을 사용하여 인증된 사용자를 말합니다. 예를 들어 AWS 호출 시 자체적으로 애플리케이션을 사용하는 회사가 있다고 가정하겠습니다. 이때는 회사의 애플리케이션 사용자 모두에게 IAM 자격 증명을 제공하는 것이 현실적으로 어렵습니다. 대신에 단일 IAM 자격 증명을 갖춘 프록시(미들 티어) 애플리케이션을 사용하거나, SAML 자격 증명 공급자(IdP)를 사용할 수 있습니다. 프록시 애플리케이션이나 SAML IdP는 회사 네트워크를 사용해 각 사용자를 인증합니다. 그런 다음 프록시 애플리케이션이 IAM 자격 증명을 사용하여 개별 사용자에 대한 임시 보안 자격 증명을 얻을 수 있습니다. SAML IdP는 AWS 임시 보안 자격 증명에 대한 ID 정보를 사실상 교환할 수 있습니다. 이후 임시 자격 증명을 사용하면 AWS 리소스에 액세스할 수 있습니다.

이와 유사한 방식으로 앱을 통해 AWS 리소스에 액세스해야 하는 모바일 디바이스용 앱을 개발하는 것도 가능합니다. 이 경우 앱이 Login with Amazon, Amazon Cognito, Facebook 또는 Google과 같은 잘 알려진 ID 공급자를 사용하여 사용자를 인증하는 OIDC 페더레이션을 사용할 수 있습니다. 인증이 완료되면 앱이 공급자의 사용자 인증 정보를 사용하여 임시 보안 자격 증명을 가져온 후 AWS 리소스에 액세스합니다.

OIDC 페더레이션을 위해 가장 바람직한 방법은 Amazon Cognito와 AWS 모바일 SDK를 이용하는 것입니다. 자세한 내용은 다음 자료를 참조하십시오.

- Amazon Cognito 사용 설명서
- 임시 자격 증명과 관련된 일반적인 시나리오

특수 문자

정책 변수 중에는 다른 특별한 의미를 갖는 문자를 나타낼 수 있도록 사전에 정의되어 있는 고정 값의 변수들도 몇 가지 있습니다. 이 특수 문자들은 일치시키려는 문자열의 일부이지만 리터럴로 삽입하였다면 오해할 가능성이 있습니다. 예를 들어 문자열에 별표(*)를 삽입하면 리터럴(*)이 아닌 모든 문자와 일치하는 와일드카드로 해석될 수 있습니다. 이 경우에는 다음과 같이 사전에 정의된 정책 변수를 사용할 수 있습니다.

- ${*} - *(별표)가 필요한 경우에 사용합니다.
- ${?} - ?(물음표)가 필요한 경우에 사용합니다.
- ${$} - $(달러 기호)가 필요한 경우에 사용합니다.

위처럼 사전 정의된 정책 변수들은 정규 정책 변수를 사용할 수 있는 문자열이라면 어디든지 사용 가능합니다.

기본값 지정하기

변수에 기본값을 추가하려면 기본값을 작은따옴표(' ')를 사용하고 변수 텍스트와 기본값을 쉼표와 공백(,)으로 묶습니다.

예를 들어, 보안 주체가 `team=yellow`(으)로 태그가 지정된 경우, `ExampleCorp's`(이)라는 이름의 `amzn-s3-demo-bucket-yellow` Amazon S3 버킷에 액세스할 수 있습니다. 이 리소스를 사용하는 정책을 사용하면 팀 구성원이 팀 버킷에 액세스할 수 있지만 다른 팀의 버킷에는 액세스할 수 없습니다. 팀 태그가 없는 사용자의 경우 버킷 이름의 기본값은 `company-wide`입니다. 이러한 사용자는 팀 참여에 대한 지침과 같은 광범위한 정보를 볼 수 있는 `amzn-s3-demo-bucket-company-wide` 버킷에만 액세스할 수 있습니다.

```
"Resource":"arn:aws:s3:::amzn-s3-demo-bucket-${aws:PrincipalTag/team, 'company-wide'}"
```

자세한 정보

정책에 대한 자세한 정보는 다음 섹션을 참조하세요.

- IAM의 정책 및 권한
- IAM 자격 증명 기반 정책의 예
- IAM JSON 정책 요소 참조
- 정책 평가 로직

- OIDC 페더레이션

IAM JSON 정책 요소: 지원되는 데이터 형식

이 단원에서는 JSON 정책에서 값을 지정할 때 지원되는 데이터 형식을 설명합니다. 정책 언어는 각 정책 요소마다 모든 형식을 지원하지 않기 때문에 각 요소에 대한 자세한 내용은 이전 섹션을 참조하세요.

- 문자열
- 숫자(정수 및 부동 소수점)
- 부울
- Null
- 목록
- 맵
- 구조(중첩 맵)

다음은 각 데이터 형식을 직렬화로 매핑한 표입니다. 모든 정책은 UTF-8 형식을 따라야 합니다. JSON 데이터 형식에 대한 자세한 내용은 RFC 4627에서 확인할 수 있습니다.

유형	JSON
문자열	문자열
Integer	숫자
Float	숫자
부울	true false
Null	null
날짜	ISO 8601의 W3C 프로파일을 준수하는 문자열
IpAddress	RFC 4632를 준수하는 문자열
목록	배열

유형	JSON
객체	객체

정책 평가 로직

보안 주체가 AWS Management Console, AWS API 또는 AWS CLI를 사용하려고 시도하면 해당 보안 주체가 요청을 AWS에 전송합니다. AWS 서비스가 요청을 받으면 AWS는 여러 단계를 완료하여 요청을 허용할지 거부할지 여부를 결정합니다.

1. 인증 - AWS는 먼저 필요하다면 요청을 생성하는 보안 주체를 인증합니다. 이 단계는 익명 사용자의 요청을 허용하는 Amazon S3와 같은 일부 서비스에서는 필요하지 않습니다.
2. 요청 컨텍스트 처리 – AWS는 요청에 담긴 내용을 처리하여 어떤 정책을 요청에 적용할지 결정합니다.
3. 단일 계정 내에서 정책 평가 – AWS는 정책의 평가 순서에 영향을 받는 모든 정책 유형을 평가합니다.
4. 계정 내에서 요청 허용 여부 결정 – AWS이때 는 요청에 따른 정책을 처리하여 요청을 허용할지 거부할지 여부를 결정합니다.

요청 컨텍스트 처리

AWS는 요청을 처리하여 다음 정보를 요청 콘텍스트에 모읍니다.

- 작업(또는 작동) - 보안 주체가 수행하고자 하는 작업 또는 작동입니다.
- 리소스 - 수행된 작업 또는 작동에 따른 AWS 리소스 객체입니다.
- 보안 주체 - 요청을 보내는 사용자, 역할, 페더레이션 사용자 또는 애플리케이션입니다. 보안 주체에 대한 정보는 보안 주체와 관련된 정책을 포함합니다.
- 환경 데이터 – IP 주소, 사용자 에이전트, SSL 사용 상태 또는 시간대와 같은 정보입니다.
- 리소스 데이터 – 요청되는 리소스와 관련된 데이터. 여기에는 DynamoDB 테이블 이름 또는 Amazon EC2 인스턴스 태그와 같은 정보가 포함될 수 있습니다.

AWS는 이러한 정보를 사용하여 요청 콘텍스트에 적용되는 정책을 찾습니다.

단일 계정 내에서 정책 평가

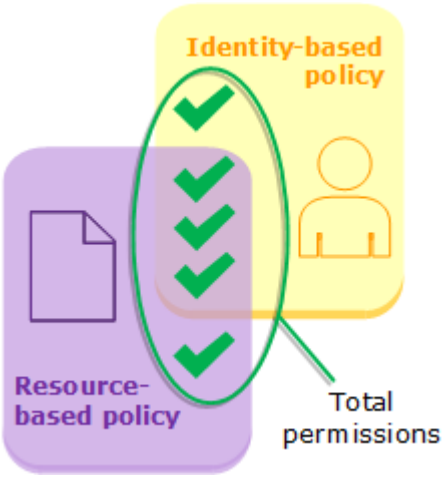
AWS는 요청 콘텍스트에 적용되는 정책 유형에 따라 정책을 평가합니다. 빈도순으로 나열된 다음 정책 유형을 단일 AWS 계정 내에서 사용할 수 있습니다. 이러한 정책 유형에 대한 자세한 정보는 IAM의 정책 및 권한 섹션을 참조하세요. AWS에서 크로스 계정 액세스에 대한 정책을 평가하는 방법에 대한 자세한 내용은 교차 계정 정책 평가 로직 섹션을 참조하세요.

1. 자격 증명 기반 정책 - 자격 증명 기반 정책은 IAM 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결되어 IAM 엔터티(사용자 및 역할)에 권한을 부여합니다. 자격 증명 기반 정책만 요청에 적용되는 경우 AWS에서는 하나 이상의 `Allow`에 대해 이러한 정책을 모두 확인합니다.
2. 리소스 기반 정책 - 리소스 기반 정책을 통해 보안 주체로서 지정된 보안 주체(계정, 사용자, 역할 및 역할 세션 및 IAM 페더레이션 사용자와 같은 세션 보안 주체)에 권한을 부여합니다. 권한은 보안 주체가 정책이 연결된 리소스를 사용하여 수행할 수 있는 작업을 정의합니다. 리소스 기반 정책 및 자격 증명 기반 정책 둘 다 요청에 적용되는 경우 AWS에서는 하나 이상의 `Allow`에 대해 이러한 정책을 모두 확인합니다. 리소스 기반 정책을 평가할 때 정책에 지정된 보안 주체 ARN은 다른 정책 유형의 암시적 거부가 최종 결정에 적용되는지 여부를 결정합니다.
3. IAM 권한 경계 - 권한 경계는 자격 증명 기반 정책을 통해 IAM 엔터티(사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 엔터티에 대한 권한 경계를 설정할 경우 해당 엔터티는 자격 증명 기반 정책 및 관련 권한 경계 모두에서 허용되는 작업만 수행할 수 있습니다. 경우에 따라, 권한 경계의 암시적 거부는 리소스 기반 정책에서 부여한 권한을 제한할 수 있습니다. 자세한 내용은 이 주제 뒷부분의 계정 내에서 요청 허용 여부 결정 섹션을 참조하세요.
4. AWS Organizations 서비스 제어 정책(SCP) - 조직 SCP는 조직 또는 조직 단위(OU)에 대한 최대 권한을 지정합니다. SCP 최대값은 각 AWS 계정 루트 사용자을 포함하여 멤버 계정의 보안 주체에 적용됩니다. SCP가 있는 경우 자격 증명 기반 및 리소스 기반 정책이 이러한 정책과 SCP에서 해당 작업을 허용하는 경우에 한해서만 멤버 계정의 보안 주체에게 권한을 부여합니다. 권한 경계와 SCP가 둘 다 있는 경우 권한 경계, SCP 및 자격 증명 기반 정책 모두에서 해당 작업을 허용해야 합니다.
5. 세션 정책 - 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 역할 세션을 프로그래밍 방식으로 생성하려면 `AssumeRole*` API 작업 중 하나를 사용합니다. 이를 수행하고 세션 정책을 전달할 때 결과적으로 얻는 세션의 권한은 IAM 엔터티의 자격 증명 기반 정책의 교차와 세션 정책입니다. 페더레이션 사용자 세션을 생성하려면 IAM 사용자 액세스 키를 사용하여 `GetFederationToken` API 작업을 프로그래밍 방식으로 호출합니다. 리소스 기반 정책에는 세션 정책 권한 평가에 대한 각기 다른 효과가 있습니다. 그 차이는 사용자 또는 역할의 ARN이나 세션의 ARN이 리소스 기반 정책의 보안 주체로 나열되는지 여부에 따라 다릅니다. 자세한 내용은 세션 정책 단원을 참조하십시오.

이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의함을 명심하세요.

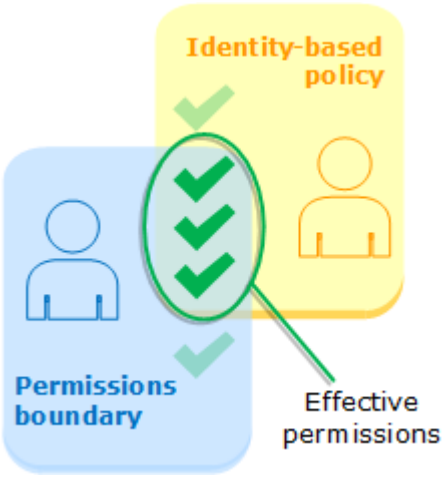
리소스 기반 정책과 함께 자격 증명 기반 정책 평가

자격 증명 기반 정책 및 리소스 기반 정책은 연결된 자격 증명이나 리소스에 권한을 부여합니다. IAM 엔터티(사용자 또는 역할)가 동일 계정 내에서 리소스에 대한 액세스를 요청할 경우 AWS는 자격 증명 기반 및 리소스 기반 정책을 통해 부여된 모든 권한을 평가합니다. 결과적으로 두 정책 유형의 모든 권한이 권한으로 부여됩니다. 자격 증명 기반 정책, 리소스 기반 정책 또는 두 정책 모두에 의해 작업이 허용되는 경우 AWS에서는 해당 작업을 허용합니다. 이들 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다.



권한 경계와 함께 자격 증명 기반 정책 평가

AWS에서 사용자의 자격 증명 기반 정책 및 권한 경계를 평가하는 경우 결과적으로 두 범주의 공통된 권한만 권한으로 부여됩니다. 기존 자격 증명 기반 정책으로 사용자에 권한 경계를 추가하면 사용자가 수행할 수 있는 작업을 축소할 수 있습니다. 또는 사용자에게서 권한 경계를 제거하면 사용자가 수행할 수 있는 작업이 늘어날 수 있습니다. 이들 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 다른 정책 유형을 권한 경계와 함께 평가하는 방식에 대해 자세히 알아보려면 경계가 있는 효과적인 권한 평가 섹션을 참조하세요.



조직 SCP와 함께 자격 증명 기반 정책 평가

사용자가 조직의 멤버인 계정에 속하는 경우 결과로 나온 권한은 사용자의 정책과 SCP의 교집합입니다. 즉, 자격 증명 기반 정책 및 SCP 모두에서 작업이 허용되어야 합니다. 이들 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다.

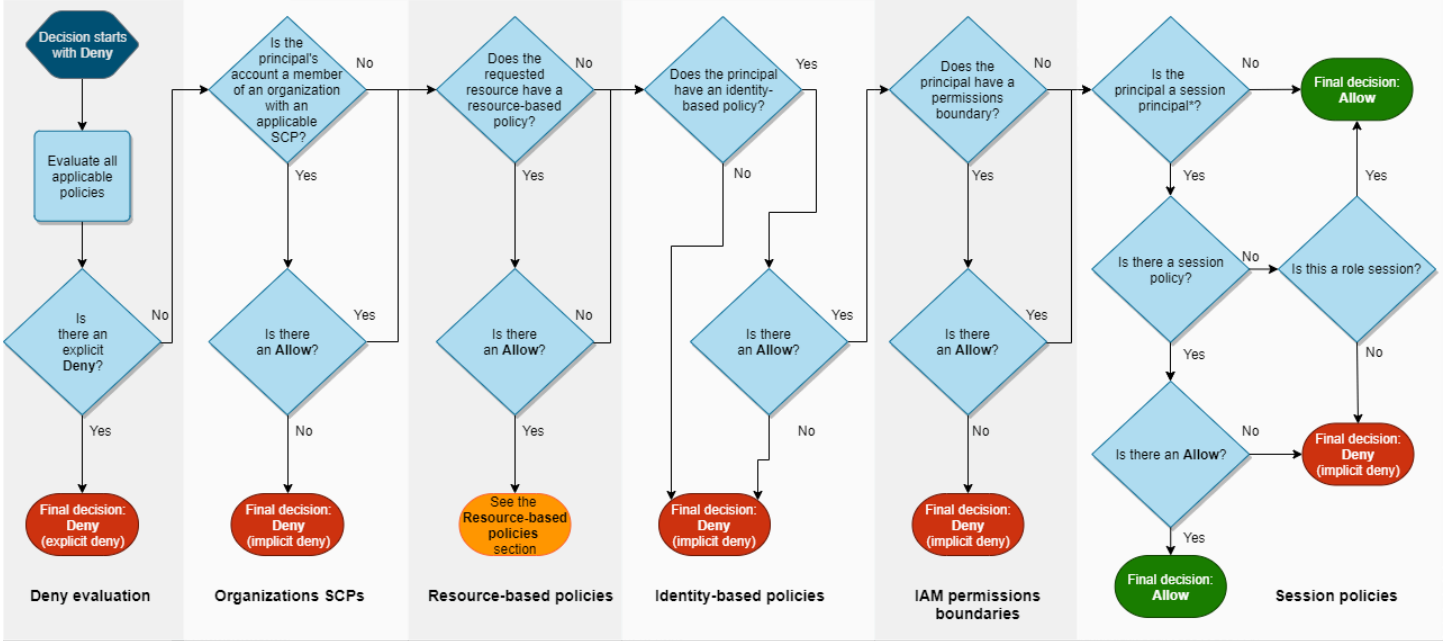
AWS Organizations에서 계정이 조직의 멤버인지 여부를 알아볼 수 있습니다. 조직 멤버가 SCP의 영향을 받을 수 있습니다. AWS CLI 명령 또는 AWS API 작업을 사용하여 이 데이터를 보려면 조직 엔터티에 대해 `organizations:DescribeOrganization` 작업 권한이 있어야 합니다. 조직 콘솔에서 작업을 수행할 추가 권한이 있어야 합니다. SCP가 특정 요청에 대한 액세스를 거부하는지 여부를 확인하거나 유효 권한을 변경하려면 AWS Organizations 관리자에게 문의하세요.

계정 내에서 요청 허용 여부 결정

보안 주체가 AWS로 요청을 보내 보안 주체의 엔터티와 동일한 계정에 있는 리소스에 액세스한다고 가정합니다. AWS 시행 코드는 요청의 허용 또는 거부 여부를 결정합니다. AWS에서는 요청 컨텍스트에 적용될 수 있는 모든 정책을 평가합니다. 다음은 단일 계정에 적용되는 이러한 정책에 대한 AWS 평가 논리를 요약한 것입니다.

- 기본적으로 모든 요청이 암시적으로 거부됩니다. 단, 전체 액세스 권한이 있는 AWS 계정 루트 사용자는 예외입니다.
- 자격 증명 기반 또는 리소스 기반 정책에 포함된 명시적 허용은 이 기본 작동을 재정의합니다.
- 권한 경계, 조직 SCP 또는 세션 정책이 있는 경우 이러한 정책 유형이 명시적 거부로 허용을 재정의할 수도 있습니다.
- 어떠한 정책의 명시적 거부도 허용을 무시합니다.

다음 순서도에 결정 방법에 대한 세부 정보가 나와 있습니다. 이 순서도는 리소스 기반 정책의 영향과 다른 유형의 정책에 대한 암시적 거부는 다루지 않습니다.



*A session principal is either a role session or an IAM federated user session.

1. 거부 평가 - 기본적으로 모든 요청이 거부됩니다. 이를 묵시적 거부라고 합니다. AWS 적용 코드는 해당 요청에 적용될 수 있는 계정 내의 모든 정책을 평가합니다. 여기에는 AWS Organizations SCP, 리소스 기반 정책, 자격 증명 기반 정책, IAM 권한 경계 및 세션 정책이 포함됩니다. 이런 모든 정책에서 적용 코드는 해당 요청에 적용되는 Deny 설명문을 찾습니다. 이를 명시적 거부라고 합니다. 적용되는 명시적 거부가 하나라도 발견되면 이 적용 코드는 최종 거부 결정을 반환합니다. 명시적 거부가 없으면 적용 코드 평가가 계속됩니다.
2. 조직 SCP - 그다음에는 적용 코드가 요청에 적용되는 AWS Organizations 서비스 제어 정책(SCP)을 평가합니다. SCP는 SCP가 연결된 계정의 보안 주체에 적용됩니다. 적용 가능한 `Allow` 문이 SCP에 없는 경우 거부가 암시적이더라도 요청이 명시적으로 거부됩니다. 적용 코드가 최종 거부 결정을 반환합니다. SCP가 없거나 요청한 작업이 SCP에서 허용된 경우 적용 코드 평가가 계속됩니다.
3. 리소스 기반 정책 - 동일한 계정 내에서 리소스 기반 정책은 리소스에 액세스하는 보안 주체의 유형과 리소스 기반 정책에서 허용되는 보안 주체에 따라 정책 평가에 다르게 영향을 줍니다. 보안 주체 유형에 따라 자격 증명 기반 정책, 권한 경계 또는 세션 정책에 암시적 거부가 있는 경우에도 리소스 기반 정책의 `Allow`은(는) `Allow`의 최종 결정을 내릴 수 있습니다.

   대부분 리소스의 경우, 액세스 권한을 부여하는 자격 증명 기반 정책 또는 리소스 기반 정책의 보안 주체에 대한 명시적인 허용만 필요합니다. IAM 역할 신뢰 정책과 KMS 키 정책은 보안 주체에 대한 액세스 권한을 명시적으로 허용해야 하므로 이 로직의 예외입니다.

지정한 보안 주체가 IAM 사용자, IAM 역할 또는 세션 보안 주체인 경우 리소스 기반 정책 로직이 기타 정책 유형과 다릅니다. 세션 보안 주체에는 IAM 역할 세션 또는 IAM 페더레이션 사용자 세션이 포함됩니다. 리소스 기반 정책이 요청을 수행하는 IAM 사용자 또는 세션 보안 주체에 직접 권한을 부여하는 경우, 자격 증명 기반 정책, 권한 경계 또는 세션 정책에서 암시적 거부가 최종 결정에 영향을 주지 않습니다.

다음 표에서는 자격 증명 기반 정책, 권한 경계 및 세션 정책에 암시적 거부가 있을 때 여러 보안 주체 유형에 대한 리소스 기반 정책의 영향을 이해하는 데 도움이 됩니다.

다음 표는 리소스 기반 정책과 동일한 계정의 다른 정책 유형의 암시적 거부를 보여줍니다.

요청하는 보안 주체	리소스 기반 정책	자격 증명 기반 정책	권한 경계	세션 정책	Result	이유
IAM 역할	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	역할 자체는 요청을 할 수 없습니다. 역할이 수임된 후 역할 세션에 대한 요청이 이루어집니다.

요청하는 보안 주체	리소스 기반 정책	자격 증명 기반 정책	권한 경계	세션 정책	Result	이유
IAM 역할 세션	역할 ARN 허용 또는 역할 세션 ARN 허용	암시적 거부	암시적 거부	암시적 거부	DENY - 역할 ARN 또는 ALLOW - 역할 세션 ARN	리소스 기반 정책의 보안 주체가 역할 ARN인 경우 권한 경계와 세션 정책은 최종 결정의 일부로 평가됩니다. 두 정책 중 하나에서 암시적 거부로 인해 DENY 결정이 발생합니다. 리소스 기반 정책의 보안 주체가 역할 세션 ARN인 경우 세션에 권한이 직접 부여됩니다. 다른 정책 유형은 결정에 영향을 주지 않습니다.

요청하는 보안 주체	리소스 기반 정책	자격 증명 기반 정책	권한 경계	세션 정책	Result	이유
IAM 사용자	IAM 사용자 ARN 허용	암시적 거부	암시적 거부	해당 사항 없음	ALLOW	권한은 사용자에게 직접 부여됩니다. 다른 정책 유형은 결정에 영향을 주지 않습니다.

요청하는 보안 주체	리소스 기반 정책	자격 증명 기반 정책	권한 경계	세션 정책	Result	이유
IAM 페더레이션 사용자 (`GetFederationToken`)	IAM 사용자 ARN 허용 또는 IAM 페더레이션 사용자 세션 ARN 허용	암시적 거부	암시적 거부	암시적 거부	DENY - IAM 사용자 ARN 또는 ALLOW - IAM 페더레이션 사용자 세션 ARN	리소스 기반 정책의 보안 주체가 IAM 사용자 ARN인 경우 권한 경계 또는 세션 정책의 암시적 거부로 인해 DENY가 발생합니다. 리소스 기반 정책의 보안 주체가 IAM 페더레이션 사용자 세션 ARN인 경우 세션에 권한이 직접 부여됩니다. 다른 정책 유형은 결정에 영향을 주지 않습니다.

요청하는 보안 주체	리소스 기반 정책	자격 증명 기반 정책	권한 경계	세션 정책	Result	이유
루트 사용자	루트 사용자 ARN 허용	해당 사항 없음	해당 사항 없음	해당 사항 없음	ALLOW	루트 사용자는 AWS 계정의 모든 리소스에는 완전한 무제한 액세스 권한이 있습니다. AWS Organizations의 계정 루트 사용자에 대한 액세스를 제어하는 방법을 알아보려면 조직 사용 설명서의 서비스 제어 정책(SCP)을 참조하세요.

요청하는 보안 주체	리소스 기반 정책	자격 증명 기반 정책	권한 경계	세션 정책	Result	이유
AWS 서비스 주체	AWS서비스 보안 주체 허용	해당 사항 없음	해당 사항 없음	해당 사항 없음	ALLOW	리소스 기반 정책이 AWS 서비스 보안 주체에게 직접 권한을 부여하면 다른 정책 유형은 결정에 영향을 주지 않습니다.

- IAM 역할 - IAM 역할 ARN에 권한을 부여하는 리소스 기반 정책은 권한 경계 또는 세션 정책의 암시적 거부에 의해 제한됩니다. Principal 요소나 `aws:PrincipalArn` 조건 키에 역할 ARN을 지정할 수 있습니다. 두 경우 모두 요청을 하는 보안 주체는 IAM 역할 세션입니다.

  자격 증명 기반 정책에 명시적 거부가 포함되지 않는 한, 권한 경계와 세션 정책은 Principal 요소에서 와일드카드(*)와 함께 `aws:PrincipalArn` 조건 키를 사용하여 부여한 권한을 제한하지 않습니다. 자세한 내용은 IAM 역할 보안 주체 단원을 참조하십시오.

  역할 ARN 예

  ```
  arn:aws:iam::111122223333:role/examplerole
  ```

- IAM 역할 세션 - 동일한 계정 내에서 IAM 역할 세션 ARN에 권한을 부여하는 리소스 기반 정책은 수임된 역할 세션에 직접 권한을 부여합니다. 세션에 직접 부여된 사용 권한은 자격 증명 기반 정책, 권한 경계 또는 세션 정책의 암시적 거부에 의해 제한되지 않습니다. 역할을 맡고 요청을 할 때 요청을 수행하는 보안 주체는 역할 자체의 ARN이 아니라 IAM 역할 세션 ARN입니다. 자세한 내용은 역할 세션 보안 주체 단원을 참조하십시오.

  역할 세션 ARN 예

  ```
  arn:aws:sts::111122223333:assumed-role/examplerole/examplerolesessionname
  ```

- IAM 사용자 - 동일한 계정 내에서 IAM 사용자 ARN(페더레이션 사용자 세션이 아님)에게 권한을 부여하는 리소스 기반 정책은 자격 증명 기반 정책 또는 권한 경계에서 암시적 거부에 의해 제한되지 않습니다.

  IAM 사용자 ARN 예

  ```
  arn:aws:iam::111122223333:user/exampleuser
  ```

- IAM 페더레이션 사용자 세션 - IAM 페더레이션 사용자 세션은 GetFederationToken 호출을 통해 생성된 세션입니다. 페더레이션 사용자가 요청을 할 때 요청을 수행하는 보안 주체는 페더레이션된 IAM 사용자의 ARN이 아니라 페더레이션 사용자 ARN입니다. 동일한 계정 내에서 페더레이션 사용자 ARN에게 권한을 부여하는 리소스 기반 정책은 세션에 직접 권한을 부여합니다. 세션에 직접 부여된 사용 권한은 자격 증명 기반 정책, 권한 경계 또는 세션 정책의 암시적 거부에 의해 제한되지 않습니다.

  그러나 리소스 기반 정책이 페더레이션한 IAM 사용자의 ARN에 권한을 부여하는 경우, 세션 중에 페더레이션 사용자가 요청한 요청은 권한 경계 또는 세션 정책의 암시적 거부에 의해 제한됩니다.

  IAM 페더레이션 사용자 세션 ARN 예

  ```
  arn:aws:sts::111122223333:federated-user/exampleuser
  ```

4. 자격 증명 기반 정책 - 그런 다음, 적용 코드는 보안 주체에 대한 자격 증명 기반 정책을 확인합니다. IAM 사용자의 경우 이러한 정책에는 사용자 정책과 사용자가 속한 그룹의 정책이 포함됩니다. 자격 증명 기반 정책이 없거나 요청된 작업을 허용하는 자격 증명 기반 정책에 대한 설명이 없는 경우, 적용 코드는 최종 Deny(거부) 결정을 반환합니다. 적용 가능한 자격 증명 기반 정책에서 요청한 작업을 허용하는 설명문이 있는 경우, 코드는 유지됩니다.
5. IAM 권한 경계 - 다음에는 코드가 보안 주체에 사용되는 IAM 엔터티에 권한 경계가 지정되어 있는지 여부를 확인합니다. 권한 경계를 설정하는 데 사용되는 정책에서 요청한 작업을 허용하지 않는 경우 요청이 묵시적으로 거부됩니다. 적용 코드가 최종 거부 결정을 반환합니다. 권한 경계가 없거나 요청한 작업이 권한 경계에서 허용된 경우 코드 실행이 계속됩니다.
6. 세션 정책 - 코드에서는 그런 다음에 보안 주체가 세션 보안 주체인지 확인합니다. 세션 보안 주체에는 IAM 역할 세션 또는 IAM 페더레이션 사용자 세션이 포함됩니다. 보안 주체가 세션 보안 주체가 아닌 경우 적용 코드는 허용 최종 결정을 반환합니다.

   세션 보안 주체의 경우 코드는 요청에 세션 정책이 전달되었는지 여부를 확인합니다. AWS CLI 또는 AWS API를 사용하는 동안 세션 정책을 전달하여 역할이나 IAM 페더레이션 사용자에 대한 임시 자격 증명을 가져올 수 있습니다.

- 세션 정책이 있지만 요청한 작업이 세션 정책에서 허용되지 않는 경우 해당 요청이 암시적으로 거부됩니다. 적용 코드가 최종 거부 결정을 반환합니다.
- 세션 정책이 없는 경우 코드는 보안 주체가 역할 세션인지 여부를 확인합니다. 보안 주체가 역할 세션인 경우 요청은 Allow(허용됨)입니다. 그렇지 않으면, 요청은 암시적으로 거부되며 코드에서는 Deny(거부)의 최종 결정을 반환합니다.
- 세션 정책이 있고 요청한 작업을 허용한 경우, 적용 코드에서는 Allow(허용)의 최종 결정을 반환합니다.

7. 오류 - AWS 적용 코드를 평가하는 도중 오류가 발생할 경우 코드는 예외를 생성한 후 닫힙니다.

자격 증명 기반 정책 및 리소스 기반 정책 평가 예제

가장 일반적인 정책 유형은 자격 증명 정책 및 리소스 기반 정책입니다. 리소스에 대한 액세스가 요청되면 AWS는 동일한 계정 내에서 하나 이상의 Allow에 대해 정책에서 부여한 모든 권한을 평가합니다. 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다.

> **Important**
>
> 동일한 계정 내의 ID 기반 정책이나 리소스 기반 정책 중 하나는 요청을 허용하고 다른 하나는 허용하지 않는 경우에도 요청은 계속 허용됩니다.

Carlos가 `carlossalazar`라는 사용자 이름을 쓰고 있고 `carlossalazar-logs` Amazon S3 버킷에 파일을 저장하고자 한다고 가정합니다.

또한 다음 정책이 `carlossalazar` IAM 사용자와 연결되었다고 가정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowS3ListRead",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
```

```
        },
        {
            "Sid": "AllowS3Self",
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::carlossalazar/*",
                "arn:aws:s3:::carlossalazar"
            ]
        },
        {
            "Sid": "DenyS3Logs",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::*log*"
        }
    ]
}
```

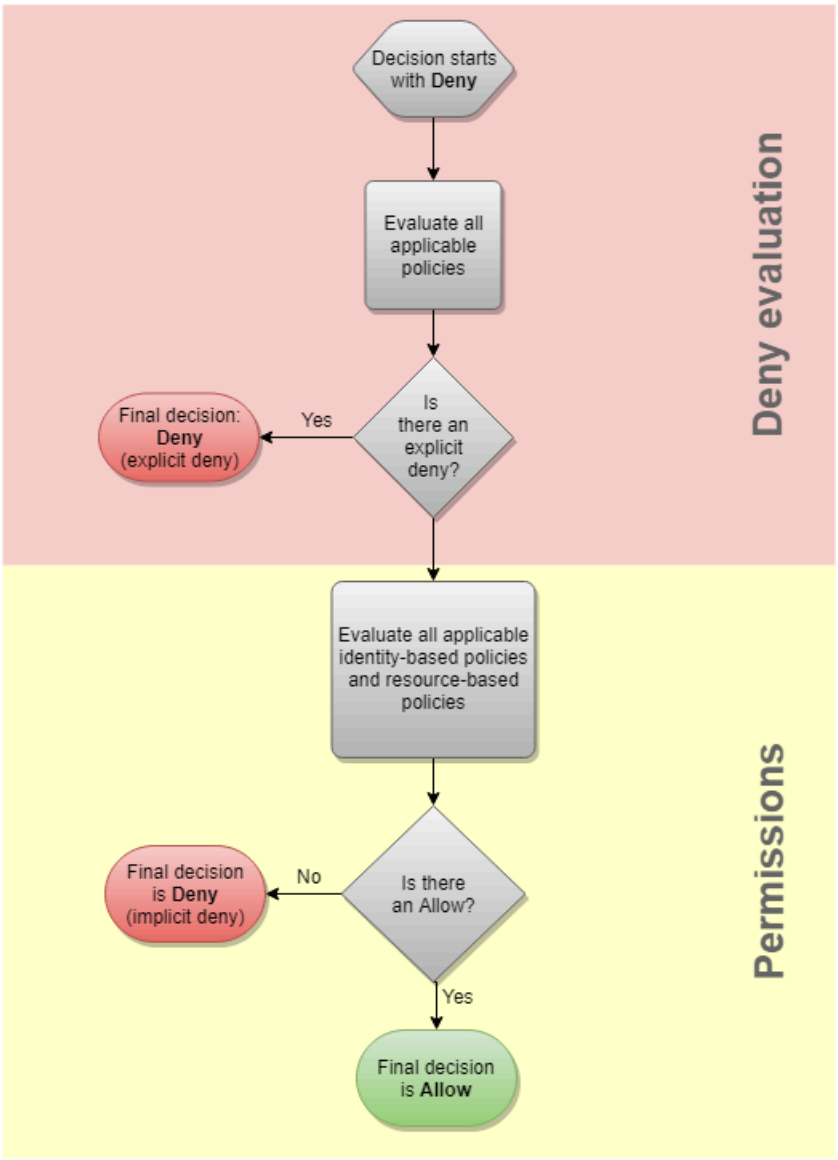
이 정책의 AllowS3ListRead 설명문은 카를로스가 계정에 있는 모든 버킷 목록을 보도록 허용합니다. AllowS3Self 설명문은 카를로스가 그의 사용자 이름과 동일한 버킷에 모두 액세스할 수 있도록 허용합니다. DenyS3Logs 설명문은 카를로스가 그의 이름 아래에 있는 log를 통해 모든 S3 버킷의 액세스를 거부합니다.

또한, 다음 리소스 기반 정책(버킷 정책이라고 함)은 carlossalazar 버킷에 연결됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/carlossalazar"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::carlossalazar/*",
                "arn:aws:s3:::carlossalazar"
            ]
        }
    ]
}
```

이 정책은 carlossalazar 사용자만 carlossalazar 버킷에 액세스할 수 있도록 지정합니다.

카를로스가 carlossalazar-logs 버킷에 파일을 저장하도록 요청하면 AWS는 해당 요청에 어떤 정책을 적용할지 결정합니다. 이 경우, 자격 증명 기반 정책과 리소스 기반 정책만 적용합니다. 이들은 모두 권한 정책입니다. 어떠한 권한 경계도 적용되지 않기 때문에 평가 로직은 다음 로직으로 줄어듭니다.



AWS는 먼저 요청 콘텍스트에 적용되는 Deny 설명문을 확인합니다. 자격 증명 기반 정책은 카를로스의 로깅을 통한 모든 S3 버킷의 액세스를 명시적으로 거부하기 때문에 이를 찾습니다. 카를로스의 액세스가 거부됩니다.

Carlos가 실수를 알아차리고 carlossalazar 버킷에 파일을 저장하고자 한다고 가정하세요. AWS는 Deny 설명문을 확인하지만 찾지 못합니다. 그러면 권한 정책을 확인합니다. 자격 증명 기반 정책과 리

소스 기반 정책 모두 요청을 허용합니다. 따라서 AWS는 요청을 허용합니다. 이들 중 하나라도 설명문을 명시적으로 거부한다면 요청은 거부됩니다. 정책 유형 중 하나는 요청을 허용하고 다른 하나는 요청을 허용하지 않는 경우에도 요청은 허용됩니다.

명시적 거부와 묵시적 거부 차이

적용 가능한 정책이 Deny 설명문을 포함한다면 요청은 명시적으로 거부됩니다. 정책이 Allow 설명문과 Deny 설명문을 포함한 요청에 적용된다면 Deny 설명문은 Allow 설명문에 우선합니다. 이 요청은 명시적으로 거부됩니다.

적용 가능한 Deny 설명문이 없고 적용 가능한 Allow 설명문도 없다면 묵시적 거부가 발생합니다. IAM 보안 주체가 기본적으로 액세스를 거부하기 때문에 명시적으로 작업을 허용해야 합니다. 그렇지 않으면 액세스는 묵시적으로 거부됩니다.

권한 부여 전략을 설계한다면 Allow 설명문으로 정책을 생성하여 보안 주체가 성공적으로 요청하도록 허용합니다. 그러나 명시적 또는 묵시적 거부 조합을 선택할 수 있습니다.

예를 들어, 허용되는 작업, 암시적으로 거부된 작업 및 명시적으로 거부된 작업을 포함하는 다음 정책을 생성할 수 있습니다. AllowGetList 설명문은 접두사 Get 및 List(으)로 시작하는 IAM 작업에 대한 읽기 전용 액세스를 허용합니다. iam:CreatePolicy와(과) 같은 IAM의 다른 모든 작업은 암시적으로 거부됩니다. DenyReports 설명문은 iam:GetOrganizationsAccessReport와 같이 Report 접미사가 포함된 작업에 대한 액세스를 거부하여 IAM 보고서에 대한 액세스를 명시적으로 거부합니다. 누군가가 이 보안 주체에 다른 정책을 추가하여 iam:GenerateCredentialReport와 같은 IAM 보고서에 대한 액세스 권한을 부여하는 경우, 보고서 관련 요청은 이 명시적 거부로 인해 계속 거부됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowGetList",
            "Effect": "Allow",
            "Action": [
                "iam:Get*",
                "iam:List*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DenyReports",
```

```
            "Effect": "Deny",
            "Action": "iam:*Report",
            "Resource": "*"
        }
    ]
}
```

교차 계정 정책 평가 로직

한 계정의 보안 주체가 두 번째 계정의 리소스에 액세스하도록 허용할 수 있습니다. 이를 크로스 계정 액세스라고 합니다. 교차 계정 액세스를 허용할 때 보안 주체가 존재하는 계정을 신뢰할 수 있는 계정이라고 합니다. 리소스가 존재하는 계정을 신뢰하는 계정이라고 합니다.

교차 계정 액세스를 허용하려면 공유하려는 리소스에 리소스 기반 정책을 연결해야 합니다. 또한 요청에서 보안 주체 역할을 하는 아이덴티티에 아이덴티티 기반 정책을 연결해야 합니다. 신뢰하는 계정의 리소스 기반 정책은 리소스에 액세스할 수 있는 신뢰할 수 있는 계정의 보안 주체를 지정해야 합니다. 전체 계정이나 해당 IAM 사용자, 페더레이션 사용자, IAM 역할 또는 위임된 역할 세션을 지정할 수 있습니다. AWS 서비스를 보안 주체로 지정할 수도 있습니다. 자세한 내용은 보안 주체 지정 섹션을 참조하세요.

보안 주체의 자격 증명 기반 정책은 신뢰 서비스의 리소스에 대한 요청된 액세스를 허용해야 합니다. 리소스의 ARN을 지정하거나 모든 리소스에 대한 액세스를 허용하여 이 작업을 수행할 수 있습니다(*).

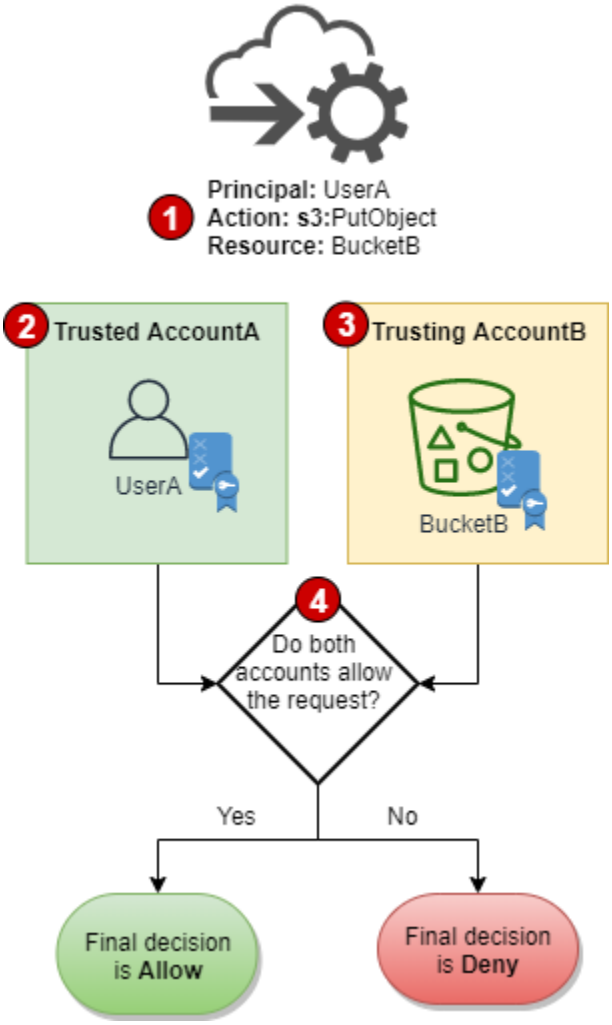
IAM에서는 리소스 기반 정책을 IAM 역할에 연결하여 다른 계정의 보안 주체가 해당 역할을 수임하도록 허용할 수 있습니다. 역할의 리소스 기반 정책을 역할 신뢰 정책이라고 합니다. 이 역할을 가정하면 허용된 보안 주체는 결과로 생성되는 임시 자격 증명을 사용하여 계정의 여러 리소스에 액세스할 수 있습니다. 이러한 액세스 권한은 역할의 자격 증명 기반 권한 정책에 정의되어 있습니다. 역할을 사용하여 교차 계정 액세스를 허용하는 것과 다른 리소스 기반 정책을 사용하여 교차 계정 액세스를 허용하는 것이 어떻게 다른지 알아보려면 IAM의 크로스 계정 리소스 액세스 섹션을 참조하세요.

> **Important**
>
> 다른 서비스는 정책 평가 로직에 영향을 줄 수 있습니다. 예를 들어 AWS Organizations에서는 하나 이상의 보안 주체 계정에 서비스 제어 정책을 적용할 수 있도록 지원합니다. AWS Resource Access Manager에서는 보안 주체가 공유되는 리소스에서 수행할 수 있는 작업을 제어하는 정책 조각을 지원합니다.

교차 계정 요청의 허용 여부 결정

교차 계정 요청의 경우 신뢰할 수 있는 AccountA의 요청자가 자격 증명 기반 정책을 가지고 있어야 합니다. 이 정책은 신뢰하는 AccountB에서 리소스에 대한 요청을 생성할 수 있도록 허용해야 합니다. 또한 AccountB의 리소스 기반 정책은 AccountA의 요청자가 리소스에 액세스할 수 있도록 허용해야 합니다.

사용자가 교차 계정 요청을 하면 AWS에서는 두 가지 평가를 수행합니다. AWS는 신뢰하는 계정 및 신뢰할 수 있는 계정에서 요청을 평가합니다. 단일 계정 내에서 요청을 평가하는 방법에 대한 자세한 내용은 계정 내에서 요청 허용 여부 결정 섹션을 참조하세요. 두 평가에서 모두 Allow라는 결정을 반환하는 경우에만 요청이 허용됩니다.



1. 한 계정의 보안 주체가 다른 계정의 리소스에 액세스하도록 요청하는 경우 이는 교차 계정 요청입니다.

2. 요청한 보안 주체는 신뢰할 수 있는 계정(AccountA)에 존재합니다. AWS는 이 계정을 평가할 때 자격 증명 기반 정책 및 자격 증명 기반 정책을 제한할 수 있는 정책을 확인합니다. 자세한 내용은 단일 계정 내에서 정책 평가 섹션을 참조하세요.
3. 요청된 리소스가 신뢰하는 계정(AccountB)에 존재합니다. AWS는 이 계정을 평가할 때 요청된 리소스에 연결된 리소스 기반 정책과 리소스 기반 정책을 제한할 수 있는 정책을 확인합니다. 자세한 내용은 단일 계정 내에서 정책 평가 섹션을 참조하세요.
4. AWS에서는 두 계정 정책 평가에서 모두 요청을 허용하는 경우에만 요청을 허용합니다.

교차 계정 정책 평가의 예

다음 예제에서는 한 계정의 사용자에게 두 번째 계정의 리소스 기반 정책에 의해 권한이 부여되는 시나리오를 보여 줍니다.

Carlos가 계정 111111111111에 carlossalazar라는 이름의 IAM 사용자가 있는 개발자라고 가정합니다. 그는 계정 222222222222에 있는 Production-logs Amazon S3 버킷에 파일을 저장하려고 합니다.

또한 다음 정책이 carlossalazar IAM 사용자와 연결되었다고 가정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowS3ListRead",
            "Effect": "Allow",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "*"
        },
        {
            "Sid": "AllowS3ProductionObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object*",
            "Resource": "arn:aws:s3:::Production/*"
        },
        {
            "Sid": "DenyS3Logs",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::*log*",
                "arn:aws:s3:::*log*/*"
```

```
            ]
        }
    ]
}
```

이 정책의 AllowS3ListRead 문은 Carlos가 Amazon S3에 있는 모든 버킷의 목록을 보도록 허용합니다. AllowS3ProductionObjectActions 문은 Carlos에게 Production 버킷의 객체에 대한 전체 액세스를 허용합니다. DenyS3Logs 설명문은 카를로스가 그의 이름 아래에 있는 log를 통해 모든 S3 버킷의 액세스를 거부합니다. 또한 해당 버킷의 모든 객체에 대한 액세스를 거부합니다.

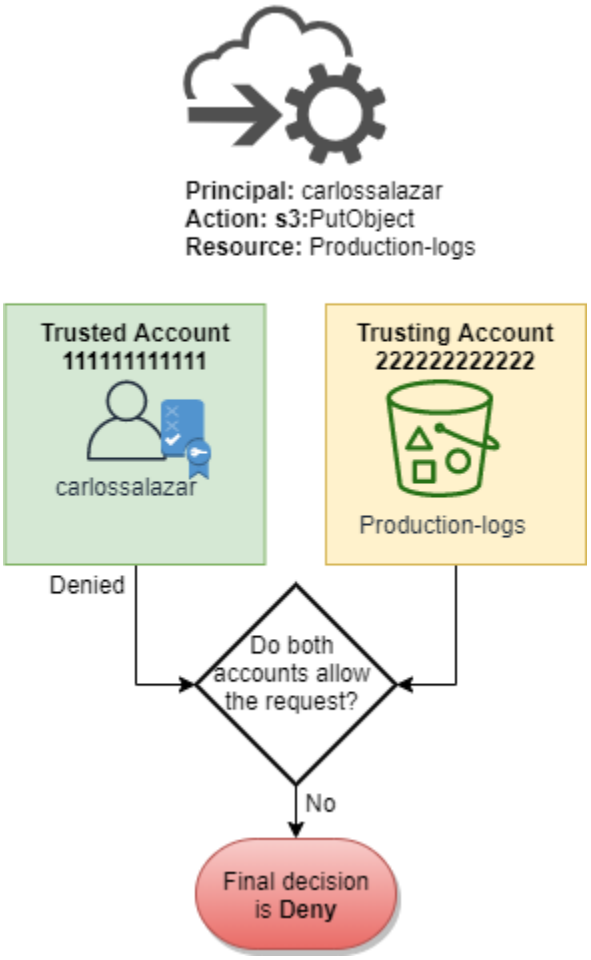
또한, 다음 리소스 기반 정책(버킷 정책이라고 함)은 계정 222222222222의 Production 버킷에 연결됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:PutObject*",
                "s3:ReplicateObject",
                "s3:RestoreObject"
            ],
            "Principal": { "AWS": "arn:aws:iam::111111111111:user/carlossalazar" },
            "Resource": "arn:aws:s3:::Production/*"
        }
    ]
}
```

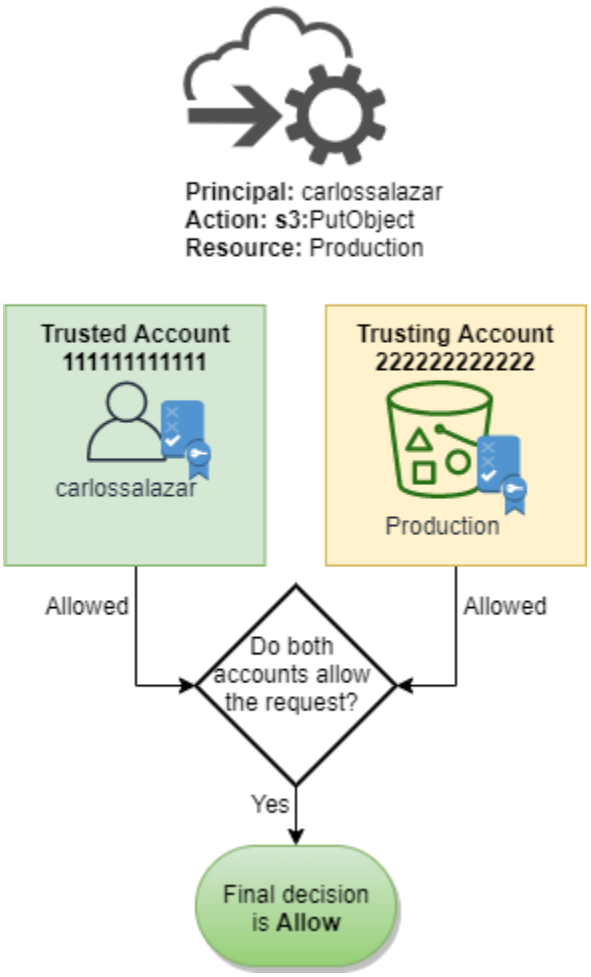
이 정책은 carlossalazar 사용자에게 Production 버킷의 객체에 대한 액세스를 허용합니다. 그는 버킷의 객체를 생성하고 편집할 수는 있지만 삭제할 수는 없습니다. 그는 버킷 자체를 관리할 수 없습니다.

카를로스가 Production-logs 버킷에 파일을 저장하도록 요청하면 AWS는 해당 요청에 어떤 정책을 적용할지 결정합니다. 이 경우 carlossalazar 사용자에게 연결된 자격 증명 기반 정책이 계정 111111111111에 적용되는 유일한 정책입니다. 계정 222222222222에서는 Production-logs 버킷에 연결된 리소스 기반 정책이 없습니다. AWS은 계정 111111111111을 평가할 때 Deny의 결정을 반환합니다. 이는 자격 증명 기반 정책의 DenyS3Logs 문이 모든 로그 버킷에 대한 액세스를 명시적으로 거부하기 때문입니다. 단일 계정 내에서 요청을 평가하는 방법에 대한 자세한 내용은 계정 내에서 요청 허용 여부 결정 섹션을 참조하세요.

요청이 계정 중 하나에서 명시적으로 거부되기 때문에 최종 결정은 요청을 거부하는 것입니다.



카를로스가 자신의 실수를 깨닫고 `Production` 버킷에 파일을 저장하려고 시도한다고 가정합니다. AWS는 먼저 계정 111111111111을 확인하여 요청이 허용되는지 여부를 확인합니다. 자격 증명 기반 정책만 적용되며 요청을 허용합니다. 그리고 AWS는 계정 222222222222를 확인합니다. `Production` 버킷에 연결된 리소스 기반 정책만 적용되며 요청을 허용합니다. 두 계정 모두 요청을 허용하므로 최종 결정은 요청을 허용하는 것입니다.



IAM JSON 정책 언어의 문법

이 페이지에서는 IAM에서 JSON 정책 생성 시 사용되는 언어의 정규 문법에 대해 살펴보겠습니다. 이 문법에 대해 살펴본 후 정책의 체계적 작성 및 검증 방법에 대해 이해할 수 있게 될 것입니다.

정책 예는 다음 주제를 참조하세요.

- IAM의 정책 및 권한
- IAM 자격 증명 기반 정책의 예
- Amazon EC2 사용 설명서의 Amazon EC2 콘솔 작업을 위한 예제 정책 및 AWS CLI, Amazon EC2 CLI 또는 AWS SDK 작업을 위한 예제 정책
- Amazon Simple Storage Service 사용 설명서의 버킷 정책의 예제 및 사용자 정책 예

다른 AWS 서비스의 정책 예는 해당 서비스 설명서를 참조하세요.

주제

정책 언어 및 JSON

정책은 JSON으로 작성됩니다. JSON 정책을 생성하거나 편집할 때 IAM은 효과적인 정책을 생성하는 데 도움이 되는 정책 검증을 수행할 수 있습니다. IAM은 JSON 구문 오류를 식별하는 반면, IAM Access Analyzer는 정책을 더욱 구체화하는 데 도움이 되는 권장 사항과 함께 추가 정책 검사를 제공합니다. 정책 검증에 대한 자세한 내용은 IAM 정책 검증 섹션을 참조하세요. IAM Access Analyzer 정책 확인 및 실행 가능한 권장 사항에 대한 자세한 내용은 IAM Access Analyzer 정책 검증을 참조하세요.

여기에서는 유효한 JSON 구성에 대해 자세히 설명하지는 않지만 다음과 같이 몇 가지 기본 JSON 규칙을 소개합니다.

- 각 개체 간에 공백을 넣을 수 있습니다.
- 값은 인용 부호로 묶입니다. 숫자나 부울(Boolean) 값에서 인용 부호는 옵션입니다.
- 대부분 요소(예: `action_string_list`, `resource_string_list`)는 JSON 배열을 값으로 사용할 수 있습니다. 배열은 하나 이상의 값을 갖습니다. 값이 2개 이상 추가되면 배열은 다음 예제와 같이 대괄호(`[` 및 `]`)로 묶여 쉼표로 구분됩니다.

  ```
  "Action" : ["ec2:Describe*","ec2:List*"]
  ```
- 기본 JSON 데이터 형식(부울, 숫자, 문자열)은 RFC 7159에 정의되어 있습니다.

JSON 문법에 사용되는 규칙

JSON 문법에는 다음과 같은 규칙이 사용됩니다.

- 다음 문자는 JSON 토큰으로서 정책에 추가됩니다.

  ```
  { } [ ] " , :
  ```
- 다음은 문법에 사용되는 특수 문자로서 정책에는 추가되지 않습니다.

  ```
  = < > ( ) |
  ```
- 한 요소에 여러 값을 추가할 수 있는 경우에는 반복되는 값, 쉼표 구분자, 그리고 줄임표(`...`)를 사용하여 나타냅니다. 예:

```
[<action_string>, <action_string>, ...]

<principal_map> = { <principal_map_entry>, <principal_map_entry>, ... }
```

여러 값이 허용되면 단일 값을 추가하는 것도 유효합니다. 값이 단 하나인 경우에는 마지막 쉼표를 반드시 생략해야 합니다. 요소가 배열([ ]로 표시)로 이루어지더라도 추가된 값이 단 하나일 때는 괄호가 선택 사항입니다. 예:

```
"Action": [<action_string>]

"Action": <action_string>
```

- 요소 뒤에 나오는 물음표(?)는 요소가 선택 사항인 것을 나타냅니다. 예제

  ```
  <version_block?>
  ```

  하지만 선택 요소에 대한 자세한 내용은 문법 목록 이후에 나오는 참고 사항을 반드시 확인하시기 바랍니다.
- 요소 사이의 수직선(|)은 다자간 택일을 나타냅니다. 이 문법에서는 괄호로 다자간 택일의 범위를 정의합니다. 예제

  ```
  ("Principal" | "NotPrincipal")
  ```
- 리터럴 문자열 요소는 큰따옴표(")로 묶습니다. 예제

  ```
  <version_block> = "Version" : ("2008-10-17" | "2012-10-17")
  ```

기타 참고 사항은 문법 목록 다음 정책 문법 참고 사항 섹션을 참조하세요.

문법

다음 목록은 정책 언어 문법에 대한 설명입니다. 문법 목록에 사용된 규칙에 대해서는 앞의 섹션을 참조하세요. 그리고, 추가 정보는 이후 참고 사항을 참조하세요.

> **Note**
>
> 이 문법은 버전이 `2008-10-17` 및 `2012-10-17`이라고 표시된 정책에 대한 설명입니다. `Version` 정책 요소는 정책 버전과 다릅니다. `Version` 정책 요소는 정책 내에서 사용되며 정책 언어의 버전을 정의합니다. 반면에 정책 버전은 IAM에서 고객 관리형 정책을 변경할 때 생성됩니다. 변경된 정책은 기존 정책을 덮어쓰지 않습니다. 대신 IAM에서 관리형 정책의 새 버전을 생성합니다. `Version` 정책 요소에 대한 자세한 정보는 IAM JSON 정책 요소: Version을

참조하세요. 정책 버전에 대한 자세한 정보는 the section called “IAM 정책 버전 관리” 섹션을 참조하세요.

```
policy  = {
     <version_block?>
     <id_block?>
     <statement_block>
}

<version_block> = "Version" : ("2008-10-17" | "2012-10-17")

<id_block> = "Id" : <policy_id_string>

<statement_block> = "Statement" : [ <statement>, <statement>, ... ]

<statement> = {
    <sid_block?>,
    <principal_block?>,
    <effect_block>,
    <action_block>,
    <resource_block>,
    <condition_block?>
}

<sid_block> = "Sid" : <sid_string>

<effect_block> = "Effect" : ("Allow" | "Deny")

<principal_block> = ("Principal" | "NotPrincipal") : ("*" | <principal_map>)

<principal_map> = { <principal_map_entry>, <principal_map_entry>, ... }

<principal_map_entry> = ("AWS" | "Federated" | "Service" | "CanonicalUser") :
    [<principal_id_string>, <principal_id_string>, ...]

<action_block> = ("Action" | "NotAction") :
    ("*" | [<action_string>, <action_string>, ...])

<resource_block> = ("Resource" | "NotResource") :
    : ("*" | <resource_string> | [<resource_string>, <resource_string>, ...])

<condition_block> = "Condition" : { <condition_map> }
```

```
<condition_map> = {
  <condition_type_string> : { <condition_key_string> : <condition_value_list> },
  <condition_type_string> : { <condition_key_string> : <condition_value_list> }, ...
}
<condition_value_list> = [<condition_value>, <condition_value>, ...]
<condition_value> = (<condition_value_string> | <condition_value_string> |
 <condition_value_string>)
```

정책 문법 참고 사항

- 단일 정책에는 다수의 문이 배열로 추가될 수 있습니다.
- 정책은 추가되는 개체에 따라 2,048~10,240 사이에서 최대 문자 수를 갖습니다. 자세한 내용은 IAM 및 AWS STS 할당량 단원을 참조하십시오. 정책 크기 계산에 공백 문자는 포함되지 않습니다.
- 개별 요소에는 동일한 키 인스턴스를 여러 개 추가할 수 없습니다. 예를 들어 동일한 문에 `Effect` 블록을 2개 추가할 수는 없습니다.
- 블록은 순서에 상관없이 표시됩니다. 예를 들어 정책에서 `version_block`은 `id_block` 뒤에 올 수 있습니다. 마찬가지로 `effect_block`, `principal_block` 및 `action_block` 역시 동일 문에서 순서에 상관없이 표시됩니다.
- 리소스 기반 정책에서는 `id_block`이 선택 사항입니다. ID 기반 정책에는 포함시킬 수 없습니다.
- `principal_block` 요소는 리소스 기반 정책(예: Amazon S3 버킷 정책)과 IAM 역할의 신뢰 정책에 필요합니다. ID 기반 정책에는 포함시킬 수 없습니다.
- Amazon S3 버킷 정책의 `principal_map` 요소에는 `CanonicalUser` ID가 포함될 수 있습니다. 대부분 리소스 기반 정책은 이러한 매핑을 지원하지 않습니다. 버킷 정책에서 표준 사용자 ID를 사용하는 방법에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 정책에서 보안 주체 지정을 참조하세요.
- 각 문자열 값(`policy_id_string`, `sid_string`, `principal_id_string`, `action_string`, `resource_string`, `condition_type_string`, `condition_key_string`, 그리고 `condition_value`의 문자열 버전)은 자체적인 최소/최대 길이 제한, 특정 허용 값 또는 필수 내부 포맷을 가질 수 있습니다.

문자열 값에 대한 참고 사항

이 섹션에서는 정책에서 각각 다른 요소에 사용되는 문자열 값에 대한 추가 정보에 대해 살펴보겠습니다.

action_string

서비스 네임스페이스, 콜론 및 작업 이름으로 구성됩니다. 작업 이름에는 와일드카드를 추가할 수 있습니다. 예:

```
"Action":"ec2:StartInstances"

"Action":[
  "ec2:StartInstances",
  "ec2:StopInstances"
]

"Action":"cloudformation:*"

"Action":"*"

"Action":[
  "s3:Get*",
  "s3:List*"
]
```

policy_id_string

정책 관련 정보를 전체적으로 추가하는 방법을 제공합니다. Amazon SQS나 Amazon SNS 같은 일부 서비스는 Id 요소를 예약 방식으로 사용합니다. 개별 서비스에서 달리 제한하지 않는다면 policy_id_string에 공백을 추가할 수 있습니다. AWS 계정 내에서 이 값의 고유성을 요구하는 서비스도 있습니다.

> **Note**
>
> id_block은 리소스 기반 정책에서는 허용되지만 ID 기반 정책에서는 사용할 수 없습니다.

이 문자열이 제한된 전체 정책 길이에 영향을 끼치기는 하지만 문자열 길이에 제한은 없습니다.

```
"Id":"Admin_Policy"

"Id":"cd3ad3d9-2776-4ef1-a904-4c229d1642ee"
```

sid_string

개별 문에 대한 정보를 추가하는 방법을 제공합니다. IAM 정책의 경우 기본 영숫자 문자(A-Z,a-z,0-9)만 Sid 값의 문자로 허용됩니다. 리소스 정책을 지원하는 다른 AWS 서비스는 Sid 값 요구 사항이 다를 수 있습니다. 예를 들어 일부 서비스는 이 값이 특정 AWS 계정에서 고유할 것을 요구하며, 일부 서비스는 Sid 값으로 공백과 같은 문자를 추가로 허용합니다.

```
"Sid":"1"

"Sid": "ThisStatementProvidesPermissionsForConsoleAccess"
```

principal_id_string

보안 주체는 AWS 계정, IAM 사용자, IAM 역할, 페더레이션 사용자, 또는 위임된 역할 사용자의 Amazon 리소스 이름(ARN)을 사용해 지정합니다. AWS 계정의 경우, 전체 ARN 대신 짧은 형식인 AWS:*accountnumber*를 사용할 수도 있습니다. AWS 서비스, 위임된 역할 등을 포함한 모든 옵션에 대해서는 보안 주체 지정 섹션을 참조하세요.

"모든 사용자/익명 사용자"를 지정할 때만 *를 사용할 수 있습니다. 이름이나 ARN의 일부를 지정하기 위해 사용할 수는 없습니다.

resource_string

대부분의 경우 Amazon 리소스 이름(ARN)으로 구성됩니다.

```
"Resource":"arn:aws:iam::123456789012:user/Bob"

"Resource":"arn:aws:s3:::examplebucket/*"
```

condition_type_string

StringEquals, StringLike, NumericLessThan, DateGreaterThanEquals, Bool, BinaryEquals, IpAddress, ArnEquals 등 테스트할 조건 형식을 식별합니다. 조건 형식에 대한 전체 목록은 IAM JSON 정책 요소: 조건 연산자 섹션을 참조하세요.

```
"Condition": {
  "NumericLessThanEquals": {
    "s3:max-keys": "10"
  }
}
```

```
"Condition": {
  "Bool": {
    "aws:SecureTransport": "true"
  }
}

"Condition": {
  "StringEquals": {
      "s3:x-amz-server-side-encryption": "AES256"
   }
}
```

condition_key_string

값을 테스트하여 조건 충족 여부를 판단할 수 있는 조건 키를 식별합니다. AWS은 `aws:PrincipalType`, `aws:SecureTransport` 및 `aws:userid`를 포함해 모든 AWS 서비스에 사용할 수 있는 조건 키 집합을 정의합니다.

AWS 조건 키 목록에 대한 자세한 내용은 AWS 글로벌 조건 컨텍스트 키 섹션을 참조하세요. 서비스별 조건 키에 대한 자세한 내용은 다음과 같은 서비스 설명서를 참조하세요.

- Amazon Simple Storage Service 사용 설명서의 정책에서 조건 지정
- Amazon EC2 사용 설명서의 Amazon EC2의 IAM 정책

```
"Condition":{
  "Bool": {
      "aws:SecureTransport": "true"
   }
}

"Condition": {
  "StringNotEquals": {
      "s3:x-amz-server-side-encryption": "AES256"
   }
}

"Condition": {
  "StringEquals": {
    "aws:ResourceTag/purpose": "test"
  }
}
```

condition_value_string

조건이 충족되는지 여부를 결정하는 condition_key_string 값을 식별합니다. 조건 유형에 대한 유효한 값의 전체 목록은 IAM JSON 정책 요소: 조건 연산자 섹션을 참조하세요.

```
"Condition":{
  "ForAnyValue:StringEquals": {
  "dynamodb:Attributes": [
   "ID",
   "PostDateTime"
         ]
  }
}
```

직무에 관한 AWS 관리형 정책

최소 권한 부여 또는 작업 수행에 요구되는 권한만 부여하는 정책을 사용하는 것이 좋습니다. 최소 권한을 부여하는 가장 안전한 방법은 팀에 필요한 권한만 있는 사용자 지정 정책을 작성하는 것입니다. 필요한 경우 팀에서 추가 권한을 요청할 수 있는 프로세스를 만들어야 합니다. 팀에 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하기 위해서는 시간과 전문 지식이 필요합니다.

IAM 자격 증명(사용자, 사용자 그룹 및 역할)에 대한 권한을 추가하려면 AWS 관리형 정책을 사용할 수 있습니다. AWS 관리형 정책은 일반적인 사용 사례를 다루며 사용자의 AWS 계정에서 사용할 수 있습니다. AWS 관리형 정책은 최소 권한을 부여하지 않습니다. 보안 주체가 작업을 수행하는 데 필요한 것보다 더 많은 권한을 부여할 경우 보안 위험을 고려해야 합니다.

AWS 관리형 정책(작업 기능 포함)을 모든 IAM 자격 증명에 연결할 수 있습니다. 최소 권한으로 전환하려면 AWS Identity and Access Management Access Analyzer를 실행하여 AWS 관리형 정책으로 보안 주체를 모니터링합니다. 사용 권한을 학습한 후 사용자 지정 정책을 작성하거나 팀에 필요한 권한만 있는 정책을 생성할 수 있습니다. 이는 덜 안전하지만 팀에서 AWS를 어떻게 사용하는지 학습할 수록 더 많은 유연성을 제공합니다..

직무에 관한 AWS 관리형 정책은 IT 업계의 일반적인 직무 기능과 긴밀하게 연결되도록 구성됩니다. 이 정책을 적용하면 특정 직무 담당자에게 기대되는 작업 수행에 필요한 권한을 부여할 수 있습니다. 이 정책은 여러 서비스에 대한 권한을 정책 하나에 통합하기 때문에, 여러 정책에 권한이 분산되어 있는 경우보다 업무 절차가 간소합니다.

역할을 이용한 서비스 결합

일부 정책은 IAM 서비스 역할을 이용하여 다른 AWS 서비스에 포함된 기능을 활용할 수 있도록 지원합니다. 이 정책은 `iam:passrole`에 대한 액세스를 허용하여, 정책에 정의된 사용자가 역할을 AWS 서비스에 전달할 수 있도록 합니다. 이 역할은 AWS 서비스에서 사용자를 대행할 수 있도록 IAM 권한을 위임합니다.

필요에 따라 역할을 만들어야 합니다. 예를 들어 네트워크 관리자 정책은 정책을 지닌 사용자가 'flow-logs-vpc'라는 역할을 Amazon CloudWatch 서비스로 전달하도록 허용합니다. CloudWatch는 이 역할을 사용하여 사용자가 생성한 VPC의 IP 트래픽을 로그하고 캡처합니다.

보안 모범 사례를 따르기 위해 직무 기능에 관한 정책에는 전달할 수 있는 유효한 역할의 이름을 제한하는 필터가 포함되어 있습니다. 따라서 불필요한 권한을 부여할 가능성이 없습니다. 사용자가 선택적 서비스 역할을 필요로 할 경우, 정책에 정의된 명명 규칙에 따라 역할을 만들어야 합니다. 그런 다음 해당 역할에 권한을 부여합니다. 그러면 사용자는 역할이 제공하는 모든 권한을 부여해 서비스에서 이 역할을 사용하도록 구성할 수 있습니다.

다음 섹션에서 각 정책의 이름에는 AWS Management Console의 정책 세부 정보 페이지로 이동하는 링크가 연결되어 있습니다. 해당 페이지에서 정책 문서를 확인하고 부여된 권한을 검토할 수 있습니다.

관리자 직무

AWS 관리형 정책 이름: AdministratorAccess

사용 사례: 이 사용자는 모든 액세스를 가지며 AWS 내 모든 서비스와 리소스에 권한을 위임할 수 있습니다.

정책 업데이트 AWS는 이 정책을 유지 관리하고 업데이트합니다. 이 정책의 변경 내역을 보려면 IAM 콘솔에서 정책을 확인한 다음 정책 버전(Policy versions) 탭을 선택합니다. 작업 기능 정책 업데이트에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 업데이트 섹션을 참조하세요.

정책 설명: 이 정책은 모든 AWS 서비스와 계정 내 모든 리소스에 대한 모든 작업을 허용합니다. 관리형 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 가이드의 AdministratorAccess를 참조하세요.

> **Note**
>
> IAM 사용자 또는 역할이 이 정책의 권한을 통해 AWS Billing and Cost Management 콘솔에 액세스할 수 있으려면, 먼저 IAM 사용자 및 역할 액세스를 활성화해야 합니다. 이를 위해 결제 콘솔에 대한 액세스 권한 부여의 지침을 따라 결제 콘솔에 액세스를 위임합니다.

결제 직무

AWS 관리형 정책 이름: Billing

사용 사례: 이 사용자는 결제 정보를 확인하고 지불을 설정 및 승인해야 합니다. 사용자가 전체 AWS 서비스에 누적된 비용을 모니터링할 수 있습니다.

정책 업데이트 AWS는 이 정책을 유지 관리하고 업데이트합니다. 이 정책의 변경 내역을 보려면 IAM 콘솔에서 정책을 확인한 다음 정책 버전(Policy versions) 탭을 선택합니다. 작업 기능 정책 업데이트에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 업데이트 섹션을 참조하세요.

정책 설명: 이 정책은 결제 관리, 비용, 결제 방법, 예산, 보고서 등에 필요한 모든 권한을 부여합니다. 추가적인 비용 관리 정책 예는 AWS Billing and Cost Management 사용 설명서의 AWS Billing 정책 예를 참조하세요. 관리형 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 가이드의 Billing를 참조하세요.

> **Note**
>
> IAM 사용자 또는 역할이 이 정책의 권한을 통해 AWS Billing and Cost Management 콘솔에 액세스할 수 있으려면, 먼저 IAM 사용자 및 역할 액세스를 활성화해야 합니다. 이를 위해 결제 콘솔에 대한 액세스 권한 부여의 지침을 따라 결제 콘솔에 액세스를 위임합니다.

데이터베이스 관리자 직무

AWS 관리형 정책 이름: DatabaseAdministrator

사용 사례: 이 사용자는 AWS 클라우드에서 데이터베이스를 설정, 구성, 유지합니다.

정책 업데이트 AWS는 이 정책을 유지 관리하고 업데이트합니다. 이 정책의 변경 내역을 보려면 IAM 콘솔에서 정책을 확인한 다음 정책 버전(Policy versions) 탭을 선택합니다. 작업 기능 정책 업데이트에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 업데이트 섹션을 참조하세요.

정책 설명: 이 정책은 데이터베이스를 생성, 구성, 유지할 수 있는 권한을 부여합니다. 여기에는 Amazon DynamoDB, Amazon Relational Database Service(RDS) 및 Amazon Redshift와 같은 AWS 데이터베이스 서비스에 대한 액세스가 포함됩니다. 이 정책이 지원하는 데이터베이스 서비스의 전체 목록에 대한 정책을 봅니다. 관리형 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 가이드의 DatabaseAdministrator를 참조하세요.

이 직무 정책은 AWS 서비스로 역할을 전달할 수 있는 기능을 지원합니다. 이 정책은 다음 표에 명시된 역할에만 `iam:PassRole` 작업을 허용합니다. 자세한 내용은 이 주제의 후반부에서 역할 생성 및 정책 연결(콘솔) 섹션을 참조하세요.

사용 사례	역할 이름(*는 와일드카드)	선택할 서비스 역할 유형	이 AWS 관리형 정책 선택
사용자가 RDS 데이터베이스를 모니터링하도록 허용	rds-monitoring-role	확장 모니터링에 대한 Amazon RDS 역할	AmazonRDSEnhancedMonitoringRole
AWS Lambda의 데이터베이스 모니터링과 외부 데이터베이스 액세스를 허용	rdbms-lambda-access	Amazon EC2	AWSLambda_FullAccess
Lambda가 DynamoDB를 사용하여 Amazon S3 및 Amazon Redshift 클러스터에 파일을 업로드하도록 허용	lambda_exec_role	AWS Lambda	AWS 빅 데이터 블로그에 정의된 대로 새로운 관리형 정책 구성
Lambda 기능이 DynamoDB 테이블의 트리거 역할을 하도록 허용	lambda-dynamodb-*	AWS Lambda	AWSLambdaDynamoDBExecutionRole
Lambda 기능이 VPC에서 Amazon RDS에 액세스하도록 허용	lambda-vpc-execution-role	AWS Lambda개발자 가이드에 정의된 대로 신뢰 정책으로 역할 생성	AWSLambdaVPCAccessExecutionRole
AWS Data Pipeline가 AWS 리소스에 액세스하도록 허용	DataPipelineDefaultRole	AWS Data Pipeline 개발자 가이드에 정의된 대로 신뢰 정책으로 역할 생성	AWS Data Pipeline 설명서에는 이 사용 사례에 필요한 사용 권한이 나열됩니다. AWS Data Pipeline에 대한 IAM 역할 참조

사용 사례	역할 이름(*는 와일드카드)	선택할 서비스 역할 유형	이 AWS 관리형 정책 선택
Amazon EC2 인스턴스에서 실행 중인 애플리케이션이 AWS 리소스에 액세스하도록 허용	DataPipelineDefaultResourceRole	AWS Data Pipeline 개발자 가이드에 정의된 대로 신뢰 정책으로 역할 생성	AmazonEC2RoleforDataPipelineRole

데이터 사이언티스트 직무

AWS 관리형 정책 이름: DataScientist

사용 사례: 이 사용자는 Hadoop 작업과 쿼리를 실행합니다. 또한 데이터 분석 및 비즈니스 인텔리전스에 관한 정보에 액세스하고 이를 분석합니다.

정책 업데이트 AWS는 이 정책을 유지 관리하고 업데이트합니다. 이 정책의 변경 내역을 보려면 IAM 콘솔에서 정책을 확인한 다음 정책 버전(Policy versions) 탭을 선택합니다. 작업 기능 정책 업데이트에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 업데이트 섹션을 참조하세요.

정책 설명: 이 정책은 Amazon EMR 클러스터에서 쿼리를 생성, 관리, 실행하고 Amazon QuickSight 같은 도구로 데이터 분석을 수행할 수 있는 권한을 부여합니다. 이 정책에는 AWS Data Pipeline, Amazon EC2, Amazon Kinesis, Amazon Machine Learning, SageMaker와 같은 추가적인 데이터 사이언티스트 서비스에 대한 액세스가 포함됩니다. 이 정책이 지원하는 데이터 과학자 서비스의 전체 목록에 대한 정책을 봅니다. 관리형 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 가이드의 DataScientist를 참조하세요.

이 직무 정책은 AWS 서비스로 역할을 전달할 수 있는 기능을 지원합니다. 한 개의 문이 역할을 SageMaker에 전달하도록 허용합니다. 또 다른 문은 다음 표에 명시된 역할에만 `iam:PassRole` 작업을 허용합니다. 자세한 내용은 이 주제의 후반부에서 역할 생성 및 정책 연결(콘솔) 섹션을 참조하세요.

사용 사례	역할 이름(*는 와일드카드)	선택할 서비스 역할 유형	선택할 AWS 관리형 정책
클러스터에 적합한 서비스와 리소스에 대한 Amazon EC2 인스턴스 액세스를 허용	EMR-EC2_DefaultRole	EC2에 대한 Amazon EMR	AmazonElasticMapReduceforEC2Role

사용 사례	역할 이름(*는 와일드카드)	선택할 서비스 역할 유형	선택할 AWS 관리형 정책
Amazon EMR이 클러스터에 대한 Amazon EC2 서비스와 리소스에 액세스하도록 허용	EMR_DefaultRole	Amazon EMR	AmazonEMR ServicePolicy_v2
Apache Flink용 Kinesis 관리형 서비스가 스트리밍 데이터 소스에 액세스하도록 허용	kinesis-*	AWS 빅 데이터 블로그에 정의된 대로 신뢰 정책으로 역할을 생성합니다.	사용 사례에 따라 네 가지 가능한 옵션을 소개한 AWS 빅 데이터 블로그 참조
AWS Data Pipeline가 AWS 리소스에 액세스하도록 허용	DataPipelineDefaultRole	AWS Data Pipeline 개발자 가이드에 정의된 대로 신뢰 정책으로 역할 생성	AWS Data Pipeline 설명서에는 이 사용 사례에 필요한 사용 권한이 나열됩니다. AWS Data Pipeline에 대한 IAM 역할 참조
Amazon EC2 인스턴스에서 실행 중인 애플리케이션이 AWS 리소스에 액세스하도록 허용	DataPipelineDefaultResourceRole	AWS Data Pipeline 개발자 가이드에 정의된 대로 신뢰 정책으로 역할 생성	AmazonEC2RoleforDataPipelineRole

개발자 고급 사용자 직무

AWS 관리형 정책 이름: PowerUserAccess

사용 사례: 이 사용자는 애플리케이션 개발 작업을 수행하며, AWS 인식 애플리케이션 개발을 지원하는 리소스와 서비스를 생성하고 구성할 수 있습니다.

정책 업데이트 AWS는 이 정책을 유지 관리하고 업데이트합니다. 이 정책의 변경 내역을 보려면 IAM 콘솔에서 정책을 확인한 다음 정책 버전(Policy versions) 탭을 선택합니다. 작업 기능 정책 업데이트에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 업데이트 섹션을 참조하세요.

정책 설명: 이 정책의 첫 번째 설명문은 `NotAction` 요소를 사용하여 모든 AWS 서비스와 모든 리소스(AWS Identity and Access Management, AWS Organizations 및 AWS Account Management 제외)

에 대해 모든 작업을 허용합니다. 두 번째 설명문은 서비스에 연결된 역할을 생성할 수 있는 IAM 권한을 부여합니다. 이것은 Amazon S3 버킷처럼 다른 서비스에서 리소스에 액세스해야 하는 서비스에 필요합니다. 또한 조직에 관리 계정 이메일과 조직 한도 등 사용자 조직에 대한 정보를 볼 권한을 부여합니다. 이 정책은 IAM 및 조직을 제한하지만 IAM Identity Center가 활성화된 경우 사용자가 모든 IAM Identity Center 작업을 수행할 수 있습니다. 또한 계정에 대해 활성화 또는 비활성화된 AWS 리전을 볼 수 있는 계정 관리 권한을 부여합니다.

네트워크 관리자 직무

AWS 관리형 정책 이름: NetworkAdministrator

사용 사례: 이 사용자는 AWS 네트워크 리소스를 설정하고 유지하는 작업을 담당합니다.

정책 업데이트 AWS는 이 정책을 유지 관리하고 업데이트합니다. 이 정책의 변경 내역을 보려면 IAM 콘솔에서 정책을 확인한 다음 정책 버전(Policy versions) 탭을 선택합니다. 작업 기능 정책 업데이트에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 업데이트 섹션을 참조하세요.

정책 설명: 이 정책은 Auto Scaling, Amazon EC2, AWS Direct Connect, Route 53, Amazon CloudFront, Elastic Load Balancing, AWS Elastic Beanstalk, Amazon SNS, CloudWatch, CloudWatch Logs, Amazon S3, IAM, Amazon Virtual Private Cloud의 네트워크 리소스를 생성하고 유지할 수 있는 권한을 부여합니다. 관리형 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 가이드의 NetworkAdministrator를 참조하세요.

이 직무는 AWS 서비스로 역할을 전달할 수 있는 기능을 필요로 합니다. 이 정책은 다음 표에 명시된 역할에만 `iam:GetRole` 및 `iam:PassRole`을 허용합니다. 자세한 내용은 이 주제의 후반부에서 역할 생성 및 정책 연결(콘솔) 섹션을 참조하세요.

사용 사례	역할 이름(*는 와일드카드)	선택할 서비스 역할 유형	선택할 AWS 관리형 정책
Amazon VPC가 사용자를 대신해 CloudWatch Logs의 로그를 생성하고 관리하여 VPC로 들어오고 나가는 IP 트래픽을 모니터링하도록 허용	flow-logs-*	Amazon VPC 사용 가이드에 정의된 대로 신뢰 정책으로 역할 생성	이 사용 사례에는 기존 AWS 관리형 정책이 없지만 설명서에는 필요한 권한이 나열되어 있습니다. Amazon VPC 사용 설명서를 참조하세요.

읽기 전용 액세스

AWS 관리형 정책 이름: ReadOnlyAccess

사용 사례: 이 사용자는 AWS 계정의 모든 리소스에 대한 읽기 전용 액세스가 필요합니다.

정책 업데이트 AWS는 이 정책을 유지 관리하고 업데이트합니다. 이 정책의 변경 내역을 보려면 IAM 콘솔에서 정책을 확인한 다음 정책 버전(Policy versions) 탭을 선택합니다. 작업 기능 정책 업데이트에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 업데이트 섹션을 참조하세요.

정책 설명: 이 정책은 리소스와 해당 속성을 나열하고, 가져오고, 설명하고 볼 수 있는 권한을 부여합니다. 생성 또는 삭제와 같은 변형 기능은 포함되지 않습니다. 이 정책에는 보안 관련 AWS 서비스(예: AWS Identity and Access Management 및 AWS Billing and Cost Management)에 대한 읽기 전용 액세스가 포함됩니다. 이 정책이 지원하는 서비스 및 작업의 전체 목록에 대한 정책을 봅니다.

보안 감사자 직무

AWS 관리형 정책 이름: SecurityAudit

사용 사례: 이 사용자는 보안 요구 사항을 준수하기 위해 계정을 모니터링합니다. 이 사용자는 로그와 이벤트에 액세스하여 잠재적인 보안 위반이나 악의적인 활동을 조사할 수 있습니다.

정책 업데이트 AWS는 이 정책을 유지 관리하고 업데이트합니다. 이 정책의 변경 내역을 보려면 IAM 콘솔에서 정책을 확인한 다음 정책 버전(Policy versions) 탭을 선택합니다. 작업 기능 정책 업데이트에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 업데이트 섹션을 참조하세요.

정책 설명: 이 정책은 많은 AWS 서비스에 대한 구성 데이터를 확인하고, 해당 로그를 검토할 수 있는 권한을 부여합니다. 관리형 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 가이드의 SecurityAudit를 참조하세요.

지원 사용자 직무

AWS 관리형 정책 이름: SupportUser

사용 사례: 이 사용자는 AWS 지원을 통해 지원 사례를 생성하고 기존 사례의 상태를 확인합니다.

정책 업데이트 AWS는 이 정책을 유지 관리하고 업데이트합니다. 이 정책의 변경 내역을 보려면 IAM 콘솔에서 정책을 확인한 다음 정책 버전(Policy versions) 탭을 선택합니다. 작업 기능 정책 업데이트에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 업데이트 섹션을 참조하세요.

정책 설명: 이 정책은 AWS Support 사례를 생성하고 업데이트할 수 있는 권한을 부여합니다. 관리형 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 가이드의 SupportUser을 참조하세요.

시스템 관리자 직무

AWS 관리형 정책 이름: SystemAdministrator

사용 사례: 이 사용자는 개발 작업에 필요한 리소스를 설정하고 유지합니다.

정책 업데이트 AWS는 이 정책을 유지 관리하고 업데이트합니다. 이 정책의 변경 내역을 보려면 IAM 콘솔에서 정책을 확인한 다음 정책 버전(Policy versions) 탭을 선택합니다. 작업 기능 정책 업데이트에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 업데이트 섹션을 참조하세요.

정책 설명: 이 정책은 AWS CloudTrail, Amazon CloudWatch, AWS CodeCommit, AWS CodeDeploy, AWS Config, AWS Directory Service, Amazon EC2, AWS Identity and Access Management, AWS Key Management Service, AWS Lambda, Amazon RDS, Route 53, Amazon S3, Amazon SES, Amazon SQS, AWS Trusted Advisor, Amazon VPC를 포함해 대량의 AWS 서비스에 걸쳐서 리소스를 생성하고 유지할 수 있는 권한을 부여합니다. 관리형 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 가이드의 SystemAdministrator를 참조하세요.

이 직무는 AWS 서비스로 역할을 전달할 수 있는 기능을 필요로 합니다. 이 정책은 다음 표에 명시된 역할에만 `iam:GetRole` 및 `iam:PassRole`을 허용합니다. 자세한 내용은 이 주제의 후반부에서 역할 생성 및 정책 연결(콘솔) 섹션을 참조하세요. 작업 기능 정책 업데이트에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 업데이트 섹션을 참조하세요.

사용 사례	역할 이름(*는 와일드카드)	선택할 서비스 역할 유형	선택할 AWS 관리형 정책
Amazon ECS 클러스터 내 EC2 인스턴스에서 실행 중인 앱이 Amazon ECS에 액세스하도록 허용	ecr-sysadmin-*	EC2 Container Service에 대한 Amazon EC2 역할	AmazonEC2ContainerServiceforEC2Role
사용자가 데이터베이스를 모니터링하도록 허용	rds-monitoring-role	확장 모니터링에 대한 Amazon RDS 역할	AmazonRDSEnhancedMonitoringRole
EC2 인스턴스에서 실행 중인 앱이 AWS 리소스에 액세스하도록 허용합니다.	ec2-sysadmin-*	Amazon EC2	Amazon EC2 사용 설명서에 나온 것과 같이 S3 버킷에 대한 액세스 권한을 부여하는 역할의 샘플

사용 사례	역할 이름(*는 와일드카드)	선택할 서비스 역할 유형	선택할 AWS 관리형 정책
			정책. 필요에 따라 사용자 지정
Lambda가 DynamoDB Streams을 읽고 CloudWatch Logs에 작성하도록 허용	lambda-sysadmin-*	AWS Lambda	AWSLambdaDynamoDBExecutionRole

보기 전용 사용자 직무

AWS 관리형 정책 이름: ViewOnlyAccess

사용 사례: 이 사용자는 여러 서비스에 걸쳐 계정 내 AWS 리소스와 기본 메타데이터 목록을 확인할 수 있습니다. 하지만 할당량을 초과하는 리소스 콘텐츠나 메타데이터, 리소스의 목록 정보를 읽을 수 없습니다.

정책 업데이트 AWS는 이 정책을 유지 관리하고 업데이트합니다. 이 정책의 변경 내역을 보려면 IAM 콘솔에서 정책을 확인한 다음 정책 버전(Policy versions) 탭을 선택합니다. 작업 기능 정책 업데이트에 대한 자세한 내용은 직무에 관한 AWS 관리형 정책 업데이트 섹션을 참조하세요.

정책 설명: 이 정책은 AWS 서비스 리소스에 대한 `List*`, `Describe*`, `Get*`, `View*`, `Lookup*` 액세스를 부여합니다. 각 서비스에 대해 이 정책에 포함된 작업을 보려면 ViewOnlyAccess 섹션을 참조하세요. 관리형 정책에 대한 자세한 내용은 AWS 관리형 정책 참조 가이드의 ViewOnlyAccess를 참조하세요.

직무에 관한 AWS 관리형 정책 업데이트

이러한 정책은 모두 AWS가 유지하며, AWS 서비스에서 정책을 추가할 때 새로운 서비스와 새로운 기능에 대한 지원을 포함시켜 모든 것을 최신 상태로 유지합니다. 이러한 정책은 고객이 수정할 수 없습니다. 정책 사본을 만든 후 이를 수정할 수 있으나 AWS가 새로운 서비스와 API 작업을 도입할 때 이 사본이 자동으로 업데이트되지는 않습니다.

직무 정책의 경우 IAM 콘솔에서 버전 기록 및 각 업데이트의 시간과 날짜를 볼 수 있습니다. 이렇게 하려면 이 페이지의 링크를 사용하여 정책 세부 정보를 봅니다. 그런 다음 정책 버전(Policy versions) 탭을 클릭하여 버전을 볼 수 있습니다. 이 페이지에는 정책의 최근 25개 버전이 표시됩니다. 정책의 모든 버전을 보려면 get-policy-version AWS CLI 명령 또는 GetPolicyVersion API 작업을 호출합니다.

> **Note**
>
> 고객 관리형 정책의 최대 5개의 버전을 사용할 수 있지만 AWS는 AWS 관리형 정책의 전체 버전 기록을 유지합니다.

역할 생성 및 정책 연결(콘솔)

앞서 나열한 여러 가지 정책은 AWS 서비스에서 사용자 대신 작업을 수행할 수 있도록 해주는 역할을 이용해 해당 서비스를 구성할 수 있는 권한을 부여합니다. 직무 정책은 반드시 사용해야 하는 정확한 역할 이름을 정의하거나, 사용할 수 있는 이름의 앞부분을 지정하는 접두사만이라도 포함합니다. 이러한 역할 중 하나를 생성하려면 다음 절차의 단계를 따릅니다.

AWS 서비스에 대한 역할 생성(IAM 콘솔)

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할을 선택하고 역할 생성을 선택합니다.
3. 신뢰할 수 있는 엔터티 유형에 AWS 서비스를 선택합니다.
4. 서비스 또는 사용 사례의 경우 서비스를 선택한 다음, 사용 사례를 선택합니다. 사용 사례는 서비스에 필요한 신뢰 정책을 포함하기 위해 서비스에서 정합니다.
5. Next(다음)를 선택합니다.
6. 권한 정책의 경우 선택한 사용 사례에 따라 옵션이 달라집니다.
   - 서비스가 역할에 대한 권한을 정의하는 경우 권한 정책을 선택할 수 없습니다.
   - 제한된 권한 정책 세트에서 선택합니다.
   - 모든 권한 정책에서 선택합니다.
   - 권한 정책 없음을 선택하고 역할이 생성된 후 정책을 생성한 다음, 정책을 역할에 연결합니다.
7. (선택 사항) 권한 경계를 선택합니다. 이는 서비스 역할에서 가능한 고급 기능이며 서비스 링크된 역할은 아닙니다.
   a. 권한 경계 설정 섹션을 열고 최대 역할 권한을 관리하기 위한 권한 경계 사용을 선택합니다.

      IAM은 계정의 AWS 관리형 또는 고객 관리형 정책 목록을 포함합니다.
   b. 정책을 선택하여 권한 경계를 사용하세요.
8. Next(다음)를 선택합니다.

9. 역할 이름의 경우 옵션은 서비스에 따라 달라집니다.

   - 서비스에서 역할 이름을 정의하는 경우 이 역할 이름을 편집할 수 없습니다.
   - 서비스에서 역할 이름에 대한 접두사를 정의하는 경우 사용자가 선택적 접미사를 입력할 수 있습니다.
   - 서비스에서 역할 이름을 정의하지 않는 경우 역할 이름을 지정할 수 있습니다.

   > **Important**
   >
   > 역할 이름을 지정할 때는 다음 사항에 유의하세요.
   >
   > - 역할 이름은 AWS 계정 내에서 고유해야 하지만 대소문자를 구분하지는 않습니다.
   >
   >   예를 들어, 이름이 **`PRODROLE`**과 **`prodrole`**, 두 가지로 지정된 역할을 만들지 마십시오. 역할 이름이 정책 또는 ARN의 일부로 사용되는 경우 역할 이름은 대소문자를 구분합니다. 그러나 로그인 프로세스와 같이 콘솔에서 역할 이름이 고객에게 표시되는 경우에는 역할 이름이 대소문자를 구분하지 않습니다.
   > - 다른 엔터티가 역할을 참조할 수 있기 때문에 역할이 생성된 후에는 역할 이름을 편집할 수 없습니다.

10. (선택 사항)설명에 역할에 대한 설명을 입력합니다.
11. (선택 사항) 역할에 대한 사용 사례와 권한을 편집하려면 1단계: 신뢰할 수 있는 엔터티 선택 또는 2단계: 권한 추가 섹션에서 편집을 선택합니다.
12. (선택 사항) 태그를 키-값 페어로 연결하여 역할을 식별, 구성 또는 검색합니다. IAM에서 태그 사용에 대한 자세한 내용을 알아보려면 IAM 사용 설명서의 IAM 리소스에 태그 지정을 참조하세요.
13. 역할을 검토한 다음 역할 생성을 선택합니다.

예시 1: 사용자를 데이터베이스 관리자로 구성(콘솔)

이 예시는 IAM 사용자 Alice를 데이터베이스 관리자로 구성하는 데 필요한 단계를 보여줍니다. 이 섹션에서 테이블 첫 번째 행의 정보를 사용하여 사용자가 Amazon RDS 모니터링을 지원하도록 허용합니다. Amazon 데이터베이스 서비스를 관리할 수 있도록 DatabaseAdministrator 정책을 Alice의 IAM 사용자에게 연결합니다. 이 정책을 통해 Alice는 `rds-monitoring-role`(이)라는 역할을 Amazon RDS 서비스로 전달할 수도 있습니다. 그러면 사용자를 대신해 Amazon RDS 데이터베이스를 모니터링합니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.

2. 정책을 선택한 후 검색 상자에 **database**를 입력하고 Enter를 누릅니다.
3. DatabaseAdministrator 정책의 라디오 버튼과 작업, 연결을 차례로 선택합니다.
4. 엔터티 목록에서 Alice를 선택한 후 정책 연결을 선택합니다. 이제 Alice가 AWS 데이터베이스를 관리할 수 있습니다. 하지만 Alice가 이 데이터베이스를 모니터링하도록 허용하려면 서비스 역할을 구성해야 합니다.
5. IAM 콘솔의 탐색 창에서 역할을 선택하고 역할 생성을 선택합니다.
6. AWS 서비스 역할 유형을 선택한 후 Amazon RDS를 선택합니다.
7. 확장 모니터링을 위한 Amazon RDS 역할(Amazon RDS Role for Enhanced Monitoring) 사용 사례를 선택합니다.
8. Amazon RDS는 역할에 대한 권한을 정의합니다. 계속하려면 Next: Review(다음: 검토)를 선택합니다.
9. 역할 이름은 현재 Alice가 적용하는 DatabaseAdministrator 정책에 지정된 것 중 하나여야 합니다. 그중 하나는 **rds-monitoring-role**입니다. Role name(역할 이름)에 이를 입력합니다.
10. (선택 사항) Role description(역할 설명)에 새 역할에 대한 설명을 입력합니다.
11. 세부 정보를 검토한 후 역할 생성을 선택합니다.
12. 이제 Alice는 Amazon RDS 콘솔의 모니터링 섹션에서 RDS 확장 모니터링(RDS Enhanced Monitoring)을 활성화할 수 있습니다. 예를 들어, DB 인스턴스 또는 읽기 전용 복제본을 생성하거나 DB 인스턴스를 수정할 때 이렇게 합니다. Enable Enhanced Monitoring(확장 모니터링 활성화)를 Yes(예)로 설정하면서 Monitoring Role(역할 모니터링) 상자에 생성한 역할 이름(rds-monitoring-role)을 입력해야 합니다.

예시 2: 사용자를 네트워크 관리자로 구성(콘솔)

이 예시는 IAM 사용자 Jorge를 네트워크 관리자로 구성하는 데 필요한 단계를 보여줍니다. 해당 섹션에서 테이블의 정보를 사용하여 Jorge가 VPC로 들어오고 나가는 IP 트래픽을 모니터링하도록 허용합니다. 또한 Jorge가 CloudWatch Logs의 로그에서 해당 정보를 캡처하도록 허용합니다. AWS 네트워크 리소스를 구성할 수 있도록 Jorge의 IAM 사용자에게 NetworkAdministrator 정책을 연결합니다. 또한 이 정책 덕분에 흐름 로그를 작성할 때 Jorge가 `flow-logs*`(으)로 시작하는 역할을 Amazon EC2로 전달하도록 설정할 수 있습니다. 예시 1과 달리 이 시나리오에서는 사전 정의된 서비스 역할 유형이 없기 때문에 몇 가지 단계를 다르게 수행해야 합니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택한 후 검색 상자에 **network**를 입력합니다.

3. NetworkAdministrator 정책 옆의 라디오 버튼에 이어 작업, 연결을 차례로 선택합니다.
4. 사용자 목록에서 Jorge 옆에 있는 확인란을 선택한 후 정책 연결(Attach policy)을 선택합니다. 이제 Jorge가 AWS 네트워크 리소스를 관리할 수 있습니다. 하지만 VPC 내 트래픽을 모니터링하도록 하려면 서비스 역할을 구성해야 합니다.
5. 생성해야 하는 서비스 역할에 사전 정의된 관리형 정책이 없기 때문에 먼저 이 정책부터 생성해야 합니다. 탐색 창에서 정책을 선택한 다음 정책 생성을 선택합니다.
6. 정책 편집기 섹션에서 JSON 옵션을 선택하고, 다음의 JSON 정책 문서에서 텍스트를 복사합니다. 이 텍스트를 JSON 텍스트 상자에 붙여 넣습니다.

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

7. 정책 검증 동안 생성된 모든 보안 경고, 오류 또는 일반 경고를 해결하고 다음을 선택합니다.

> **Note**
>
> 언제든지 시각적 편집기 옵션과 JSON 편집기 옵션을 서로 전환할 수 있습니다. 그러나 변경을 적용하거나 시각적 편집기에서 다음을 선택한 경우 IAM은 시각적 편집기에 최적화되도록 정책을 재구성할 수 있습니다. 자세한 내용은 정책 재구성 단원을 참조하십시오.

8. 검토 및 생성 페이지에서 정책 이름에 **`vpc-flow-logs-policy-for-service-role`**을 입력합니다. 이 정책에 정의된 권한을 검토하여 정책이 부여한 권한을 확인한 다음 정책 생성을 선택하여 작업을 저장합니다.

   새로운 정책이 관리형 정책 목록에 나타나며 연결 준비가 완료됩니다.

9. IAM 콘솔의 탐색 창에서 역할을 선택하고 역할 생성을 선택합니다.
10. AWS 서비스 역할 유형을 선택한 후 Amazon EC2를 선택합니다.
11. Amazon EC2 사용 사례를 선택합니다.
12. 권한 정책 연결 페이지에서 앞서 생성한 정책을 선택하고 vpc-flow-logs-policy-for-service-role과 Next: Review(다음: 검토)를 차례로 선택합니다.
13. 역할 이름은 현재 Jorge가 적용하는 NetworkAdministrator 정책에서 허용한 것이어야 합니다. `flow-logs-`로 시작하는 이름은 무엇이든 허용됩니다. 이 예시에서는 Role name(역할 이름)에 **`flow-logs-for-jorge`**을(를) 입력합니다.
14. (선택 사항) Role description(역할 설명)에 새 역할에 대한 설명을 입력합니다.
15. 세부 정보를 검토한 후 역할 생성을 선택합니다.
16. 이제 이 시나리오에 필요한 신뢰 정책을 구성할 수 있습니다. Roles(역할) 페이지에서 flow-logs-for-jorge 역할(확인란이 아닌 이름)을 선택합니다. 새 역할의 세부 정보 페이지에서 신뢰 관계 탭을 선택한 다음 Edit trust relationship(신뢰 관계 편집)을 선택합니다.
17. "Service" 라인을 다음과 같이 변경해 `ec2.amazonaws.com`의 항목을 교체합니다.

```
        "Service": "vpc-flow-logs.amazonaws.com"
```

18. 이제 Jorge가 Amazon EC2 콘솔에서 VPC나 서브넷의 흐름 로그를 생성할 수 있습니다. 흐름 로그를 생성할 때 flow-logs-for-jorge 역할을 지정합니다. 이 역할에는 로그를 생성하고 데이터를 쓸 수 있는 권한이 있습니다.

AWS 글로벌 조건 컨텍스트 키

보안 주체가 AWS에 요청하면 AWS는 요청 정보를 요청 컨텍스트로 수집합니다. JSON 정책의 `Condition` 요소를 사용하여 요청 컨텍스트의 키를 정책에서 지정한 키 값과 비교할 수 있습니다. 요청 정보는 요청을 수행하는 보안 주체, 요청의 대상이 되는 리소스, 요청 자체에 대한 메타데이터 등 다양한 소스를 통해 제공됩니다.

전역 조건 키는 모든 AWS 서비스에서 사용할 수 있습니다. 이러한 조건 키는 모든 정책에서 사용할 수 있지만 모든 요청 컨텍스트에서 키를 사용할 수 있는 것은 아닙니다. 예를 들어 `aws:SourceAccount` 조건 키는 AWS 서비스 보안 주체가 리소스를 직접적으로 호출하는 경우에만 사용할 수 있습니다. 요청 컨텍스트에 전역 키가 포함되는 상황에 대한 자세한 내용은 각 키의 가용성 정보를 참조하세요.

일부 개별 서비스는 다른 서비스의 요청 컨텍스트에서 사용할 수 있는 자체 조건 키를 생성합니다. 교차 서비스 조건 키는 `ec2:` 또는 `lambda:`와 같이 서비스 이름과 일치하는 접두사를 포함하지만 다른 서비스에서 사용할 수 있는 전역 조건 키의 한 유형입니다.

서비스별 조건 키는 개별 AWS 서비스에 사용할 수 있도록 정의됩니다. 예를 들어 Amazon S3에서는 `s3:VersionId` 조건 키를 사용하여 Amazon S3 객체의 특정 버전에 대한 액세스를 제한하는 정책을 작성할 수 있습니다. 이 조건 키는 서비스에 고유합니다. 즉, Amazon S3 서비스에 대한 요청에만 작동합니다. 서비스별 조건 키의 경우 AWS 서비스에 대한 작업, 리소스 및 조건 키를 참조하고 키를 보려는 서비스를 선택합니다.

Note

일부 상황에서만 사용할 수 있는 조건 키를 사용하는 경우 조건 연산자의 IfExists 버전을 사용할 수 있습니다. 요청 컨텍스트에 조건 키가 누락된 경우 정책이 평가에 실패할 수 있습니다. 예를 들어, 특정 IP 범위 또는 특정 VPC로부터 요청이 오는 경우 `...IfExists` 연산자와 함께 다음 조건 블록을 사용하여 일치시킵니다. 요청 컨텍스트에 두 키 중 하나 또는 둘 다 포함되어 있지 않은 경우에도 조건은 여전히 `true`를 반환합니다. 값은 지정된 키가 요청 컨텍스트에 포함된 경우에만 검사됩니다. 키가 다른 연산자에 없을 때 정책이 평가되는 방법에 대한 자세한 내용은 조건 연산자를 참조하세요.

```
"Condition": {
    "IpAddressIfExists": {"aws:SourceIp" : ["xxx"] },
    "StringEqualsIfExists" : {"aws:SourceVpc" : ["yyy"]}
}
```

Important

조건을 여러 키 값이 있는 요청 컨텍스트와 비교하려면 `ForAllValues` 또는 `ForAnyValue` 연산자를 설정해야 합니다. 집합 연산자는 다중 값 조건 키에만 사용합니다. 단일 값 조건 키(single-valued condition keys)에는 집합 연산자를 사용하지 마세요. 자세한 내용은 다중 값 컨텍스트 키 단원을 참조하십시오.

보안 주체의 속성	역할 세션의 속성	네트워크의 속성	리소스의 속성	요청의 속성
aws:PrincipalArn aws:PrincipalAccount	aws:FederatedProvider aws:TokenIssueTime	aws:SourceIp aws:SourceVpc aws:SourceVpce	aws:ResourceAccount aws:ResourceOrgPaths	aws:CalledVia aws:CalledViaFirst

보안 주체의 속성	역할 세션의 속성	네트워크의 속성	리소스의 속성	요청의 속성
aws:PrincipalOrgPaths	aws:MultiFactorAuthAge	aws:VpcSourceIp	aws:ResourceOrgID	aws:CalledViaLast
aws:PrincipalOrgID	aws:MultiFactorAuthPresent		aws:ResourceTag/tag-key	aws:ViaAWSService
aws:PrincipalTag/tag-key	aws:Ec2InstanceSourceVpc			aws:CurrentTime
aws:PrincipalIsAWSService	aws:Ec2InstanceSourcePrivateIPv4			aws:EpochTime
aws:PrincipalServiceName	aws:SourceIdentity			aws:referer
aws:PrincipalServiceNamesList	ec2:RoleDelivery			aws:RequestedRegion
aws:PrincipalType	ec2:SourceInstanceArn			aws:RequestTag/tag-key
aws:userid	glue:RoleAssumedBy			aws:TagKeys
aws:username	glue:CredentialIssuingService			aws:SecureTransport
	lambda:SourceFunctionArn			aws:SourceArn
	ssm:SourceInstanceArn			aws:SourceAccount
	identitystore:UserId			aws:SourceOrgPaths
				aws:SourceOrgID
				aws:UserAgent

보안 주체의 속성

다음 조건 키를 사용하여 요청하는 보안 주체에 대한 세부 정보를 정책에 지정한 보안 주체 속성과 비교합니다. 요청할 수 있는 보안 주체의 목록은 보안 주체 지정 섹션을 참조하세요.

목차

aws:PrincipalArn

이 키를 사용하여 요청한 보안 주체의 Amazon 리소스 이름(ARN)을 정책에서 지정한 ARN과 비교합니다. IAM 역할의 경우 요청 컨텍스트는 역할을 수임한 사용자의 ARN이 아니라 역할의 ARN을 반환합니다.

- 가용성 - 이 키는 서명된 모든 요청에 대한 요청 컨텍스트에 포함됩니다. 익명 요청에는 이 키가 포함되지 않습니다. 이 조건 키에서 다음 유형의 보안 주체를 지정할 수 있습니다.
  - IAM 역할
  - IAM 사용자
  - AWS STS 페더레이션 사용자 세션
  - AWS 계정 루트 사용자
- 데이터 유형 – ARN, 문자열

  AWS는 ARN을 비교할 때 문자열 연산자 대신 ARN 연산자를 사용하는 것이 좋습니다.
- 값 유형-단일 값

- 예제 값 다음 목록은 aws:PrincipalArn 조건 키에서 지정할 수 있는 여러 유형의 보안 주체에 대해 반환된 요청 컨텍스트 값을 보여줍니다.
  - IAM 역할 – 요청 컨텍스트에는 조건 키 aws:PrincipalArn에 대한 다음 값이 포함됩니다. 맡은 역할 세션 ARN을 조건 키의 값으로 지정하지 마십시오. 위임된 역할 세션 보안 주체에 대한 자세한 내용은 역할 세션 보안 주체 섹션을 참조하세요.

    ```
    arn:aws:iam::123456789012:role/role-name
    ```

  - IAM 사용자 - 요청 컨텍스트에 조건 키 aws:PrincipalArn에 대한 다음 값이 포함됩니다.

    ```
    arn:aws:iam::123456789012:user/user-name
    ```

  - AWS STS 페더레이션 사용자 세션 - 요청 컨텍스트에 조건 키 aws:PrincipalArn에 대한 다음 값이 포함됩니다.

    ```
    arn:aws:sts::123456789012:federated-user/user-name
    ```

  - AWS 계정 루트 사용자 - 요청 컨텍스트에 조건 키 aws:PrincipalArn에 대한 다음 값이 포함됩니다. 루트 사용자 ARN을 aws:PrincipalArn 조건 키에 대한 값으로 지정하는 경우 AWS 계정의 루트 사용자에 대한 권한만 제한합니다. 이는 리소스 기반 정책의 보안 주체 요소에서 루트 사용자 ARN을 지정하는 것 즉, AWS 계정에 권한을 위임하는 것과는 다릅니다. 리소스 기반 정책의 보안 주체 요소에 루트 사용자 ARN 지정하는 방법에 대한 자세한 내용은 AWS 계정 보안 주체 섹션을 참조하세요.

    ```
    arn:aws:iam::123456789012:root
    ```

루트 사용자 ARN을 AWS Organizations 서비스 제어 정책(SCP)의 조건 키 aws:PrincipalArn의 값으로 지정할 수 있습니다. SCP는 조직의 권한을 관리하는 데 사용되는 조직 정책 유형이며 조직의 구성원 계정에만 영향을 미칩니다. SCP는 멤버 계정의 IAM 사용자 및 역할(멤버 계정의 루트 사용자 포함)에 대해 권한을 제한합니다. SCP가 권한에 미치는 영향에 대한 자세한 내용은 조직 사용 설명서의 SCP가 권한에 미치는 영향을 참조하세요.

aws:PrincipalAccount

이 키를 사용하여 요청한 보안 주체가 속한 계정과 정책에서 지정한 계정 식별자를 비교합니다. 익명 요청의 경우 요청 컨텍스트는 anonymous를 반환합니다.

- 가용성-이 키는 익명 요청을 포함한 모든 요청에 대한 요청 컨텍스트에 포함됩니다.

- 데이터 유형-문자열
- 값 유형-단일 값

다음 예에서는 계정 번호가 123456789012인 보안 주체를 제외하고 액세스가 거부되었습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessFromPrincipalNotInSpecificAccount",
      "Action": "service:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:service:region:accountID:resource"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalAccount": [
            "123456789012"
          ]
        }
      }
    }
  ]
}
```

aws:PrincipalOrgPaths

이 키를 사용하여 요청 중인 보안 주체의 AWS Organizations 경로를 정책의 경로와 비교합니다. 이 보안 주체는 IAM 사용자, IAM 역할, 페더레이션 사용자 또는 AWS 계정 루트 사용자일 수 있습니다. 정책에서 이 조건 키는 요청자가 AWS Organizations의 지정된 조직 루트 또는 조직 단위(OU) 내의 계정 멤버인지 확인합니다. AWS Organizations 경로는 조직 엔터티 구조의 텍스트 표현입니다. 경로 사용 및 이해에 대한 자세한 내용은 AWS Organizations 엔터티 경로 이해 섹션을 참조하세요.

- 가용성-이 키는 보안 주체가 조직의 멤버인 경우에만 요청 컨텍스트에 포함됩니다. 익명 요청에는 이 키가 포함되지 않습니다.
- 데이터 형식-문자열(목록)
- 값 유형-다중 값

> **Note**
>
> 조직 ID는 전역적으로 고유하지만 OU ID와 루트 ID는 조직 내에서만 고유합니다. 즉, 두 조직이 동일한 조직 ID를 공유하지 않습니다. 그러나 다른 조직에는 사용자 ID와 동일한 OU 또는 루트가 있을 수 있습니다. OU 또는 루트를 지정할 때는 항상 조직 ID를 포함하는 것이 좋습니다.

예를 들어, 다음 조건은 `ou-ab12-22222222` OU에 직접 연결되지만 하위 OU에는 연결되지 않은 계정의 보안 주체에 대해 `true`를 반환합니다.

```
"Condition" : { "ForAnyValue:StringEquals" : {
     "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/"]
}}
```

다음 조건은 OU 또는 해당 하위 OU에 직접 연결된 계정의 보안 주체에 대해 `true`를 반환합니다. 와일드카드를 포함할 때는 `StringLike` 조건 연산자를 사용해야 합니다.

```
"Condition" : { "ForAnyValue:StringLike" : {
     "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/
*"]
}}
```

다음 조건은 하위 OU(단 상위 OU에 직접으로는 제외)에 직접 연결된 계정의 보안 주체에 대해 `true`를 반환합니다. 이전 조건은 OU 또는 모든 하위 항목을 위한 것입니다. 다음 조건은 하위 항목(및 해당 하위 항목의 하위 항목)에만 해당됩니다.

```
"Condition" : { "ForAnyValue:StringLike" : {
     "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/
ou-*"]
}}
```

다음 조건에서는 상위 OU와 관계없이 `o-a1b2c3d4e5` 조직의 모든 보안 주체에 대한 액세스를 허용합니다.

```
"Condition" : { "ForAnyValue:StringLike" : {
     "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/*"]
}}
```

aws:PrincipalOrgPaths는 다중 값 조건 키입니다. 다중 값 키는 요청 컨텍스트에서 여러 값을 가질 수 있습니다. ForAnyValue 조건 연산자에서 여러 값을 사용하는 경우 보안 주체의 경로는 정책에 나열된 경로 중 하나와 일치해야 합니다. 다중 값 조건 키에 대한 자세한 내용은 다중 값 컨텍스트 키 섹션을 참조하세요.

```
    "Condition": {
        "ForAnyValue:StringLike": {
            "aws:PrincipalOrgPaths": [
                "o-a1b2c3d4e5/r-ab12/ou-ab12-33333333/*",
                "o-a1b2c3d4e5/r-ab12/ou-ab12-22222222/*"
            ]
        }
    }
```

aws:PrincipalOrgID

이 키를 사용하여 요청한 보안 주체가 속한 AWS Organizations의 조직 식별자와 정책에 지정된 식별자를 비교합니다.

- 가용성-이 키는 보안 주체가 조직의 멤버인 경우에만 요청 컨텍스트에 포함됩니다. 익명 요청에는 이 키가 포함되지 않습니다.
- 데이터 유형-문자열
- 값 유형-단일 값

이 전역 키는 조직 내 모든 AWS 계정의 계정 ID를 전부 나열하는 대안을 제공합니다. 이 조건 키를 사용하여 리소스 기반 정책에서 Principal 요소를 간단하게 지정할 수 있습니다. 조건 요소에서 조직 ID를 지정할 수 있습니다. 계정을 추가 및 제거할 때 aws:PrincipalOrgID 키가 포함된 정책에는 자동으로 올바른 계정이 포함되므로 수동 업데이트가 필요하지 않습니다.

예를 들어, 다음 Amazon S3 버킷 정책을 통해 o-xxxxxxxxxxx 조직 내 모든 계정의 멤버는 policy-ninja-dev 버킷에 객체를 추가할 수 있습니다.

```
{
 "Version": "2012-10-17",
 "Statement": {
   "Sid": "AllowPutObject",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "s3:PutObject",
```

```
    "Resource": "arn:aws:s3:::policy-ninja-dev/*",
    "Condition": {"StringEquals":
      {"aws:PrincipalOrgID":"o-xxxxxxxxxxx"}
    }
  }
}
```

> **Note**
>
> 이 전역 조건은 AWS 조직의 관리 계정에도 적용됩니다. 이 정책은 지정된 조직 외부의 모든 보안 주체가 Amazon S3 버킷에 액세스하지 못하도록 합니다. 여기에는 AWS CloudTrail이(가) Amazon S3 버킷에 로그 데이터를 전송하는 것과 같이 내부 리소스와 상호 작용하는 모든 AWS 서비스가 포함됩니다. AWS 서비스에 대한 액세스 권한을 안전하게 부여하는 방법을 알아보려면 aws:PrincipalIsAWSService 섹션을 참조하세요.

AWS Organizations에 대한 자세한 내용은 AWS Organizations 사용 설명서의 AWS Organizations이란 무엇입니까?를 참조하세요.

aws:PrincipalTag/tag-key

이 키를 사용하여 요청한 보안 주체에 연결된 태그를 정책에서 지정한 태그와 비교합니다. 보안 주체에 둘 이상의 태그가 연결되어 있는 경우 요청 컨텍스트에는 연결된 각 태그 키에 대해 `aws:PrincipalTag` 키가 하나씩 포함됩니다.

- 가용성 - 이 키는 보안 주체가 태그가 연결된 IAM 사용자를 사용하는 경우 요청 컨텍스트에 포함됩니다. 연결된 태그 또는 세션 태그가 있는 IAM 역할을 사용하는 보안 주체에 포함됩니다. 익명 요청에는 이 키가 포함되지 않습니다.
- 데이터 유형-문자열
- 값 유형-단일 값

사용자 또는 역할에 사용자 지정 속성을 키 값 페어의 형태로 추가할 수 있습니다. IAM에서의 태그 사용에 대한 자세한 내용은 AWS Identity and Access Management 리소스용 태그 섹션을 참조하세요. `aws:PrincipalTag`를 사용하여 AWS 보안 주체에 대한 액세스를 제어할 수 있습니다.

이 예제는 **`department=hr`** 태그가 지정된 사용자가 IAM 사용자, 그룹 또는 역할을 관리하도록 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책을 사용하려면 정책 예제의 `#### ## ### ###`를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/department": "hr"
        }
      }
    }
  ]
}
```

aws:PrincipalIsAWSService

이 키를 사용하여 리소스에 대한 호출이 AWS 서비스 보안 주체에 의해 직접 수행되고 있는지를 확인합니다. 예를 들어 AWS CloudTrail은 서비스 보안 주체 `cloudtrail.amazonaws.com`을 사용하여 Amazon S3 버킷에 로그를 작성합니다. 서비스가 서비스 보안 주체를 사용하여 리소스에 대한 직접 작업을 수행하는 경우 요청 컨텍스트 키는 true로 설정됩니다. 서비스가 IAM 보안 주체의 자격 증명을 사용해 보안 주체를 대신하여 요청을 수행하면 요청 컨텍스트 키가 false로 설정됩니다. 서비스가 서비스 역할 또는 서비스 연결 역할을 사용해 보안 주체를 대신하여 호출을 하는 경우에도 false로 설정됩니다.

- 가용성 - 이 키는 AWS 자격 증명을 사용하는 모든 API 요청에 대한 요청 컨텍스트에 표시됩니다. 익명 요청에는 이 키가 포함되지 않습니다.
- 데이터 형식 – 부울
- 값 유형-단일 값

이 조건 키를 사용하여 신뢰할 수 있는 자격 증명 및 예상 네트워크 위치에 대한 액세스를 제한하면서 AWS 서비스에 대한 액세스 권한를 안전하게 부여합니다.

다음 Amazon S3 버킷 정책 예에서 `vpc-111bbb22`에서 시작된 요청 또는 CloudTrail 과 같은 서비스 주체로부터 온 경우를 제외하고 버킷에 대한 액세스는 제한됩니다.

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

```
      "Sid": "Expected-network+service-principal",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/AWSLogs/AccountNumber/*",
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:SourceVpc": "vpc-111bbb22"
        },
        "BoolIfExists": {
          "aws:PrincipalIsAWSService": "false"
        }
      }
    }
  ]
}
```

다음 비디오에서 aws:PrincipalIsAWSService 조건 키를 정책에서 사용하는 방법에 대해 자세히 알아보세요.

권한 있는 사용자, 예상 네트워크 위치 및 AWS 서비스에 대한 액세스 권한을 함께 안전하게 부여합니다.

aws:PrincipalServiceName

이 키를 사용하여 정책의 서비스 보안 주체이름을 리소스에 요청하는 서비스 보안 주체와 비교합니다. 이 키를 사용하여 이 호출이 특정 서비스 보안 주체에 의해 수행되었는지를 확인할 수 있습니다. 서비스 보안 주체가 리소스에 직접 요청하면 aws:PrincipalServiceName 키에는 서비스 보안 주체의 이름이 포함됩니다. 예를 들어 AWS CloudTrail 서비스 보안 주체의 이름은 cloudtrail.amazonaws.com입니다.

- 가용성 - 이 키는 호출이 AWS 서비스 보안 주체에 의해 이루어질 때 요청에 표시됩니다. 이 키는 다음과 같은 다른 상황에서는 나타나지 않습니다.
  - 서비스가 서비스 역할 또는 서비스 연결 역할을 사용해 보안 주체를 대신하여 호출을 하는 경우.
  - 서비스가 IAM 보안 주체의 자격 증명을 사용해 보안 주체를 대신하여 요청을 수행하는 경우.
  - IAM 보안 주체가 직접 호출한 경우.
  - 익명의 요청자가 호출하는 경우.
- 데이터 유형-문자열
- 값 유형-단일 값

이 조건 키를 사용하여 신뢰할 수 있는 자격 증명 및 예상 네트워크 위치에 대한 액세스를 제한하면서 AWS 서비스에 대한 액세스 권한를 안전하게 부여합니다.

다음 Amazon S3 버킷 정책 예에서 vpc-111bbb22에서 시작된 요청 또는 CloudTrail 과 같은 서비스 주체로부터 온 경우를 제외하고 버킷에 대한 액세스는 제한됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "expected-network+service-principal",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/AWSLogs/AccountNumber/*",
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:SourceVpc": "vpc-111bbb22",
          "aws:PrincipalServiceName": "cloudtrail.amazonaws.com"
        }
      }
    }
  ]
}
```

aws:PrincipalServiceNamesList

이 키는 서비스에 속한 모든 서비스 보안 주체 이름의 목록을 제공합니다. 이 키는 고급 조건 키입니다. 이를 사용해 서비스가 특정 리전에서만 리소스에 액세스하지 못하도록 제한할 수 있습니다. 일부 서비스는 특정 리전 내에서 서비스의 특정 인스턴스를 나타내는 리전 서비스 보안 주체를 생성할 수 있습니다. 리소스에 대한 액세스를 서비스의 특정 인스턴스로 제한할 수 있습니다. 서비스 보안 주체가 리소스에 직접 요청하는 경우 aws:PrincipalServiceNamesList에는 서비스의 리전별 인스턴스와 연결된 모든 서비스 보안 주체 이름의 순서가 지정되지 않은 목록이 포함되어 있습니다.

- 가용성 - 이 키는 호출이 AWS 서비스 보안 주체에 의해 이루어질 때 요청에 표시됩니다. 이 키는 다음과 같은 다른 상황에서는 나타나지 않습니다.
  - 서비스가 서비스 역할 또는 서비스 연결 역할을 사용해 보안 주체를 대신하여 호출을 하는 경우.
  - 서비스가 IAM 보안 주체의 자격 증명을 사용해 보안 주체를 대신하여 요청을 수행하는 경우.
  - IAM 보안 주체가 직접 호출한 경우.

  - 익명의 요청자가 호출하는 경우.
- 데이터 형식-문자열(목록)
- 값 유형-다중 값

`aws:PrincipalServiceNamesList`는 다중 값 조건 키입니다. 다중 값 키는 요청 컨텍스트에서 여러 값을 가질 수 있습니다. 이 키에 대한 문자열 조건 연산자와 함께 `ForAnyValue` 또는 `ForAllValues` 집합 연산자를 사용해야 합니다. 다중 값 조건 키에 대한 자세한 내용은 다중 값 컨텍스트 키 섹션을 참조하세요.

aws:PrincipalType

이 키를 사용하여 요청하는 보안 주체의 유형을 정책에서 지정한 보안 주체 유형과 비교합니다. 자세한 내용은 보안 주체 지정 섹션을 참조하세요. `principal` 키 값의 구체적인 예제를 보려면 보안 주체 키 값을(를) 참조하세요.

- 가용성-이 키는 익명 요청을 포함한 모든 요청에 대한 요청 컨텍스트에 포함됩니다.
- 데이터 유형-문자열
- 값 유형-단일 값

aws:userid

이 키를 사용하여 요청자의 보안 주체 식별자를 정책에서 지정한 ID와 비교합니다. IAM 사용자의 경우 요청 컨텍스트 값은 사용자 ID입니다. IAM 역할의 경우 이 값 형식은 다를 수 있습니다. 다른 보안 주체에 대한 정보가 표시되는 방법에 대한 자세한 내용은 보안 주체 지정 섹션을 참조하세요. `principal` 키 값의 구체적인 예제를 보려면 보안 주체 키 값 섹션을 참조하세요.

- 가용성-이 키는 익명 요청을 포함한 모든 요청에 대한 요청 컨텍스트에 포함됩니다.
- 데이터 유형-문자열
- 값 유형-단일 값

aws:username

이 키를 사용하여 요청자의 사용자 이름을 정책에서 지정한 사용자 이름과 비교합니다. 다른 보안 주체에 대한 정보가 표시되는 방법에 대한 자세한 내용은 보안 주체 지정 섹션을 참조하세요. `principal` 키 값의 구체적인 예제를 보려면 보안 주체 키 값 섹션을 참조하세요.

- 가용성 - 이 키는 항상 IAM 사용자의 요청 컨텍스트에 포함됩니다. 익명 요청 및 AWS 계정 루트 사용자 또는 IAM 역할에서 생성된 요청에는 이 키가 포함되지 않습니다. IAM Identity Center 보안 인증을 사용하여 수행된 요청에는 컨텍스트에 이 키가 포함되지 않습니다.
- 데이터 유형-문자열
- 값 유형-단일 값

역할 세션의 속성

다음 조건 키를 사용하여 세션이 생성된 시점의 역할 세션 속성을 비교합니다. 이러한 조건 키는 보안 주체가 역할 세션 또는 페더레이션 사용자 보안 인증 정보를 사용하여 요청하는 경우에만 사용 가능합니다. 이러한 조건 키의 값은 역할의 세션 토큰에 포함됩니다.

역할은 보안 주체의 한 유형입니다. 보안 주체의 속성 섹션의 조건 키를 사용하여 역할이 요청할 때 역할의 속성을 평가할 수도 있습니다.

목차

aws:FederatedProvider

이 키를 사용하여 요청자의 보안 주체 식별자 공급자(IdP)를 정책에서 지정한 IdP와 비교합니다. 이는 `AssumeRoleWithWebIdentity` AWS STS 작업을 사용하여 IAM 역할을 수임했음을 의미합니다. 결과 역할 세션의 임시 자격 증명이 요청을 수행하는 데 사용되는 경우 요청 컨텍스트는 원래 페더레이션 자격 증명을 인증한 IdP를 식별합니다.

- 가용성 - 이 키는 보안 주체가 역할 세션 보안 주체이고 `AssumeRoleWithWebIdentity`를 사용하여 역할이 수임되었을 때 해당 세션이 발급된 경우에 표시됩니다.
- 데이터 유형-문자열
- 값 유형-단일 값

예를 들어 Amazon Cognito를 통해 사용자가 인증된 경우 요청 컨텍스트에 `cognito-identity.amazonaws.com` 값이 포함됩니다. 마찬가지로 Login with Amazon을 통해 사용자가 인증된 경우 요청 컨텍스트에 `www.amazon.com` 값이 포함됩니다.

단일 값 조건 키를 변수로 사용할 수 있습니다. 다음 리소스 기반 정책 예에서는 `aws:FederatedProvider` 키를 리소스 ARN의 정책 변수로 사용합니다. 이 정책은 IdP를 사용하여 인증한 모든 보안 주체가 발급 자격 증명 공급자와 관련된 경로를 사용하여 Amazon S3 버킷에서 객체를 가져올 수 있도록 허용합니다.

aws:TokenIssueTime

이 키를 사용하여 임시 보안 자격 증명이 발급된 날짜와 시간을 정책에서 지정한 날짜 및 시간과 비교할 수 있습니다.

- 가용성-이 키는 보안 주체가 임시 자격 증명을 사용하여 요청한 경우에만 요청 컨텍스트에 포함됩니다. 액세스 키를 사용하는 AWS CLI, AWS API 또는 AWS SDK 요청에는 이 키가 존재하지 않습니다.
- 데이터 형식 - 날짜
- 값 유형-단일 값

임시 자격 증명의 사용을 지원하는 서비스에 대해 알아보려면 AWS IAM으로 작업하는 서비스 섹션을 참조하세요.

aws:MultiFactorAuthAge

이 키를 사용하여 요청한 보안 주체가 MFA를 사용하여 승인된 이후의 시간(초)과 정책에서 지정한 시간을 비교합니다. MFA에 대한 자세한 내용은 IAM의 AWS 다중 인증 섹션을 참조하세요.

> **Important**
>
> 이 조건 키는 페더레이션형 ID 또는 액세스 키를 사용하여 AWS CLI, AWS API 또는 AWS SDK 요청에 서명하기 위해 만든 요청에는 존재하지 않습니다. 임시 보안 자격 증명으로 API 작업에 MFA 보호를 추가하는 방법에 대한 자세한 내용은 MFA를 통한 보안 API 액세스 섹션을 참조하세요.
> IAM 페더레이션형 ID를 검증하는 데 MFA가 사용되는지 확인하려면 ID 제공업체의 인증 방법을 AWS에 세션 태그로 전달합니다. 세부 정보는 AWS STS에서 세션 태그 전달을 참조하세요.
> IAM Identity Center ID에 대해 MFA를 적용하려면 액세스 제어 속성에서 ID 제공업체의 인증 방법과 함께 SAML 어설션 클레임을 IAM Identity Center에 전달할 수 있게 합니다.

- 가용성 - 이 키는 보안 주체가 임시 보안 자격 증명을 사용하여 요청할 때만 요청 컨텍스트에 포함됩니다. MFA 조건이 포함된 정책은 다음에 연결할 수 있습니다.
  - IAM 사용자 또는 그룹
  - Amazon S3 버킷, Amazon SQS 대기열, Amazon SNS 주제 등과 같은 리소스
  - 사용자가 수임할 수 있는 IAM 역할의 신뢰 정책
- 데이터 형식 – 숫자
- 값 유형-단일 값

aws:MultiFactorAuthPresent

이 키를 사용하여 다중 인증(MFA)를 통해 요청을 한 임시 보안 자격 증명의 유효성을 검사했는지 여부를 확인합니다.

> **Important**
>
> 이 조건 키는 페더레이션형 ID 또는 액세스 키를 사용하여 AWS CLI, AWS API 또는 AWS SDK 요청에 서명하기 위해 만든 요청에는 존재하지 않습니다. 임시 보안 자격 증명으로 API 작업에 MFA 보호를 추가하는 방법에 대한 자세한 내용은 MFA를 통한 보안 API 액세스 섹션을 참조하세요.

IAM 페더레이션형 ID를 검증하는 데 MFA가 사용되는지 확인하려면 ID 제공업체의 인증 방법을 AWS에 세션 태그로 전달합니다. 세부 정보는 AWS STS에서 세션 태그 전달을 참조하세요. IAM Identity Center ID에 대해 MFA를 적용하려면 액세스 제어 속성에서 ID 제공업체의 인증 방법과 함께 SAML 어설션 클레임을 IAM Identity Center에 전달할 수 있게 합니다.

- 가용성-이 키는 보안 주체가 임시 자격 증명을 사용하여 요청한 경우에만 요청 컨텍스트에 포함됩니다. MFA 조건이 포함된 정책은 다음에 연결할 수 있습니다.
  - IAM 사용자 또는 그룹
  - Amazon S3 버킷, Amazon SQS 대기열, Amazon SNS 주제 등과 같은 리소스
  - 사용자가 수임할 수 있는 IAM 역할의 신뢰 정책
- 데이터 형식 – 부울
- 값 유형-단일 값

임시 자격 증명은 AssumeRole 또는 GetSessionToken의 임시 토큰을 사용하여 IAM 역할 및 IAM 사용자와 AWS Management Console 사용자를 인증하는 데 사용됩니다.

IAM 사용자 액세스 키는 장기 자격 증명이지만 경우에 따라 AWS는 작업을 수행하기 위해 IAM 사용자를 대신하여 임시 자격 증명을 생성합니다. 이러한 경우 `aws:MultiFactorAuthPresent` 키는 요청에 있으며 `false` 값으로 설정됩니다. 이런 일이 발생할 수 있는 두 가지 일반적인 경우가 있습니다.

- AWS Management Console의 IAM 사용자는 자신도 모르게 임시 자격 증명을 사용합니다. 사용자는 장기 자격 증명인 사용자 이름과 암호를 사용하여 콘솔에 로그인합니다. 하지만 백그라운드에서는 콘솔이 사용자를 대신하여 임시 자격 증명을 생성합니다.
- IAM 사용자가 AWS 서비스를 호출하는 경우 서비스는 사용자의 자격 증명을 다시 사용하여 다른 서비스에 다른 요청을 합니다. 예를 들어 Athena를 호출하여 Amazon S3 버킷에 액세스하거나 AWS CloudFormation을 사용해 Amazon EC2 인스턴스를 생성하는 경우입니다. 후속 요청의 경우 AWS는 임시 자격 증명을 사용합니다.

임시 자격 증명의 사용을 지원하는 서비스에 대해 알아보려면 AWS IAM으로 작업하는 서비스 섹션을 참조하세요.

`aws:MultiFactorAuthPresent` 키는 사용자 액세스 키 페어와 같은 장기 자격 증명으로 API 또는 CLI 명령을 호출하는 경우 존재하지 않습니다. 따라서 이 키를 확인할 때 조건 연산자의 `...IfExists` 버전을 사용하는 것이 좋습니다.

다음 Condition 요소는 MFA를 사용하여 요청을 인증했는지를 확인할 수 있는 신뢰성 있는 방법이 아니라는 점을 이해해야 합니다.

```
#####   WARNING: NOT RECOMMENDED   #####
"Effect" : "Deny",
"Condition" : { "Bool" : { "aws:MultiFactorAuthPresent" : "false" } }
```

Deny 효과, Bool 요소 및 false 값을 이렇게 조합할 경우, MFA를 사용하여 인증 가능하나 인증받지 않은 요청을 거부합니다. 이러한 조합은 MFA의 사용을 지원하는 임시 자격 증명에만 지원됩니다. 이 문은 장기 자격 증명을 사용하는 요청 또는 MFA를 사용하여 인증되는 요청에 대한 액세스를 거부하지 않습니다. 이 예의 로직이 복잡하며 MFA 인증이 실제로 사용되었는지 테스트되지 않으므로 이 예를 사용할 때는 주의해야 합니다.

또한 Deny 효과, Null 요소 및 true의 조합은 동일한 방식으로 작동하며 그 로직이 훨씬 더 복잡하기 때문에 이 조합을 사용하지 말아야 합니다.

권장되는 조합

BoolIfExists 연산자를 사용하여, 요청이 MFA를 사용하여 인증되는지 여부를 확인하는 것이 좋습니다.

```
"Effect" : "Deny",
"Condition" : { "BoolIfExists" : { "aws:MultiFactorAuthPresent" : "false" } }
```

Deny, BoolIfExists 및 false를 조합할 경우, MFA를 사용해 인증되지 않은 요청을 거부합니다. 특히 MFA를 포함하지 않는 임시 자격 증명의 요청을 거부합니다. 또한 액세스 키를 사용하는 AWS CLI 또는 AWS API 작업과 같은 장기 자격 증명을 사용하는 요청을 거부합니다. *IfExists 연산자는 aws:MultiFactorAuthPresent 키의 존재성 및 존재 가능성 여부를 해당 키의 존재 여부로 표시된 대로 확인합니다. MFA를 사용해 인증되지 않은 요청을 거부하려면 이 연산자를 사용합니다. 이 방법이 더욱 안전하기는 하지만 액세스 키를 사용해 AWS CLI 또는 AWS API에 액세스하는 코드나 스크립트를 손상시킬 수 있습니다.

대체 조합

또한 BoolIfExists 연산자를 사용하여 MFA로 인증된 요청 및 장기 자격 증명을 사용하는 AWS CLI 또는 AWS API 요청을 허용할 수 있습니다.

```
"Effect" : "Allow",
"Condition" : { "BoolIfExists" : { "aws:MultiFactorAuthPresent" : "true" } }
```

이 조건은 키가 존재하든 존재하지 않든 마찬가지로 일치합니다. `Allow`, `BoolIfExists` 및 `true`를 조합할 경우, MFA를 사용해 인증된 요청 또는 MFA를 사용해 인증받지 않은 요청을 허용합니다. 이 말은 요청자가 장기 액세스 키를 사용할 경우 AWS CLI, AWS API 및 AWS SDK 작업이 허용된다는 것을 의미합니다. 이러한 조합은 MFA를 포함할 수도 있지만 실제로 포함하지 않는 임시 자격 증명의 요청을 허용하지 않습니다.

IAM 콘솔 비주얼 편집기를 사용해 정책을 생성한 후 MFA 필수(MFA required)를 선택하면 이 조합이 적용됩니다. 이러한 설정에서는 콘솔 액세스를 위해 MFA가 필요하지만 MFA 없이 프로그래밍 방식으로 액세스하는 방법도 있습니다.

또는 MFA를 사용해 인증되는 경우에 한해서 `Bool` 연산자를 사용해 프로그래밍 방식 요청과 콘솔 요청을 허용할 수도 있습니다.

```
"Effect" : "Allow",
"Condition" : { "Bool" : { "aws:MultiFactorAuthPresent" : "true" } }
```

`Allow`, `Bool` 및 `true`를 조합할 경우, MFA를 사용해 인증된 요청만을 허용합니다. 이러한 조합은 MFA의 사용을 지원하는 임시 자격 증명에만 지원됩니다. 이 문은 장기 액세스 키를 사용하는 요청 또는 MFA 없이 임시 자격 증명을 사용하는 요청에 대한 액세스를 허용하지 않습니다.

MFA 키가 있는지 여부를 확인하는 데 다음과 유사한 정책 구문을 사용하지 마십시오.

```
#####   WARNING: USE WITH CAUTION   #####

"Effect" : "Allow",
"Condition" : { "Null" : { "aws:MultiFactorAuthPresent" : "false" } }
```

`Allow` 효과, `Null` 요소 및 `false` 값을 조합할 경우, 그 요청의 실제 인증 여부와 상관없이, MFA를 사용해 인증받을 수 있는 요청만을 허용합니다. 이렇게 하여 임시 자격 증명을 사용하는 모든 요청을 허용하고 장기 자격 증명에 대한 액세스를 거부합니다. 이 예에서는 MFA 인증이 실제로 사용되었는지 여부를 테스트하지 않으므로 이 예를 사용할 때는 주의해야 합니다.

aws:Ec2InstanceSourceVpc

이 키는 Amazon EC2 IAM 역할 보안 인증 정보가 전송된 VPC를 식별합니다. 정책에서 이 키를 `aws:SourceVPC` 글로벌 키와 함께 사용하여 보안 인증 정보가 전송된 VPC(`aws:Ec2InstanceSourceVpc`)와 일치하는 VPC(`aws:SourceVPC`)에서 호출이 이루어졌는지 확인할 수 있습니다.

- 가용성 - 이 키는 요청자가 Amazon EC2 역할 보안 인증 정보를 사용하여 요청에 서명할 때마다 요청 컨텍스트에 포함됩니다. IAM 정책, 서비스 제어 정책, VPC 엔드포인트 정책 및 리소스 정책에 이 키를 사용할 수 있습니다.
- 데이터 유형-문자열
- 값 유형-단일 값

이 키는 VPC 식별자 값과 함께 사용할 수 있지만 `aws:SourceVpc` 컨텍스트 키와 함께 변수로 사용할 때 가장 유용합니다. `aws:SourceVpc` 컨텍스트 키는 요청자가 VPC 엔드포인트를 사용하여 요청한 경우에만 요청 컨텍스트에 포함됩니다. `aws:Ec2InstanceSourceVpc`를 `aws:SourceVpc`와 함께 사용하면, 일반적으로 함께 변경되는 값을 비교하는 `aws:Ec2InstanceSourceVpc`를 더 광범위하게 사용할 수 있습니다.

> **Note**
>
> 이 조건 키는 EC2-Classic에 사용할 수 없습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RequireSameVPC",
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
            "aws:SourceVpc": "${aws:Ec2InstanceSourceVpc}"
        },
        "Null": {
          "ec2:SourceInstanceARN": "false"
        },
        "BoolIfExists": {
          "aws:ViaAWSService": "false"
        }
      }
    }
  ]
}
```

위의 예에서 aws:SourceVpc 값이 aws:Ec2InstanceSourceVpc 값과 일치하지 않으면 액세스가 거부됩니다. 정책 문은 ec2:SourceInstanceARN 조건 키의 존재 여부를 테스트하는 방식으로, Amazon EC2 인스턴스 역할로 사용되는 역할로만 제한됩니다.

이 정책은 사용자의 Amazon EC2 인스턴스 역할을 대신하여 요청을 할 때 aws:ViaAWSService를 사용하여 AWS가 요청을 승인할 수 있도록 합니다. 예를 들어 Amazon EC2 인스턴스에서 암호화된 Amazon S3 버킷에 대한 요청을 하면 Amazon S3가 사용자를 대신하여 AWS KMS를 호출합니다. AWS KMS에 요청할 때 일부 키가 표시되지 않습니다.

aws:Ec2InstanceSourcePrivateIPv4

이 키는 Amazon EC2 IAM 역할 보안 인증 정보가 전송된 기본 탄력적 네트워크 인터페이스 프라이빗 IPv4 주소를 식별합니다. 이 조건 키를 해당 컴패니언 키 aws:Ec2InstanceSourceVpc와 함께 사용하여 전역적으로 고유한 VPC ID와 소스 프라이빗 IP의 조합이 만들어지도록 해야 합니다. 이 키를 aws:Ec2InstanceSourceVpc와 함께 사용하면 보안 인증 정보가 전송된 것과 동일한 프라이빗 IP 주소에서 요청이 이루어졌는지 확인할 수 있습니다.

- 가용성 - 이 키는 요청자가 Amazon EC2 역할 보안 인증 정보를 사용하여 요청에 서명할 때마다 요청 컨텍스트에 포함됩니다. IAM 정책, 서비스 제어 정책, VPC 엔드포인트 정책 및 리소스 정책에 이 키를 사용할 수 있습니다.
- 데이터 형식 - IP 주소
- 값 유형-단일 값

> **Important**
>
> Allow 문에 이 키를 단독으로 사용해서는 안 됩니다. 프라이빗 IP 주소는 정의상 전역적으로 고유하지 않습니다. aws:Ec2InstanceSourcePrivateIPv4 키를 사용할 때마다 aws:Ec2InstanceSourceVpc 키를 사용하여 사용 가능한 Amazon EC2 인스턴스 보안 인증 정보가 있는 VPC를 지정해야 합니다.

> **Note**
>
> 이 조건 키는 EC2-Classic에 사용할 수 없습니다.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Deny",
            "Action":  "*",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:Ec2InstanceSourceVpc": "${aws:SourceVpc}"
                },
                "Null": {
                    "ec2:SourceInstanceARN": "false"
                },
                "BoolIfExists": {
                    "aws:ViaAWSService": "false"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action":  "*",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:Ec2InstanceSourcePrivateIPv4": "${aws:VpcSourceIp}"
                },
                "Null": {
                    "ec2:SourceInstanceARN": "false"
                },
                "BoolIfExists": {
                    "aws:ViaAWSService": "false"
                }
            }
        }
    ]
}
```

aws:SourceIdentity

이 키를 사용하여 보안 주체가 설정한 소스 자격 증명과 정책에 지정한 소스 자격 증명을 비교합니다.

- 가용성 - 이 키는 AWS STS assume-role CLI 명령 또는 AWS STS `AssumeRole` API 작업을 사용해 역할이 수임된 경우 소스 자격 증명이 설정된 후에 요청 컨텍스트에 포함됩니다.
- 데이터 유형-문자열

- 값 유형-단일 값

정책에 이 키를 사용하여, 역할을 수임할 때 소스 자격 증명을 설정한 보안 주체에 의해 AWS에서 작업을 허용할 수 있습니다. 역할의 지정된 소스 자격 증명이 AWS CloudTrail에 표시되는 활동. 이를 통해 관리자는 AWS의 역할로 누가 어떤 작업을 수행했는지 쉽게 확인할 수 있습니다.

sts:RoleSessionName과 달리 소스 자격 증명을 설정한 후에는 값을 변경할 수 없습니다. 이것은 역할에서 수행되는 모든 작업의 요청 컨텍스트에 있습니다. 세션 자격 증명을 사용하여 다른 역할을 수임하는 경우 값이 후속 역할 세션에 유지됩니다. 한 역할에서 다른 역할을 맡는 것을 역할 체인이라고 합니다.

AWS STS assume-role CLI 명령 또는 AWS STS AssumeRole API 작업을 사용해 역할을 수임하는 동안 보안 주체가 소스 자격 증명을 처음 설정할 때 sts:SourceIdentity 키가 요청에 표시됩니다. 소스 자격 증명 집합이 있는 역할 세션에서 수행된 모든 작업에 대한 요청에 aws:SourceIdentity 키가 표시됩니다.

CriticalRole에 대한 계정 111122223333의 다음 역할 신뢰 정책은 Saanvi 또는 Diego로 설정된 소스 자격 증명이 없는 보안 주체가 역할을 수임하는 것을 방지하는 aws:SourceIdentity에 대한 조건을 포함합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AssumeRoleIfSourceIdentity",
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::123456789012:role/CriticalRole"},
            "Action": [
                "sts:AssumeRole",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "StringLike": {
                    "aws:SourceIdentity": ["Saanvi","Diego"]
                }
            }
        }
    ]
}
```

소스 자격 증명 정보에 대한 자세한 내용은 위임된 역할로 수행한 작업 모니터링 및 제어 섹션을 참조하세요.

ec2:RoleDelivery

이 키를 사용하여 서명된 요청의 인스턴스 메타데이터 서비스 버전을 Amazon EC2에 대한 IAM 역할 자격 증명 정보와 비교합니다. 인스턴스 메타데이터 서비스는 특정 요청에 대해 IMDSv2에 고유한 PUT 또는 GET 헤더가 해당 요청에 있는지 여부에 따라 IMDSv1 및 IMDSv2 요청 간을 구별합니다.

- 가용성 - 이 키는 Amazon EC2 인스턴스에서 역할 세션이 생성될 때마다 요청 컨텍스트에 포함됩니다.
- 데이터 형식 – 숫자
- 값 유형-단일 값
- 예제 값 – 1.0, 2.0

로컬 코드 또는 사용자가 IMDSv2를 사용해야 하도록 각 인스턴스에서 인스턴스 메타데이터 서비스(IMDS)를 구성할 수 있습니다. IMDSv2를 사용해야 하도록 지정하면 IMDSv1는 더 이상 작동하지 않습니다.

- 인스턴스 메타데이터 서비스 버전 1(IMDSv1) – 요청/응답 방법
- 인스턴스 메타데이터 서비스 버전 2(IMDSv2) – 세션 지향 방법

인스턴스에서 IMDSv2를 사용하도록 구성하는 방법에 대한 자세한 내용은 인스턴스 메타데이터 옵션 구성을 참조하세요.

다음 예에서는 요청 컨텍스트의 EC2:RoleDelivery 값이 1.0(IMDSv1)인 경우 액세스가 거부됩니다. 이 정책 문은 일반적으로 적용됩니다. Amazon EC2 역할 자격 증명으로 요청에 서명하지 않으면 요청은 아무 효과가 없습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
                {
            "Sid": "RequireAllEc2RolesToUseV2",
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "NumericLessThan": {
```

```
                    "ec2:RoleDelivery": "2.0"
                }
            }
        }
    ]
}
```

자세한 내용은 인스턴스 메타데이터 작업을 위한 정책 예제를 참조하세요.

ec2:SourceInstanceArn

이 키를 사용하여 해당 역할의 세션이 생성된 인스턴스의 ARN을 비교합니다.

- 가용성 - 이 키는 Amazon EC2 인스턴스에서 역할 세션이 생성될 때마다 요청 컨텍스트에 포함됩니다.
- 데이터 형식 - ARN
- 값 유형-단일 값
- 예제 값 – arn:aws:ec2:us-west-2:111111111111:instance/instance-id

정책 예제는 특정 인스턴스에서 다른 AWS 서비스의 리소스를 보는 것을 허용을 참조하세요.

glue:RoleAssumedBy

AWS Glue 서비스는 AWS Glue가 고객 대신(작업이나 개발자 엔드포인트가 아니라 AWS Glue 서비스를 통해 직접) 서비스 역할을 사용하여 요청을 하는 각 AWS API 요청에 대해 이 조건 키를 설정합니다. 이 키를 사용하여 AWS 리소스에 대한 호출이 AWS Glue 서비스로부터 왔는지 확인합니다.

- 가용성 - 이 키는 AWS Glue가 고객 대신 서비스 역할을 사용하여 요청을 할 때 요청 컨텍스트에 포함됩니다.
- 데이터 유형-문자열
- 값 유형-단일 값
- 예제 값 – 이 키는 항상 `glue.amazonaws.com`으로 설정됩니다.

다음 예제에서는 AWS Glue 서비스가 Amazon S3 버킷에서 객체를 가져오도록 허용하는 조건을 추가합니다.

```
{
    "Effect": "Allow",
```

```
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::confidential-bucket/*",
    "Condition": {
        "StringEquals": {
            "glue:RoleAssumedBy": "glue.amazonaws.com"
        }
    }
}
```

glue:CredentialIssuingService

AWS Glue 서비스는 작업 또는 개발자 엔드포인트로부터 오는 서비스 역할을 사용하여 각 AWS API 요청에 대해 이 키를 설정합니다. 이 키를 사용하여 AWS 리소스에 대한 호출이 AWS Glue 작업 또는 개발자 엔드포인트로부터 왔는지 확인합니다.

- 가용성 - 이 키는 AWS Glue가 작업 또는 개발자 엔드포인트로부터 오는 요청을 할 때 요청 컨텍스트에 포함됩니다.
- 데이터 유형-문자열
- 값 유형-단일 값
- 예제 값 – 이 키는 항상 `glue.amazonaws.com`으로 설정됩니다.

다음 예제에서는 AWS Glue 작업에서 사용하는 IAM 역할에 연결된 조건을 추가합니다. 이렇게 하면 역할 세션이 AWS Glue 작업 런타임 환경에 사용되는지 여부에 따라 특정 작업이 허용/거부됩니다.

```
{
    "Effect": "Allow",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::confidential-bucket/*",
    "Condition": {
        "StringEquals": {
            "glue:CredentialIssuingService": "glue.amazonaws.com"
        }
    }
}
```

lambda:SourceFunctionArn

이 키를 사용하여 IAM 역할 보안 인증 정보가 전송된 Lambda 함수 ARN을 식별합니다. Lambda 서비스는 함수의 실행 환경으로부터 오는 각 AWS API 요청에 대해 이 키를 설정합니다. 이 키를 사용하여 AWS 리소스에 대한 호출이 특정 Lambda 함수의 코드로부터 왔는지 확인합니다. 또한 Lambda는

CloudWatch에 로그를 작성하고 X-Ray에 추적을 전송하는 등 실행 환경 외부로부터 온 일부 요청에 대해 이 키를 설정합니다.

- 가용성 - 이 키는 Lambda 함수 코드가 호출될 때마다 요청 컨텍스트에 포함됩니다.
- 데이터 형식 - ARN
- 값 유형-단일 값
- 예제 값 – arn:aws:lambda:us-east-1:123456789012:function:TestFunction

다음 예제에서는 하나의 특정 Lambda 함수에서 `s3:PutObject`가 지정된 버킷에 액세스할 수 있도록 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExampleSourceFunctionArn",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition": {
                "ArnEquals": {
                    "lambda:SourceFunctionArn": "arn:aws:lambda:us-
east-1:123456789012:function:source_lambda"
                }
            }
        }
    ]
}
```

자세한 내용은 AWS Lambda 개발자 안내서의 Working with Lambda execution environment credentials를 참조하세요.

ssm:SourceInstanceArn

이 키를 사용하여 IAM 역할 보안 인증 정보가 전송된 AWS Systems Manager 관리형 인스턴스 ARN을 식별합니다. 요청이 Amazon EC2 인스턴스 프로파일과 연결된 IAM 역할이 있는 관리형 인스턴스로부터 오는 경우에는 이 조건 키가 존재하지 않습니다.

- 가용성 - 이 키는 역할 보안 인증 정보가 AWS Systems Manager 관리형 인스턴스에 전송될 때마다 요청 컨텍스트에 포함됩니다.

- 데이터 형식 - ARN
- 값 유형-단일 값
- 예제 값 – arn:aws:ec2:us-west-2:111111111111:instance/instance-id

identitystore:UserId

이 키를 사용하여 서명된 요청에 있는 IAM Identity Center 인력 ID를 정책에 지정된 ID와 비교합니다.

- 가용성 - 이 키는 요청의 호출자가 IAM Identity Center의 사용자인 경우 포함됩니다.
- 데이터 유형-문자열
- 값 유형-단일 값
- 예제 값 – 94482488-3041-7026-18f3-be45837cd0e4

AWS CLI, AWS API 또는 AWS SDK로 GetUserId에 요청하여 IAM Identity Center에서 사용자의 UserId를 찾을 수 있습니다.

네트워크의 속성

다음 조건 키를 사용하여 요청이 시작되었거나 통과한 네트워크에 대한 세부 정보를 정책에 지정한 네트워크 속성과 비교합니다.

목차

aws:SourceIp

이 키를 사용하여 요청자의 IP 주소를 정책에서 지정한 IP 주소와 비교합니다. `aws:SourceIp` 조건 키는 퍼블릭 IP 주소 범위에만 사용할 수 있습니다.

- 가용성 - 이 키는 요청자가 VPC 엔드포인트를 사용하여 요청한 경우를 제외하고 요청 컨텍스트에 포함됩니다.
- 데이터 형식 - IP 주소
- 값 유형-단일 값

정책 내에서 aws:SourceIp 조건 키를 사용하여 보안 주체가 지정된 IP 범위 내에서만 요청하도록 할 수 있습니다.

> **Note**
>
> aws:SourceIp는 IPv4 및 IPv6 주소 또는 IP 주소 범위를 모두 지원합니다. IPv6를 지원하는 AWS 서비스 목록의 경우 Amazon VPC 사용 설명서의 IPv6를 지원하는 AWS 서비스를 참조하세요.

예를 들어 다음 자격 증명 기반 정책을 IAM 역할에 연결할 수 있습니다. 이 정책은 사용자가 지정된 IPv4 주소 범위에서 호출을 수행하는 경우 객체를 amzn-s3-demo-bucket3 Amazon S3 버킷에 넣을 수 있도록 허용합니다. 또한 이 정책은 AWS 서비스가 사용자를 대신하여 이 작업을 수행하기 위해 전달 액세스 세션을 사용하도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PrincipalPutObjectIfIpAddress",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket3/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "203.0.113.0/24"
                }
            }
        }
    ]
}
```

IPv4 및 IPv6 주소 지정을 모두 지원하는 네트워크에서의 액세스를 제한해야 하는 경우 IAM 정책 조건에 IPv4 및 IPv6 주소 또는 IP 주소 범위를 포함시킬 수 있습니다. 다음 자격 증명 기반 정책은 사용자가 지정된 IPv4 또는 IPv6 주소 범위에서 호출을 수행하는 경우 객체를 amzn-s3-demo-bucket3 Amazon S3 버킷에 넣을 수 있도록 허용합니다. IAM 정책에 IPv6 주소 범위를 포함하기 전에 사용 중인 AWS 서비스가 IPv6를 지원하는지 확인합니다. IPv6를 지원하는 AWS 서비스 목록의 경우 Amazon VPC 사용 설명서의 IPv6를 지원하는 AWS 서비스를 참조하세요.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PrincipalPutObjectIfIpAddress",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket3/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "203.0.113.0/24",
                        "2001:DB8:1234:5678::/64"
                    ]
                }
            }
        }
    ]
}
```

요청이 Amazon VPC 엔드포인트를 사용하는 호스트로부터 오는 경우, aws:SourceIp 키를 사용할 수 없습니다. 대신에 aws:VpcSourceIp와 같은 VPC 전용 키를 사용해야 합니다. VPC 엔드포인트 사용에 대한 자세한 내용을 알아보려면 AWS PrivateLink 가이드의 VPC 엔드포인트 및 VPC 엔드포인트 서비스에 대한 ID 및 액세스 관리를 참조하세요.

aws:SourceVpc

이 키를 사용하여 요청이 VPC 엔드포인트가 연결된 VPC를 통해 이동하는지 확인합니다. 정책에서 이 키를 사용하여 특정 VPC에 대한 액세스만 허용할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 특정 VPC에 대한 액세스 제한을 참조하세요.

- 가용성-이 키는 요청자가 VPC 엔드포인트를 사용하여 요청한 경우에만 요청 컨텍스트에 포함됩니다.
- 데이터 유형-문자열
- 값 유형-단일 값

aws:SourceVpce

이 키를 사용하여 요청의 VPC 엔드포인트 식별자를 정책에서 지정한 엔드포인트 ID와 비교합니다. 정책에서 이 키를 사용하여 특정 VPC에 대한 액세스를 제한할 수 있습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 특정 VPC 엔드포인트에 대한 액세스 제한을 참조하세요.

- 가용성-이 키는 요청자가 VPC 엔드포인트를 사용하여 요청한 경우에만 요청 컨텍스트에 포함됩니다.
- 데이터 유형-문자열
- 값 유형-단일 값

aws:VpcSourceIp

이 키를 사용하여 요청한 IP 주소를 정책에서 지정한 IP 주소와 비교합니다. 정책에서 이 키는 요청이 지정된 IP 주소에서 시작되고 VPC 엔드포인트를 통과하는 경우에만 일치합니다.

- 가용성 - 이 키는 요청이 VPC 엔드포인트를 사용하여 이루어진 경우에만 요청 컨텍스트에 포함됩니다.
- 데이터 형식 - IP 주소
- 값 유형-단일 값

자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

> **Note**
>
> `aws:VpcSourceIp`는 IPv4 및 IPv6 주소 또는 IP 주소 범위를 모두 지원합니다. IPv6를 지원하는 AWS 서비스 목록의 경우 Amazon VPC 사용 설명서의 IPv6를 지원하는 AWS 서비스를 참조하세요.

리소스의 속성

다음 조건 키를 사용하여 요청 대상인 리소스에 대한 세부 정보를 정책에 지정한 리소스 속성과 비교합니다.

목차

- aws:ResourceAccount
- aws:ResourceOrgPaths
- aws:ResourceOrgID
- aws:ResourceTag/tag-key

aws:ResourceAccount

이 키를 사용하여 요청된 리소스 소유자의 AWS 계정 ID를 정책의 리소스 계정과 비교합니다. 그런 다음 리소스를 소유한 계정을 기반으로 해당 리소스에 대한 액세스를 허용하거나 거부할 수 있습니다.

- 가용성 - 이 키는 항상 대다수 서비스 작업의 요청 컨텍스트에 포함됩니다. 다음 작업은 이 키를 지원하지 않습니다.
  - AWS Audit Manager
    - `auditmanager:UpdateAssessmentFrameworkShare`
  - Amazon Detective
    - `detective:AcceptInvitation`
  - Amazon Elastic Block Store – 모든 작업
  - Amazon EC2
    - `ec2:AcceptTransitGatewayPeeringAttachment`
    - `ec2:AcceptVpcEndpointConnections`
    - `ec2:AcceptVpcPeeringConnection`
    - `ec2:CopyImage`
    - `ec2:CopySnapshot`
    - `ec2:CreateTransitGatewayPeeringAttachment`
    - `ec2:CreateVolume`
    - `ec2:CreateVpcEndpoint`
    - `ec2:CreateVpcPeeringConnection`
    - `ec2:DeleteTransitGatewayPeeringAttachment`
    - `ec2:DeleteVpcPeeringConnection`
    - `ec2:RejectTransitGatewayPeeringAttachment`
    - `ec2:RejectVpcEndpointConnections`
    - `ec2:RejectVpcPeeringConnection`
  - Amazon EventBridge
    - `events:PutEvents` – EventBridge `PutEvents`는 해당 이벤트 버스가 2023년 3월 2일 전에 크로스 계정 EventBridge 대상으로 구성된 경우 다른 계정의 이벤트 버스를 호출합니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 Grant permissions to allow events from other AWS accounts를 참조하세요.
  - Amazon GuardDuty

    - `guardduty:AcceptAdministratorInvitation`
  - Amazon Macie
    - `macie2:AcceptInvitation`
  - Amazon OpenSearch Service
    - `es:AcceptInboundConnection`
    - `es:CreateOutboundConnection`
  - Amazon Route 53
    - `route53:AssociateVpcWithHostedZone`
    - `route53:CreateVPCAssociationAuthorization`
    - `route53:DeleteVPCAssociationAuthorization`
    - `route53:DisassociateVPCFromHostedZone`
    - `route53:ListHostedZonesByVPC`
  - AWS Security Hub
    - `securityhub:AcceptAdministratorInvitation`
- 데이터 유형-문자열
- 값 유형-단일 값

> **Note**
>
> 위의 지원되지 않는 작업에 대한 추가 고려 사항은 데이터 경계 정책 예제 리포지토리를 참조하세요.

이 키는 요청에서 평가된 리소스가 있는 계정의 AWS 계정 ID와 같습니다.

계정에 있는 대부분의 리소스에 대해 ARN에는 해당 리소스의 소유자 계정 ID가 포함되어 있습니다. Amazon S3 버킷과 같은 특정 리소스의 경우 리소스 ARN에 계정 ID가 포함되지 않습니다. 다음 두 예제는 ARN에 계정 ID가 있는 리소스와 계정 ID가 없는 Amazon S3 ARN의 차이점을 보여줍니다.

- `arn:aws:iam::123456789012:role/AWSExampleRole` - 계정 123456789012 내에서 생성되고 소유된 IAM 역할.
- `arn:aws:s3:::amzn-s3-demo-bucket2`- 계정 111122223333 내에서 생성되고 소유된 Amazon S3 버킷, ARN에 표시되지 않음

AWS 콘솔, API 또는 CLI를 사용하여 모든 리소스와 해당 ARN 찾기

리소스 소유자의 계정 ID를 기반으로 리소스에 대한 권한을 거부하는 정책을 작성합니다. 예를 들어, 다음 아이덴티티 기반 정책은 리소스가 지정된 계정에 속하지 않는 경우 지정된 리소스에 대한 액세스를 거부합니다.

이 정책을 사용하려면 기울임꼴 자리 표시자 텍스트를 계정 정보로 바꿉니다.

> **Important**
>
> 이 정책은 어떤 작업도 허용하지 않습니다. 대신 이는 나열된 계정에 속하지 않는 문에 나열된 모든 리소스에 대한 액세스를 명시적으로 거부하는 Deny 효과를 사용합니다. 이 정책을 특정 리소스에 대한 액세스를 허용하는 다른 정책과 함께 사용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyInteractionWithResourcesNotInSpecificAccount",
      "Action": "service:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:service:region:account:*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "account"
          ]
        }
      }
    }
  ]
}
```

이 정책은 지정된 AWS 계정이 리소스를 소유하지 않는 한 특정 AWS 서비스의 모든 리소스에 대한 액세스를 거부합니다.

> **Note**
>
> 일부 AWS 서비스은(는) 다른 AWS 계정에서 호스팅되는 AWS 소유 리소스에 액세스해야 합니다. 아이덴티티 기반 정책에서 `aws:ResourceAccount`를 사용하면 이러한 리소스에 대한 아이덴티티의 액세스 가능성에 영향을 있을 수 있습니다.

AWS Data Exchange 등의 특정 AWS 서비스는 정상적인 작업을 위해 AWS 계정 외부의 리소스에 대한 액세스에 의존합니다. 정책에서 `aws:ResourceAccount` 요소를 사용하는 경우 해당 서비스에 대한 예외를 생성하는 추가 문을 포함합니다. 예제 정책 AWS: AWS Data Exchange를 제외한 계정 외부의 Amazon S3 리소스에 대한 액세스 거부은 서비스 소유 리소스에 대한 예외를 정의하면서 리소스 계정을 기반으로 액세스를 거부하는 방법을 보여줍니다.

이 정책 예제를 고유한 사용자 지정 정책 생성을 위한 템플릿으로 사용하세요. 자세한 내용은 서비스 설명서를 참조하세요.

aws:ResourceOrgPaths

이 키를 사용하여 액세스한 리소스의 AWS Organizations 경로를 정책의 경로와 비교합니다. 정책에서 이 조건 키는 리소스가 AWS Organizations의 지정된 조직 루트 또는 조직 단위(OU) 내의 계정 멤버에게 속하도록 합니다. AWS Organizations 경로는 조직 엔터티 구조의 텍스트 표현입니다. 경로 사용 및 이해에 대한 자세한 내용을 알아보려면 AWS Organizations 엔터티 경로 이해 섹션을 참조하세요.

- 가용성 - 이 키는 리소스를 소유한 계정이 조직의 멤버인 경우에만 요청 컨텍스트에 포함됩니다. 이 전역 조건 키는 다음 작업을 지원하지 않습니다.
  - AWS Audit Manager
    - `auditmanager:UpdateAssessmentFrameworkShare`
  - Amazon Detective
    - `detective:AcceptInvitation`
  - Amazon Elastic Block Store – 모든 작업
  - Amazon EC2
    - `ec2:AcceptTransitGatewayPeeringAttachment`
    - `ec2:AcceptVpcEndpointConnections`
    - `ec2:AcceptVpcPeeringConnection`
    - `ec2:CopyImage`
    - `ec2:CopySnapshot`

    - `ec2:CreateTransitGatewayPeeringAttachment`
    - `ec2:CreateVolume`
    - `ec2:CreateVpcEndpoint`
    - `ec2:CreateVpcPeeringConnection`
    - `ec2:DeleteTransitGatewayPeeringAttachment`
    - `ec2:DeleteVpcPeeringConnection`
    - `ec2:RejectTransitGatewayPeeringAttachment`
    - `ec2:RejectVpcEndpointConnections`
    - `ec2:RejectVpcPeeringConnection`
  - Amazon EventBridge
    - `events:PutEvents` – EventBridge `PutEvents`는 해당 이벤트 버스가 2023년 3월 2일 전에 크로스 계정 EventBridge 대상으로 구성된 경우 다른 계정의 이벤트 버스를 호출합니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 Grant permissions to allow events from other AWS accounts를 참조하세요.
  - Amazon GuardDuty
    - `guardduty:AcceptAdministratorInvitation`
  - Amazon Macie
    - `macie2:AcceptInvitation`
  - Amazon OpenSearch Service
    - `es:AcceptInboundConnection`
    - `es:CreateOutboundConnection`
  - Amazon Route 53
    - `route53:AssociateVpcWithHostedZone`
    - `route53:CreateVPCAssociationAuthorization`
    - `route53:DeleteVPCAssociationAuthorization`
    - `route53:DisassociateVPCFromHostedZone`
    - `route53:ListHostedZonesByVPC`
  - AWS Security Hub
    - `securityhub:AcceptAdministratorInvitation`
- 데이터 형식-문자열(목록)
- 값 유형-다중 값

> **Note**
>
> 위의 지원되지 않는 작업에 대한 추가 고려 사항은 데이터 경계 정책 예제 리포지토리를 참조하세요.

`aws:ResourceOrgPaths`는 다중 값 조건 키입니다. 다중 값 키는 요청 컨텍스트에서 여러 값을 가질 수 있습니다. 이 키에 대한 문자열 조건 연산자와 함께 `ForAnyValue` 또는 `ForAllValues` 집합 연산자를 사용해야 합니다. 다중 값 조건 키에 대한 자세한 내용은 다중 값 컨텍스트 키 섹션을 참조하세요.

예를 들어, 다음 조건은 `o-a1b2c3d4e5` 조직에 속한 리소스에 대해 `True`를 반환합니다. 와일드카드를 포함할 때는 StringLike 조건 연산자를 사용해야 합니다.

```
"Condition": {
      "ForAnyValue:StringLike": {
            "aws:ResourceOrgPaths":["o-a1b2c3d4e5/*"]
   }
}
```

다음의 조건은 OU ID가 `ou-ab12-11111111`인 리소스에 대해 `True`를 반환합니다. OU `ou-ab12-11111111` 또는 하위 OU에 연결된 계정에서 소유한 리소스와 일치하게 됩니다.

```
"Condition": { "ForAnyValue:StringLike" : {
     "aws:ResourceOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/*"]
}}
```

다음 조건은 하위 OU가 아닌 OU ID `ou-ab12-22222222`에 직접 연결된 계정이 소유한 리소스에 대해 `True`를 반환합니다. 다음 예제에서는 StringEquals 조건 연산자를 사용하여 와일드카드 일치가 아닌 OU ID에 대한 정확한 일치 요구 사항을 지정합니다.

```
"Condition": { "ForAnyValue:StringEquals" : {
     "aws:ResourceOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/"]
}}
```

> **Note**
>
> 일부 AWS 서비스은(는) 다른 AWS 계정에서 호스팅되는 AWS 소유 리소스에 액세스해야 합니다. 아이덴티티 기반 정책에서 `aws:ResourceOrgPaths`를 사용하면 이러한 리소스에 대한 아이덴티티의 액세스 가능성에 영향을 있을 수 있습니다.

AWS Data Exchange 등의 특정 AWS 서비스는 정상적인 작업을 위해 AWS 계정 외부의 리소스에 대한 액세스에 의존합니다. 정책에서 `aws:ResourceOrgPaths` 키를 사용하는 경우 해당 서비스에 대한 예외를 생성하는 추가 문을 포함합니다. 예제 정책 AWS: AWS Data Exchange를 제외한 계정 외부의 Amazon S3 리소스에 대한 액세스 거부은 서비스 소유 리소스에 대한 예외를 정의하면서 리소스 계정을 기반으로 액세스를 거부하는 방법을 보여줍니다. 유사한 정책을 만들어 서비스 소유 리소스를 고려하면서 `aws:ResourceOrgPaths` 키를 사용하여 조직 단위(OU) 내 리소스에 대한 액세스를 제한할 수 있습니다.

이 정책 예제를 고유한 사용자 지정 정책 생성을 위한 템플릿으로 사용하세요. 자세한 내용은 서비스 설명서를 참조하세요.

aws:ResourceOrgID

이 키를 사용하여 요청된 리소스가 속한 AWS Organizations의 조직 식별자를 정책에 지정된 식별자와 비교합니다.

- 가용성 - 이 키는 리소스를 소유한 계정이 조직의 멤버인 경우에만 요청 컨텍스트에 포함됩니다. 이 전역 조건 키는 다음 작업을 지원하지 않습니다.
  - AWS Audit Manager
    - `auditmanager:UpdateAssessmentFrameworkShare`
  - Amazon Detective
    - `detective:AcceptInvitation`
  - Amazon Elastic Block Store – 모든 작업
  - Amazon EC2
    - `ec2:AcceptTransitGatewayPeeringAttachment`
    - `ec2:AcceptVpcEndpointConnections`
    - `ec2:AcceptVpcPeeringConnection`
    - `ec2:CopyImage`
    - `ec2:CopySnapshot`

    - `ec2:CreateTransitGatewayPeeringAttachment`
    - `ec2:CreateVolume`
    - `ec2:CreateVpcEndpoint`
    - `ec2:CreateVpcPeeringConnection`
    - `ec2:DeleteTransitGatewayPeeringAttachment`
    - `ec2:DeleteVpcPeeringConnection`
    - `ec2:RejectTransitGatewayPeeringAttachment`
    - `ec2:RejectVpcEndpointConnections`
    - `ec2:RejectVpcPeeringConnection`
- Amazon EventBridge
    - `events:PutEvents` – EventBridge `PutEvents`는 해당 이벤트 버스가 2023년 3월 2일 전에 크로스 계정 EventBridge 대상으로 구성된 경우 다른 계정의 이벤트 버스를 호출합니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 Grant permissions to allow events from other AWS accounts를 참조하세요.
- Amazon GuardDuty
    - `guardduty:AcceptAdministratorInvitation`
- Amazon Macie
    - `macie2:AcceptInvitation`
- Amazon OpenSearch Service
    - `es:AcceptInboundConnection`
    - `es:CreateOutboundConnection`
- Amazon Route 53
    - `route53:AssociateVpcWithHostedZone`
    - `route53:CreateVPCAssociationAuthorization`
    - `route53:DeleteVPCAssociationAuthorization`
    - `route53:DisassociateVPCFromHostedZone`
    - `route53:ListHostedZonesByVPC`
- AWS Security Hub
    - `securityhub:AcceptAdministratorInvitation`
- 데이터 유형-문자열
- 값 유형-단일 값

> **Note**
>
> 위의 지원되지 않는 작업에 대한 추가 고려 사항은 데이터 경계 정책 예제 리포지토리를 참조하세요.

이 전역 키는 지정된 요청에 대한 리소스 조직 ID를 반환합니다. 이를 통해 아이덴티티 기반 정책의 `Resource` 요소에 지정된 조직의 모든 리소스에 적용되는 규칙을 생성할 수 있습니다. 조건 요소에서 조직 ID를 지정할 수 있습니다. 계정을 추가하고 제거할 때 `aws:ResourceOrgID` 키가 포함된 정책에 올바른 계정이 자동으로 포함되므로 수동으로 업데이트할 필요가 없습니다.

예를 들어, 다음 정책은 Amazon S3 리소스가 요청하는 보안 주체와 동일한 조직에 속하지 않는 한 보안 주체가 `policy-genius-dev` 리소스에 객체를 추가하지 못하도록 합니다.

> **Important**
>
> 이 정책은 어떤 작업도 허용하지 않습니다. 대신 이는 나열된 계정에 속하지 않는 문에 나열된 모든 리소스에 대한 액세스를 명시적으로 거부하는 `Deny` 효과를 사용합니다. 이 정책을 특정 리소스에 대한 액세스를 허용하는 다른 정책과 함께 사용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "DenyPutObjectToS3ResourcesOutsideMyOrganization",
    "Effect": "Deny",
    "Action": "s3:PutObject",
    "Resource": "arn:partition:s3:::policy-genius-dev/*",
    "Condition": {
      "StringNotEquals": {
        "aws:ResourceOrgID": "${aws:PrincipalOrgID}"
      }
    }
  }
}
```

> **Note**
>
> 일부 AWS 서비스은(는) 다른 AWS 계정에서 호스팅되는 AWS 소유 리소스에 액세스해야 합니다. 아이덴티티 기반 정책에서 `aws:ResourceOrgID`를 사용하면 이러한 리소스에 대한 아이덴티티의 액세스 가능성에 영향을 있을 수 있습니다.

AWS Data Exchange 등의 특정 AWS 서비스는 정상적인 작업을 위해 AWS 계정 외부의 리소스에 대한 액세스에 의존합니다. 정책에서 `aws:ResourceOrgID` 키를 사용하는 경우 해당 서비스에 대한 예외를 생성하는 추가 문을 포함합니다. 예제 정책 AWS: AWS Data Exchange를 제외한 계정 외부의 Amazon S3 리소스에 대한 액세스 거부은 서비스 소유 리소스에 대한 예외를 정의하면서 리소스 계정을 기반으로 액세스를 거부하는 방법을 보여줍니다. 유사한 정책을 만들어 서비스 소유 리소스를 고려하면서 `aws:ResourceOrgID` 키를 사용하여 조직 내 리소스에 대한 액세스를 제한할 수 있습니다.

이 정책 예제를 고유한 사용자 지정 정책 생성을 위한 템플릿으로 사용하세요. 자세한 내용은 서비스 설명서를 참조하세요.

다음 비디오에서 `aws:ResourceOrgID` 조건 키를 정책에서 사용하는 방법에 대해 자세히 알아보세요.

ID와 네트워크는 신뢰할 수 있는 리소스에 액세스하는 데만 사용할 수 있는지 확인합니다..

aws:ResourceTag/tag-key

이 키를 사용하여 정책에서 지정한 태그 키-값 쌍을 리소스에 연결된 키-값 쌍과 비교합니다. 예를 들어 리소스에 값이 `"Marketing"`인 태그 키 `"Dept"`와 연결된 경우에만 리소스에 대한 액세스가 필요할 수 있습니다. 자세한 내용은 AWS 리소스에 대한 액세스 제어 단원을 참조하십시오.

- 가용성 - 이 키는 요청된 리소스에 이미 태그가 연결된 경우 요청 컨텍스트에 포함되거나 태그가 연결된 리소스를 생성하는 요청에 포함됩니다. 이 키는 태그를 기반으로 권한 부여를 지원하는 리소스에 대해서만 반환됩니다. 각 태그 키 값 페어에는 하나의 컨텍스트 키가 있습니다.
- 데이터 유형-문자열
- 값 유형-단일 값

이 컨텍스트 키는 `"aws:ResourceTag/`*`tag-key`*`":"`*`tag-value`*`"` 형식으로, 여기서 *`tag-key`* 및 *`tag-value`*는 태그 키와 값 페어입니다. 태그 키와 값은 대/소문자를 구분하지 않습니다. 따라서 정책의 조건 요소에서 `"aws:ResourceTag/TagKey1": "Value1"` 지정을 완료한 경우 조건은 이름이 `TagKey1` 또는 `tagkey1`인 리소스 태그 키와 일치하지만 두 가지 모두와 일치하지는 않습니다.

`aws:ResourceTag` 키를 사용하여 IAM 리소스에 대한 액세스를 제어하는 예는 AWS 리소스에 대한 액세스 제어 섹션을 참조하세요.

`aws:ResourceTag` 키를 사용하여 다른 AWS 리소스에 대한 액세스를 제어하는 예는 태그를 사용한 AWS 리소스 액세스 제어 섹션을 참조하세요.

속성 기반 액세스 제어(ABAC)에 `aws:ResourceTag` 조건 키를 사용하는 방법에 대한 자습서는 IAM 튜토리얼: 태그를 기반으로 AWS 리소스에 액세스할 수 있는 권한 정의 섹션을 참조하세요.

요청의 속성

다음 조건 키를 사용하여 요청 자체에 대한 세부 정보와 요청 내용을 정책에 지정한 요청 속성과 비교합니다.

목차

aws:CalledVia

이 키를 사용하여 정책의 서비스를 IAM 보안 주체(사용자 또는 역할)를 대신하여 요청한 서비스와 비교합니다. 보안 주체가 AWS 서비스에 요청을 하면 해당 서비스는 보안 주체의 자격 증명을 사용하여 다른 서비스에 대한 후속 요청을 수행할 수 있습니다. `aws:CalledVia` 키에는 보안 주체를 대신하여 요청을 수행한 체인의 각 서비스 목록이 정렬되어 있습니다.

예를 들어 AWS CloudFormation를 사용하여 Amazon DynamoDB 테이블에서 읽고 쓸 수 있습니다. 그런 다음 DynamoDB가 AWS Key Management Service(AWS KMS)에서 제공한 암호화를 사용합니다.

- 가용성 - 이 키는 `aws:CalledVia`를 지원하는 서비스가 IAM 보안 주체의 자격 증명을 사용하여 다른 서비스에 요청을 수행할 때 요청에 표시됩니다. 서비스가 서비스 역할 또는 서비스 연결 역할을 사용해 보안 주체를 대신하여 호출을 하는 경우에는 이 키가 제공되지 않습니다. 이 키는 보안 주체가 직접 호출을 할 때도 존재하지 않습니다.
- 데이터 형식-문자열(목록)
- 값 유형-다중 값

정책에서 `aws:CalledVia` 조건 키를 사용하려면 AWS 서비스 요청을 허용 또는 거부할 서비스 보안 주체를 제공해야 합니다. AWS에서는 `aws:CalledVia`와 함께 다음 서비스 보안 주체를 사용할 수 있도록 지원합니다.

서비스 보안 주체
aoss.amazonaws.com
athena.amazonaws.com
backup.amazonaws.com
cloud9.amazonaws.com
cloudformation.amazonaws.com
databrew.amazonaws.com
dataexchange.amazonaws.com
dynamodb.amazonaws.com

서비스 보안 주체
imagebuilder.amazonaws.com
kms.amazonaws.com
mgn.amazonaws.com
nimble.amazonaws.com
omics.amazonaws.com
ram.amazonaws.com
robomaker.amazonaws.com
servicecatalog-appregistry.amazonaws.com
sqlworkbench.amazonaws.com
ssm-guiconnect.amazonaws.com

어떤 서비스든 보안 주체의 자격 증명을 사용하여 요청을 할 때 액세스를 허용 또는 거부하려면 `aws:ViaAWSService` 조건 키를 사용합니다. 이 조건 키는 모든 AWS 서비스를 지원합니다.

`aws:CalledVia` 키는 다중값 키입니다. 그러나 조건에서 이 키를 사용하여 주문을 수행할 수는 없습니다. 위의 예제를 사용하여 사용자 1이 AWS CloudFormation을 호출하도록 DynamoDB에 요청하면 AWS KMS가 호출됩니다. 이들은 세 가지의 별도 요청입니다. AWS KMS에 대한 마지막 호출은 사용자 1에 의해 AWS CloudFormation 및 DynamoDB를 통해 수행됩니다.

이 경우 요청 컨텍스트의 `aws:CalledVia` 키에는 `cloudformation.amazonaws.com`와 `dynamodb.amazonaws.com`이 순서대로 포함되어 있습니다. 요청 체인의 어딘가에서 DynamoDB를 통해 통화가 이루어졌다는 것만 유념한다면 정책에서 이 조건 키를 사용할 수 있습니다.

예를 들어 다음 정책에서는 `my-example-key`이라는 AWS KMS 키를 관리할 수 있도록 허용하지만, DynamoDB가 요청 서비스 중 하나인 경우에만 한합니다. `ForAnyValue:StringEquals` 조건 연산자는 DynamoDB가 호출하는 서비스 중 하나인지 확인합니다. 보안 주체가 AWS KMS에 직접 호출을 하는 경우에는 조건이 `false`을 반환하고 이 요청이 이 정책에서 허용되지 않습니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "KmsActionsIfCalledViaDynamodb",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:region:111122223333:key/my-example-key",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": ["dynamodb.amazonaws.com"]
                }
            }
        }
    ]
}
```

체인에서 첫 번째 또는 마지막 호출을 하는 서비스를 적용하려는 경우에는 aws:CalledViaFirst 및 aws:CalledViaLast 키를 사용할 수 있습니다. 예를 들어 다음 정책은 AWS KMS에서 my-example-key이라는 키를 관리할 수 있도록 허용합니다. 이러한 AWS KMS 작업은 체인에 여러 요청이 포함된 경우에만 허용됩니다. 첫 번째 요청은 AWS CloudFormation을 통해, 그리고 마지막으로 DynamoDB를 통해 수행되어야 합니다. 다른 서비스가 체인 중간에 요청을 해도 작업은 계속 허용됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "KmsActionsIfCalledViaChain",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey",
                "kms:DescribeKey"
            ],
```

```
            "Resource": "arn:aws:kms:region:111122223333:key/my-example-key",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "cloudformation.amazonaws.com",
                    "aws:CalledViaLast": "dynamodb.amazonaws.com"
                }
            }
        }
    ]
}
```

aws:CalledViaFirst 및 aws:CalledViaLast 키는 서비스가 IAM 보안 주체의 자격 증명을 사용하여 다른 서비스를 호출할 때 요청에 표시됩니다. 이들 키는 요청 체인에서 호출을 한 첫 번째 서비스와 마지막 서비스를 나타냅니다. 예를 들어 AWS CloudFormation이 X Service라는 다른 서비스를 호출하고, 이 서비스는 다시 DynamoDB를 호출한 다음 AWS KMS을 호출한다고 가정해 보겠습니다. AWS KMS에 대한 마지막 호출은 AWS CloudFormation, X Service 및 DynamoDB를 통해 User 1이 수행합니다. 처음 호출은 AWS CloudFormation를 통해, 마지막 호출은 DynamoDB를 통해 이루어졌습니다.

aws:CalledViaFirst

이 키를 사용하여 정책의 서비스를 IAM 보안 주체(사용자 또는 역할)를 대신하여 요청을 수행한 첫 번째 서비스와 비교합니다. 자세한 내용은 aws:CalledVia 단원을 참조하십시오.

- 가용성 - 이 키는 서비스가 IAM 보안 주체의 자격 증명을 사용하여 다른 서비스에 최소 1개의 요청을 수행할 때 요청에 표시됩니다. 서비스가 서비스 역할 또는 서비스 연결 역할을 사용해 보안 주체를 대신하여 호출을 하는 경우에는 이 키가 제공되지 않습니다. 이 키는 보안 주체가 직접 호출을 할 때도 존재하지 않습니다.
- 데이터 유형-문자열
- 값 유형-단일 값

aws:CalledViaLast

이 키를 사용하여 정책의 서비스를 IAM 보안 주체(사용자 또는 역할)를 대신하여 요청을 수행한 마지막 서비스와 비교합니다. 자세한 내용은 aws:CalledVia 단원을 참조하십시오.

- 가용성 - 이 키는 서비스가 IAM 보안 주체의 자격 증명을 사용하여 다른 서비스에 최소 1개의 요청을 수행할 때 요청에 표시됩니다. 서비스가 서비스 역할 또는 서비스 연결 역할을 사용해 보안 주체를

대신하여 호출을 하는 경우에는 이 키가 제공되지 않습니다. 이 키는 보안 주체가 직접 호출을 할 때도 존재하지 않습니다.

- 데이터 유형-문자열
- 값 유형-단일 값

aws:ViaAWSService

이 키를 사용하여 AWS 서비스가 사용자를 대신하여 다른 서비스에 요청하는지 여부를 확인합니다.

서비스가 IAM 보안 주체의 자격 증명을 사용해 보안 주체를 대신하여 요청을 수행하면 요청 컨텍스트 키에서 `true`를 반환합니다. 서비스가 서비스 역할 또는 서비스 연결 역할을 사용해 보안 주체를 대신하여 호출을 하는 경우에는 컨텍스트 키에서 `false`를 반환합니다. 보안 주체가 직접 호출을 할 때도 요청 컨텍스트 키에서 `false`를 반환합니다.

- 가용성-이 키는 항상 요청 컨텍스트에 포함됩니다.
- 데이터 형식 – 부울
- 값 유형-단일 값

이 조건 키를 사용하여 서비스에 의해 요청이 수행되었는지 여부에 따라 액세스를 허용하거나 거부할 수 있습니다.

aws:CurrentTime

이 키를 사용하여 요청의 날짜 및 시간을 정책에서 지정한 날짜 및 시간과 비교합니다. 이 조건 키를 사용하는 예시 정책을 보려면 AWS: 날짜 및 시간에 따라 액세스 허용 섹션을 참조하세요.

- 가용성-이 키는 항상 요청 컨텍스트에 포함됩니다.
- 데이터 형식 - 날짜
- 값 유형-단일 값

aws:EpochTime

이 키를 사용하여 epoch 또는 Unix 시간의 요청 날짜 및 시간을 정책에서 지정한 값과 비교합니다. 또한 이 키는 1970년 1월 1일 이후의 초 수를 허용합니다.

- 가용성-이 키는 항상 요청 컨텍스트에 포함됩니다.
- 데이터 형식 - 날짜, 숫자

- 값 유형-단일 값

aws:referer

이 키를 사용하여 클라이언트 브라우저에서 요청을 참조한 사람과 정책에서 지정한 참조자를 비교합니다. `aws:referer` 요청 컨텍스트 값은 HTTP 헤더의 호출자에 의해 제공됩니다. 웹 페이지에서 링크를 선택하면 웹 브라우저 요청에 `Referer` 헤더가 포함됩니다. `Referer` 헤더에는 링크가 선택된 웹 페이지의 URL이 포함됩니다.

- 가용성 - 이 키는 브라우저의 웹 페이지 URL에서 링크하여 AWS 리소스에 대한 요청을 호출한 경우에만 요청 컨텍스트에 포함됩니다. 이 키는 AWS 리소스에 액세스할 때 브라우저 링크를 사용하지 않기 때문에 프로그래밍 방식의 요청에 포함되지 않습니다.
- 데이터 유형-문자열
- 값 유형-단일 값

예를 들어 URL을 사용하거나 직접 API 호출을 사용하여 Amazon S3 객체에 직접 액세스할 수 있습니다. 자세한 내용은 웹 브라우저를 직접 사용한 Amazon S3 API 작업을 참조하세요. 웹 페이지에 있는 URL에서 Amazon S3 객체에 액세스하면 소스 웹 페이지의 URL이 `aws:referer`에 사용됩니다. 브라우저에 URL을 입력하여 Amazon S3 객체에 액세스하는 경우에는 `aws:referer`가 없습니다. API를 직접 호출하는 경우에도 `aws:referer`가 없습니다. 정책에서 `aws:referer` 조건 키를 사용하여 회사 도메인의 웹 페이지에 대한 링크와 같은 특정 참조자의 요청을 허용할 수 있습니다.

> **Warning**
>
> 이 키를 사용할 때는 주의해야 합니다. 공개적으로 알려진 참조자 헤더 값을 포함하는 것은 위험합니다. 권한이 없는 사용자가 수정된 브라우저나 사용자 지정 브라우저를 사용하여 원하는 `aws:referer` 값을 제공할 수 있습니다. 따라서 무단 사용자의 직접 AWS 요청을 차단할 목적으로 `aws:referer`을 사용해서는 안 됩니다. 이러한 값은 고객이 Amazon S3에 저장된 콘텐츠 등의 디지털 콘텐츠를 권한이 없는 타사 사이트에서 참조하지 못하도록 보호하기 위해서만 사용하세요.

aws:RequestedRegion

이 키를 사용하여 요청에서 호출된 AWS 리전을 정책에서 지정한 리전과 비교합니다. 이 전역 조건 키를 사용하여 요청할 수 있는 리전을 제어할 수 있습니다. 각 서비스의 AWS 리전을 보려면 Amazon Web Services 일반 참조의 서비스 엔드포인트 및 할당량을 참조하세요.

- 가용성-이 키는 항상 요청 컨텍스트에 포함됩니다.
- 데이터 유형-문자열
- 값 유형-단일 값

IAM 등과 같은 일부 전역 서비스에는 단일 엔드포인트가 있습니다. 엔드포인트는 미국 동부(버지니아 북부) 리전에 실제로 위치하기 때문에 IAM 호출은 항상 us-east-1 리전에 대해 생성됩니다. 예를 들어, 요청된 리전이 us-west-2가 아닌 경우 모든 서비스에 대한 액세스를 거부하는 정책을 생성하면 IAM 호출이 항상 실패합니다. 이 문제에 대한 해결 방법을 보여주는 예는 NotAction 및 Deny 섹션을 참조하세요.

> **Note**
>
> `aws:RequestedRegion` 조건 키를 사용하면 서비스의 어떤 엔드포인트를 호출할지 제어할 수 있지만 작업의 영향은 제어할 수 없습니다. 일부 서비스의 경우 교차 리전 영향이 있습니다. 예를 들어, Amazon S3에는 여러 리전으로 확장되는 API 작업이 있습니다.
>
> - `s3:PutBucketReplication` 조건 키의 영향을 받는 한 리전에서 `aws:RequestedRegion`을 호출할 수 있는데 다른 리전은 복제 구성 설정에 따라 영향을 받습니다.
> - `s3:CreateBucket`을 간접적으로 호출하여 다른 리전에서 버킷을 생성하고, `s3:LocationConstraint` 조건 키를 사용하여 해당 리전을 제어할 수 있습니다.

이 컨텍스트 키를 사용하여 지정된 리전 세트 내에서 AWS 서비스에 대한 액세스를 제한할 수 있습니다. 예를 들어, 다음 정책은 사용자가 AWS Management Console에서 모든 Amazon EC2 인스턴스를 조회하도록 허용합니다. 그러나 이 정책은 아일랜드(eu-west-1), 런던(eu-west-2) 또는 파리(eu-west-3)의 인스턴스만 변경하도록 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "InstanceConsoleReadOnly",
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "ec2:Export*",
                "ec2:Get*",
```

```
                "ec2:Search*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "InstanceWriteRegionRestricted",
            "Effect": "Allow",
            "Action": [
                "ec2:Associate*",
                "ec2:Import*",
                "ec2:Modify*",
                "ec2:Monitor*",
                "ec2:Reset*",
                "ec2:Run*",
                "ec2:Start*",
                "ec2:Stop*",
                "ec2:Terminate*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": [
                        "eu-west-1",
                        "eu-west-2",
                        "eu-west-3"
                    ]
                }
            }
        }
    ]
}
```

aws:RequestTag/tag-key

이 키를 사용하여 요청에서 전달된 태그 키 값 페어를 정책에서 지정한 태그 페어와 비교합니다. 예를 들어, 요청에 태그 키 "Dept"가 포함되어 있으며 값이 "Accounting"인지 확인할 수 있습니다. 자세한 내용은 AWS 요청 중 액세스 제어 단원을 참조하십시오.

- 가용성 - 이 키는 태그 키-값 페어가 요청에 전달될 때 요청 컨텍스트에 포함됩니다. 요청에 여러 태그가 전달되면 각 태그 키 값 페어에 대해 하나의 컨텍스트 키가 있습니다.
- 데이터 유형-문자열
- 값 유형-단일 값

이 컨텍스트 키는 "aws:RequestTag/*tag-key*":"*tag-value*" 형식으로, 여기서 *tag-key* 및 *tag-value*는 태그 키와 값 페어입니다. 태그 키와 값은 대/소문자를 구분하지 않습니다. 따라서 정책의 조건 요소에서 "aws:RequestTag/TagKey1": "Value1" 지정을 완료한 경우 조건은 이름이 TagKey1 또는 tagkey1인 요청 태그 키와 일치하지만 두 가지 모두와 일치하지는 않습니다.

이 예제는 키가 단일 값이지만 키가 다른 경우 요청에 여러 키-값 페어를 사용할 수 있음을 보여줍니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:::instance/*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/environment": [
          "preprod",
          "production"
        ],
        "aws:RequestTag/team": [
          "engineering"
        ]
      }
    }
  }
}
```

aws:TagKeys

이 키를 사용하여 요청의 태그 키를 정책에서 지정한 키와 비교합니다. 정책을 사용하여 태그를 통해 액세스를 제어할 때 aws:TagKeys 조건 키를 사용하여 어떤 태그 키가 허용되는지 정의하는 것이 좋습니다. 정책 예제 및 자세한 내용은 the section called “태그 키를 기반으로 액세스 제어” 섹션을 참조하세요.

- 가용성 - 작업이 요청에서 태그 전달을 지원하는 경우 이 키는 요청 컨텍스트에 포함됩니다.
- 데이터 형식-문자열(목록)
- 값 유형-다중 값

이 컨텍스트 키는 "aws:TagKeys":"*tag-key*" 형식이며, 여기서 *tag-key*는 값이 없는 태그 키 목록입니다(예: ["Dept","Cost-Center"]).

요청에 여러 개의 태그 키 값 페어를 포함할 수 있으므로 요청 콘텐츠는 다중 값 요청이 될 수 있습니다. 이 경우 `ForAllValues` 또는 `ForAnyValue` 설정 연산자를 사용해야 합니다. 자세한 내용은 다중 값 컨텍스트 키 단원을 참조하십시오.

일부 서비스는 리소스 생성, 수정 또는 삭제와 같은 리소스 작업을 포함한 태그 지정을 지원합니다. 태그 지정 및 단일 호출과 같은 작업을 허용하려면 태그 지정 작업 및 리소스 수정 작업을 모두 포함하는 정책을 생성해야 합니다. 그런 다음 `aws:TagKeys` 조건 키를 사용하여 요청 내 특정 태그 키 사용을 적용할 수 있습니다. 예를 들어 누군가 Amazon EC2 스냅샷을 생성할 때 태그를 제한하려면 `ec2:CreateSnapshot` 생성 작업 및 `ec2:CreateTags` 태그 지정 작업을 정책에 포함시켜야 합니다. `aws:TagKeys`를 사용하는 이 시나리오에 대한 정책을 보려면 Amazon EC2 사용 설명서의 태그를 사용하여 스냅샷 생성을 참조하세요.

aws:SecureTransport

이 키를 사용하여 요청이 SSL을 사용하여 전송되었는지 여부를 확인합니다. 요청 컨텍스트는 `true` 또는 `false`를 반환합니다. 정책에서 SSL을 사용하여 요청이 전송된 경우에만 특정 작업을 허용할 수 있습니다.

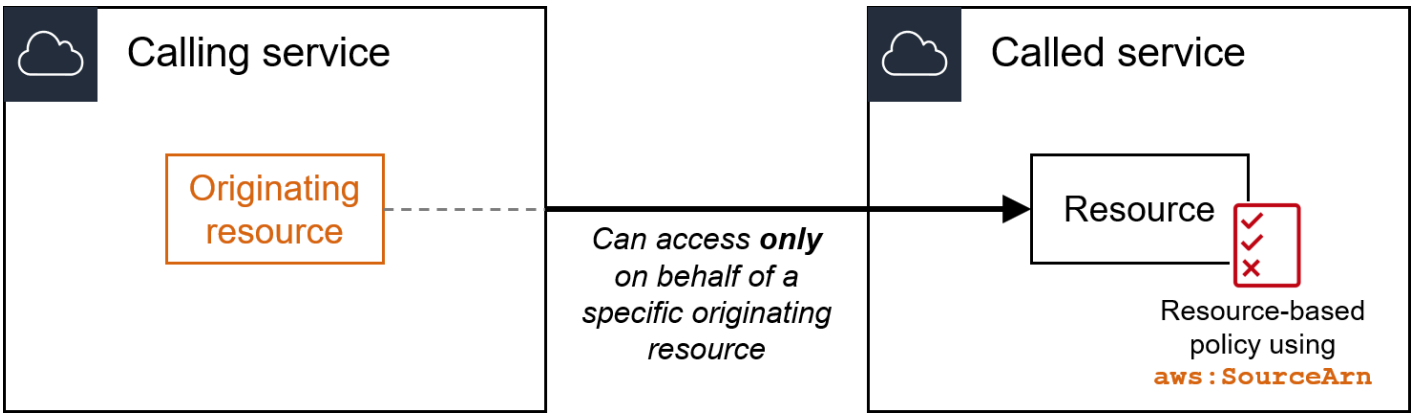
- 가용성-이 키는 항상 요청 컨텍스트에 포함됩니다.
- 데이터 형식 – 부울
- 값 유형-단일 값

aws:SourceArn

이 키를 사용하여 서비스 간 요청을 하는 리소스의 Amazon Resource Name(ARN)을 정책에 지정한 ARN과 비교합니다. 단, AWS 서비스 보안 주체가 요청을 하는 경우에만 해당됩니다. 소스의 ARN에는 계정 ID가 포함되어 있는 경우 `aws:SourceAccount`와 함께 `aws:SourceArn`을 사용할 필요가 없습니다.

이 키는 요청하는 보안 주체의 ARN에서는 작동하지 않습니다. 대신 `aws:PrincipalArn`를 사용합니다.

- 가용성 - 이 키는 구성에서 서비스 간 요청을 트리거한 리소스를 대신하여 AWS서비스 주체가 리소스를 직접 호출하는 경우에만 요청 컨텍스트에 포함됩니다. 호출하는 서비스는 원래 리소스의 ARN을 호출된 서비스로 전달합니다.



다음 서비스 통합은 이 전역 조건 키를 지원하지 않습니다.

호출하는 서비스(서비스 주체)	호출 서비스(리소스 기반 정책)	설명
logdelivery.elb.amazonaws.com	Amazon S3 버킷	Amazon S3 버킷에서 Elastic Load Balancing 액세스 로깅을 활성화합니다.
logdelivery.elasticloadbalancing.amazonaws.com	Amazon S3 버킷	Amazon S3 버킷에서 Elastic Load Balancing 액세스 로깅을 활성화합니다.

> **Note**
>
> AWS Security Token Service(AWS STS) 및 AWS Key Management Service (AWS KMS)와의 모든 서비스 통합이 지원되는 것은 아닙니다. 자세한 내용은 호출하는 서비스의 설명서를 참조하세요. KMS 키 부여를 통해 AWS 서비스가 사용하는 키에 KMS 키 정책에서 `aws:SourceArn`을 사용하면 예상치 못한 동작이 발생할 수 있습니다.

- 데이터 유형 – ARN, 문자열

  AWS는 ARN을 비교할 때 문자열 연산자 대신 ARN 연산자를 사용하는 것이 좋습니다.

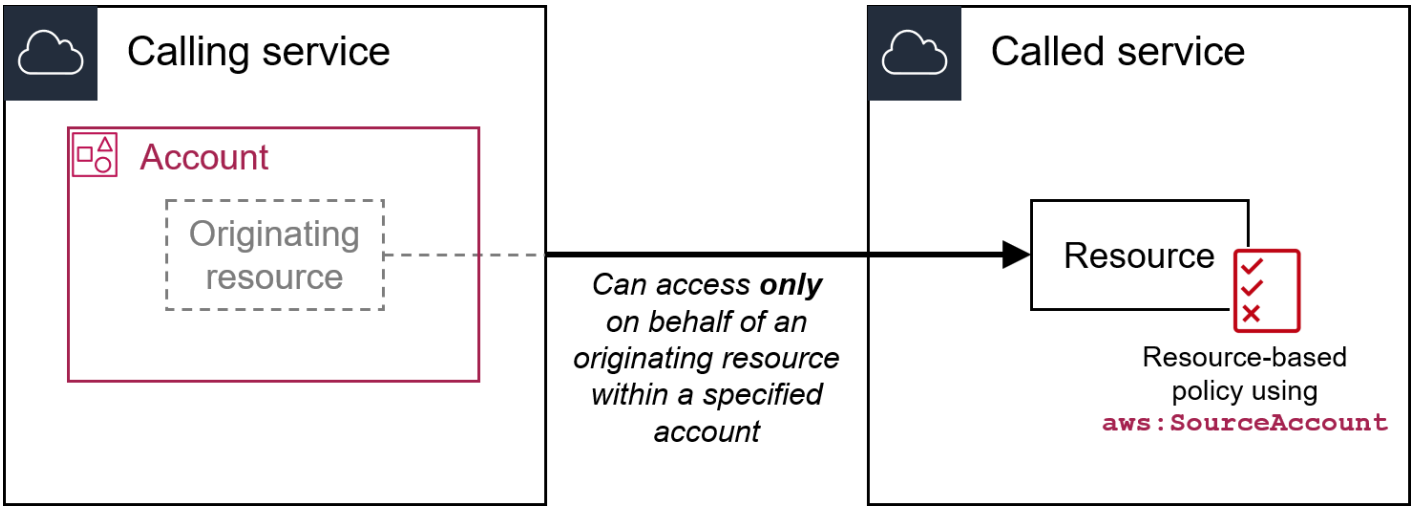
- 값 유형-단일 값

이 조건 키를 사용하여 AWS 서비스가 서비스간 트랜잭션 중에 혼동된 대리자로 사용되는 것을 방지합니다. 이 키는 `Principal`가 AWS 서비스 보안 주체인 리소스 기반 정책에서만 사용합니다. 이 조건 키의 값을 요청에 있는 리소스의 ARN으로 설정합니다. 예를 들어, Amazon S3 버킷 업데이트가 Amazon SNS 주제 게시물을 트리거하면 Amazon S3 서비스에서 `sns:Publish` API 작업을 호출합니다. `sns:Publish` 작업을 허용하는 주제 정책에서 조건 키의 값을 Amazon S3 버킷의 ARN으로 설정합니다. 이러한 조건 키가 권장되는 방법 및 시기에 대한 자세한 내용은 사용하시는 AWS 서비스의 설명서를 참조하세요.

aws:SourceAccount

이 키를 사용하여 서비스 간 요청을 하는 리소스의 계정 ID를 정책에서 지정한 계정 ID와 비교합니다. 단, AWS 서비스 주체가 요청을 하는 경우에만 해당됩니다.

- 가용성 - 이 키는 구성에서 서비스 간 요청을 트리거한 리소스를 대신하여 AWS서비스 주체가 리소스를 직접 호출하는 경우에만 요청 컨텍스트에 포함됩니다. 호출하는 서비스는 원래 리소스 계정 ID를 호출된 서비스로 전달합니다.



다음 서비스 통합은 이 전역 조건 키를 지원하지 않습니다.

호출하는 서비스(서비스 주체)	호출 서비스(리소스 기반 정책)	설명
logdelivery.elb.amazonaws.com	Amazon S3 버킷	Amazon S3 버킷에서 Elastic Load Balancing 액세스 로깅을 활성화합니다.

호출하는 서비스(서비스 주체)	호출 서비스(리소스 기반 정책)	설명
logdelivery.elasticloadbalancing.amazonaws.com	Amazon S3 버킷	Amazon S3 버킷에서 Elastic Load Balancing 액세스 로깅을 활성화합니다.

> **Note**
>
> AWS Security Token Service(AWS STS) 및 AWS Key Management Service (AWS KMS) 와의 모든 서비스 통합이 지원되는 것은 아닙니다. 자세한 내용은 호출하는 서비스의 설명서를 참조하세요. KMS 키 부여를 통해 AWS 서비스가 사용하는 키에 KMS 키 정책에서 `aws:SourceAccount`을 사용하면 예상치 못한 동작이 발생할 수 있습니다.

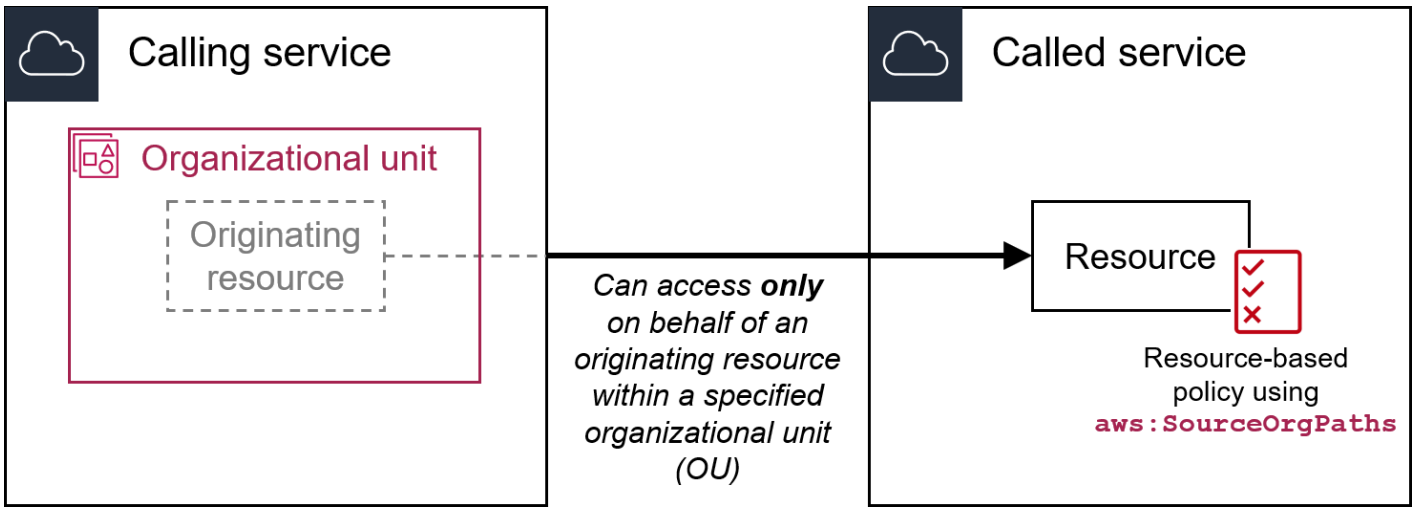
- 데이터 유형-문자열
- 값 유형-단일 값

이 조건 키를 사용하여 AWS 서비스가 서비스간 트랜잭션 중에 혼동된 대리자로 사용되는 것을 방지합니다. 이 키는 `Principal`가 AWS 서비스 보안 주체인 리소스 기반 정책에서만 사용합니다. 이 조건 키의 값을 요청에 있는 리소스의 계정 ID로 설정합니다. 예를 들어, Amazon S3 버킷 업데이트가 Amazon SNS 주제 게시물을 트리거하면 Amazon S3 서비스에서 `sns:Publish` API 작업을 호출합니다. `sns:Publish` 작업을 허용하는 주제 정책에서 조건 키의 값을 Amazon S3 버킷의 계정 ID로 설정합니다. 이러한 조건 키가 권장되는 방법 및 시기에 대한 자세한 내용은 사용하시는 AWS 서비스의 설명서를 참조하세요.

aws:SourceOrgPaths

이 키를 사용하여 서비스 대 서비스 요청을 하는 리소스의 AWS Organizations 경로를 정책에서 지정한 조직의 경로와 비교합니다. 단, 서비스 대 서비스 요청을 하는 조직의 경로를 AWS 서비스 보안 주체가 요청한 경우에만 비교합니다. Organizations 경로는 조직 엔터티 구조의 텍스트 표현입니다. 경로 사용 및 이해에 대한 자세한 내용은 AWS Organizations 엔터티 경로 이해하기 섹션을 참조하세요.

- 가용성 - 이 키는 조직의 구성원인 계정이 소유한 리소스를 대신하여 AWS서비스 보안 주체가 사용자의 리소스를 직접 호출하는 경우에만 요청 컨텍스트에 포함됩니다. 호출하는 서비스는 원래 리소스의 조직 경로를 호출된 서비스로 전달해야 합니다.



다음 서비스 통합은 이 전역 조건 키를 지원하지 않습니다.

호출하는 서비스(서비스 주체)	호출 서비스(리소스 기반 정책)	설명
logdelivery.elb.amazonaws.com	Amazon S3 버킷	Amazon S3 버킷에서 Elastic Load Balancing 액세스 로깅을 활성화합니다.
logdelivery.elasticloadbalancing.amazonaws.com	Amazon S3 버킷	Amazon S3 버킷에서 Elastic Load Balancing 액세스 로깅을 활성화합니다.
모든 서비스 보안 주체	Amazon Lex 봇	AWS 서비스가 Amazon Lex 봇을 사용하도록 허용

> **Note**
>
> AWS Security Token Service(AWS STS) 및 AWS Key Management Service (AWS KMS)와의 모든 서비스 통합이 지원되는 것은 아닙니다. 자세한 내용은 호출하는 서비스의 설명서를 참조하세요. KMS 키 부여를 통해 AWS 서비스가 사용하는 키에 KMS 키 정책에서 `aws:SourceOrgPaths`을 사용하면 예상치 못한 동작이 발생할 수 있습니다.

- 데이터 형식-문자열(목록)
- 값 유형-다중 값

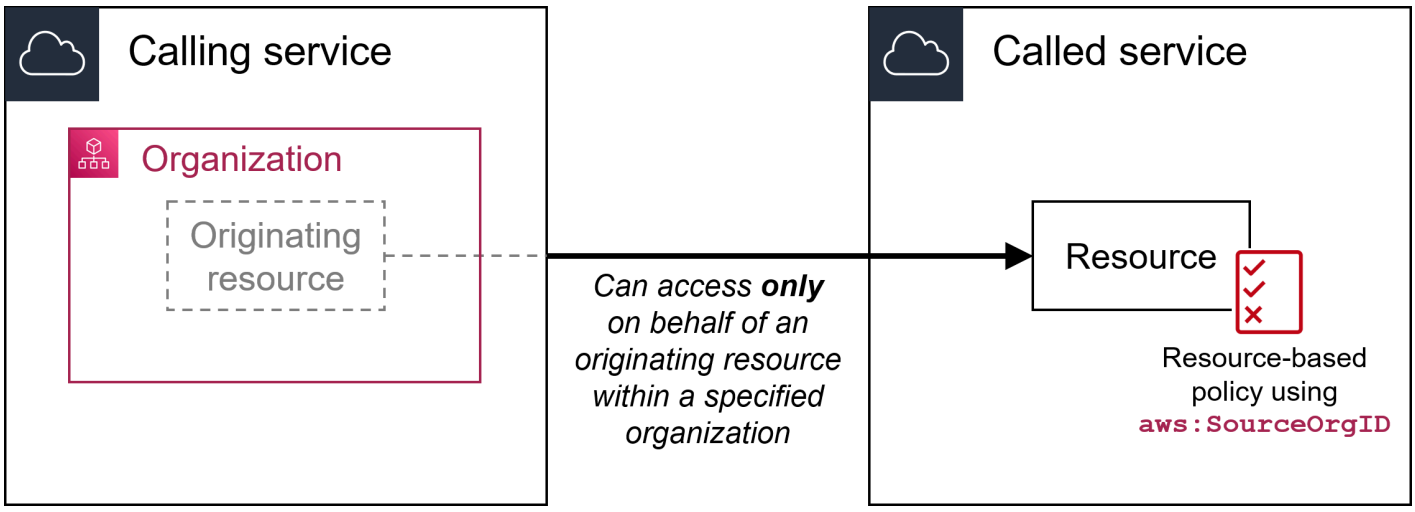
이 조건 키를 사용하여 AWS 서비스가 서비스간 트랜잭션 중에 혼동된 대리자로 사용되는 것을 방지합니다. 이 키는 `Principal`가 AWS 서비스 보안 주체인 리소스 기반 정책에서만 사용합니다. 이 조건 키의 값을 요청에 있는 리소스의 조직 경로로 설정합니다. 예를 들어, Amazon S3 버킷 업데이트가 Amazon SNS 주제 게시물을 트리거하면 Amazon S3 서비스에서 `sns:Publish` API 작업을 호출합니다. `sns:Publish` 작업을 허용하는 주제 정책에서 조건 키의 값을 Amazon S3 버킷의 조직 경로로 설정합니다. 이러한 조건 키가 권장되는 방법 및 시기에 대한 자세한 내용은 사용하시는 AWS 서비스의 설명서를 참조하세요.

`aws:SourceOrgPaths`는 다중 값 조건 키입니다. 다중 값 키는 요청 컨텍스트에서 여러 값을 가질 수 있습니다. 이 키에 대한 문자열 조건 연산자와 함께 `ForAnyValue` 또는 `ForAllValues` 집합 연산자를 사용해야 합니다. 다중 값 조건 키에 대한 자세한 내용은 다중 값 컨텍스트 키 섹션을 참조하세요.

aws:SourceOrgID

이 키를 사용하여 서비스 대 서비스 요청을 하는 리소스의 조직 ID를 정책에서 지정한 조직 ID와 비교합니다. 단, 서비스 대 서비스 요청을 하는 리소스의 조직 ID를 AWS 서비스 보안 주체가 요청한 경우에만 비교합니다. AWS Organizations에서 조직에 계정을 추가하고 제거하면 `aws:SourceOrgID` 키가 포함된 정책에 올바른 계정이 자동으로 포함되므로 정책을 수동으로 업데이트할 필요가 없습니다.

- 가용성 - 이 키는 조직의 구성원인 계정이 소유한 리소스를 대신하여 AWS서비스 보안 주체가 사용자의 리소스를 직접 호출하는 경우에만 요청 컨텍스트에 포함됩니다. 호출하는 서비스는 원래 리소스의 조직 ID를 호출된 서비스로 전달해야 합니다.



다음 서비스 통합은 이 전역 조건 키를 지원하지 않습니다.

호출하는 서비스(서비스 주체)	호출 서비스(리소스 기반 정책)	설명
logdelivery.elb.amazonaws.com	Amazon S3 버킷	Amazon S3 버킷에서 Elastic Load Balancing 액세스 로깅을 활성화합니다.
logdelivery.elasticloadbalancing.amazonaws.com	Amazon S3 버킷	Amazon S3 버킷에서 Elastic Load Balancing 액세스 로깅을 활성화합니다.
모든 서비스 보안 주체	Amazon Lex 봇	AWS 서비스가 Amazon Lex 봇을 사용하도록 허용

> **Note**
>
> AWS Security Token Service(AWS STS) 및 AWS Key Management Service (AWS KMS)와의 모든 서비스 통합이 지원되는 것은 아닙니다. 자세한 내용은 호출하는 서비스의 설명서를 참조하세요. KMS 키 부여를 통해 AWS 서비스가 사용하는 키에 KMS 키 정책에서 `aws:SourceOrgID`을 사용하면 예상치 못한 동작이 발생할 수 있습니다.

- 데이터 유형-문자열
- 값 유형-단일 값

이 조건 키를 사용하여 AWS 서비스가 서비스간 트랜잭션 중에 혼동된 대리자로 사용되는 것을 방지합니다. 이 키는 `Principal`가 AWS 서비스 보안 주체인 리소스 기반 정책에서만 사용합니다. 이 조건 키의 값을 요청에 있는 리소스의 조직 ID로 설정합니다. 예를 들어, Amazon S3 버킷 업데이트가 Amazon SNS 주제 게시물을 트리거하면 Amazon S3 서비스에서 `sns:Publish` API 작업을 호출합니다. `sns:Publish` 작업을 허용하는 주제 정책에서 조건 키의 값을 Amazon S3 버킷의 조직 ID로 설정합니다. 이러한 조건 키가 권장되는 방법 및 시기에 대한 자세한 내용은 사용하시는 AWS 서비스의 설명서를 참조하세요.

aws:UserAgent

이 키를 사용하여 요청자의 클라이언트 애플리케이션을 정책에서 지정한 애플리케이션과 비교합니다.

- 가용성-이 키는 항상 요청 컨텍스트에 포함됩니다.
- 데이터 유형-문자열
- 값 유형-단일 값

> **Warning**
>
> 이 키를 사용할 때는 주의해야 합니다. `aws:UserAgent` 값은 HTTP 헤더의 호출자가 제공하기 때문에, 권한이 없는 사용자가 수정된 브라우저나 사용자 지정 브라우저를 사용하여 원하는 `aws:UserAgent` 값을 제공할 수 있습니다. 따라서 무단 사용자의 직접 AWS 요청을 차단할 목적으로 `aws:UserAgent`을 사용해서는 안 됩니다. 특정 클라이언트 애플리케이션을 허용하는 데 사용할 수 있으며 정책을 테스트한 후에만 사용할 수 있습니다.

기타 교차 서비스 조건 키

AWS STS는 OIDC 페더레이션을 위한 SAML 기반 페더레이션 조건 키와 교차 서비스 조건 키를 지원합니다. 이러한 키는 SAML을 사용하여 연동된 사용자가 다른 서비스에서 AWS 작업을 수행할 때 사용될 수 있습니다.

IAM 및 AWS STS 조건 컨텍스트 키

JSON 정책의 `Condition` 요소를 사용하여 모든 AWS 요청의 요청 컨텍스트에 포함된 키 값을 테스트할 수 있습니다. 이러한 키는 요청 자체 또는 해당 요청이 참조하는 리소스에 대한 정보를 제공합니다. 사용자가 요청한 작업을 허용하기 전에 키에 값이 지정되었는지 확인할 수 있습니다. 이렇게 하면 JSON 정책 문이 수신 요청과 일치 또는 불일치할 경우 보다 세분화된 제어가 가능합니다. JSON 정책의 `Condition` 요소 사용에 대한 자세한 방법은 IAM JSON 정책 요소: Condition 섹션을 참조하세요.

이 주제에서는 IAM 서비스(`iam:` 접두사 포함) 및 AWS Security Token Service(AWS STS) 서비스(`sts:` 접두사 포함)에서 정의 및 제공하는 키에 대해 설명합니다. 다른 여러 AWS 서비스에서도 해당 서비스가 정의한 작업 및 리소스와 관련된 서비스 고유 키를 제공합니다. 자세한 내용은 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요. 대개의 경우 조건 키를 지원하는 서비스의 설명서에 추가 정보를 확인할 수 있습니다. 예를 들어 Amazon S3 리소스 정책에서 사용할 수 있는 키에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon S3 정책 키를 참조하세요.

주제

- IAM에서 사용할 수 있는 키
- AWS OIDC 페더레이션에서 사용 가능한 키

- SAML 기반 AWS STS 연동에 사용할 수 있는 키
- 교차 서비스 SAML 기반 AWS STS 페더레이션 컨텍스트 키
- AWS STS에서 사용할 수 있는 키

IAM에서 사용할 수 있는 키

IAM 리소스에 대한 액세스 제어 정책에서는 다음과 같은 조건 키를 사용할 수 있습니다.

iam:AssociatedResourceArn

ARN 연산자를 사용합니다.

대상 서비스에서 이 역할이 연결될 리소스의 ARN을 지정합니다. 리소스는 일반적으로 보안 주체가 역할을 전달하는 서비스에 속합니다. 경우에 따라 리소스는 세 번째 서비스에 속할 수 있습니다. 예를 들어 Amazon EC2 인스턴스에서 사용하는 역할을 Amazon EC2 Auto Scaling에 전달할 수 있습니다. 이 경우 조건은 Amazon EC2 인스턴스의 ARN과 일치합니다.

이 조건 키는 정책의 PassRole 작업에만 적용됩니다. 다른 작업을 제한하는 데 사용할 수 없습니다.

정책에서 이 조건 키를 사용하여 엔터티가 역할을 전달하도록 허용하지만, 해당 역할이 지정된 리소스와 연결된 경우에만 가능합니다. 와일드카드(*)를 사용하면 리전 또는 리소스 ID를 제한하지 않고 특정 유형의 리소스에서 작업을 수행하도록 허용할 수 있습니다. 예를 들어 IAM 사용자 또는 역할이 `us-east-1` 또는 `us-west-1` 리전의 인스턴스에서 사용할 Amazon EC2 서비스에 모든 역할을 전달하도록 허용할 수 있습니다. IAM 사용자 또는 역할은 다른 서비스에 역할을 전달할 수 없습니다. 또한 Amazon EC2가 다른 리전의 인스턴스에서 역할을 사용하는 것을 허용하지 않습니다.

```
{
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "*",
    "Condition": {
        "StringEquals": {"iam:PassedToService": "ec2.amazonaws.com"},
        "ArnLike": {
            "iam:AssociatedResourceARN": [
                "arn:aws:ec2:us-east-1:111122223333:instance/*",
                "arn:aws:ec2:us-west-1:111122223333:instance/*"
            ]
        }
    }
}
```

> **Note**
>
> iam:PassedToService를 지원하는 AWS 서비스는 이 조건 키도 지원합니다.

iam:AWSServiceName

문자열 연산자를 사용합니다.

이 역할이 연결되는 AWS 서비스를 지정합니다.

이 예에서는 서비스 이름이 access-analyzer.amazonaws.com인 경우 엔터티가 서비스 연결 역할을 생성하도록 허용합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
       "Effect": "Allow",
       "Action": "iam:CreateServiceLinkedRole",
       "Resource": "*",
       "Condition": {
         "StringLike": {
           "iam:AWSServiceName": "access-analyzer.amazonaws.com"
         }
       }
     }]
 }
```

iam:FIDO-certification

문자열 연산자를 사용합니다.

FIDO 보안 키 등록 시 MFA 디바이스 FIDO 인증 등급을 확인합니다. 디바이스 인증은 FIDO Alliance 메타데이터 서비스(MDS)에서 가져옵니다. FIDO 보안 키의 인증 상태 또는 등급이 변경될 경우 디바이스를 등록 취소하고 업데이트된 인증 정보를 가져오기 위해 다시 등록하지 않는 한 디바이스 인증은 업데이트되지 않습니다.

가능한 값: L1, L1plus, L2, L2plus, L3, L3plus

이 예에서는 보안 키를 등록하고 디바이스에 대한 FIDO Level 1 plus 인증을 가져옵니다.

```
{
```

```
      "Version": "2012-10-17",
      "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-certification": "L1plus"
                }
            }
        }
    ]

}
```

iam:FIDO-FIPS-140-2-certification

문자열 연산자를 사용합니다.

FIDO 보안 키 등록 시 MFA 디바이스 FIPS-140-2 검증 인증 등급을 확인합니다. 디바이스 인증은 FIDO Alliance 메타데이터 서비스(MDS)에서 가져옵니다. FIDO 보안 키의 인증 상태 또는 등급이 변경될 경우 디바이스를 등록 취소하고 업데이트된 인증 정보를 가져오기 위해 다시 등록하지 않는 한 디바이스 인증은 업데이트되지 않습니다.

가능한 값: L1, L2, L3, L4

이 예에서는 보안 키를 등록하고 디바이스에 대한 FIPS-140-2 Level 2 인증을 가져옵니다.

```
{
      "Version": "2012-10-17",
      "Statement": [{
            "Effect": "Allow",
```

```
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-2-certification": "L2"
                }
            }
        }
    ]

}
```

iam:FIDO-FIPS-140-3-certification

문자열 연산자를 사용합니다.

FIDO 보안 키 등록 시 MFA 디바이스 FIPS-140-3 검증 인증 등급을 확인합니다. 디바이스 인증은 FIDO Alliance 메타데이터 서비스(MDS)에서 가져옵니다. FIDO 보안 키의 인증 상태 또는 등급이 변경될 경우 디바이스를 등록 취소하고 업데이트된 인증 정보를 가져오기 위해 다시 등록하지 않는 한 디바이스 인증은 업데이트되지 않습니다.

L1, L2, L3, L4의 가능한 값

이 예에서는 보안 키를 등록하고 디바이스에 대한 FIPS-140-3 Level 3 인증을 가져옵니다.

```
{
      "Version": "2012-10-17",
      "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-3-certification": "L3"
                }
            }
        }
    ]

}
```

iam:RegisterSecurityKey

문자열 연산자를 사용합니다.

MFA 디바이스 활성화의 현재 상태를 확인합니다.

가능한 값: `Create` 또는 `Activate`

이 예에서는 보안 키를 등록하고 디바이스에 대한 FIPS-140-3 Level 1 인증을 가져옵니다.

```
{
      "Version": "2012-10-17",
      "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
```

```
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-3-certification": "L1"
                }
            }
        }
    ]

}
```

iam:OrganizationsPolicyId

문자열 연산자를 사용합니다.

지정된 AWS Organizations ID가 포함된 정책이 요청에 사용된 정책과 일치하는지 확인합니다. 이 조건 키를 사용하는 예시 IAM 정책을 보려면 IAM: Organizations 정책에 대해 마지막으로 액세스한 서비스 정보 보기 섹션을 참조하세요.

iam:PassedToService

문자열 연산자를 사용합니다.

역할을 전달할 수 있는 서비스의 서비스 보안 주체를 지정합니다. 이 조건 키는 정책의 PassRole 작업에만 적용됩니다. 다른 작업을 제한하는 데 사용할 수 없습니다.

정책에서 이 조건 키를 사용할 때 서비스 보안 주체를 사용하여 서비스를 지정합니다. 서비스 보안 주체는 정책의 `Principal` 요소에 지정할 수 있는 서비스 이름입니다. `SERVICE_NAME_URL.amazonaws.com`이 일반적인 형식입니다.

`iam:PassedToService`를 사용하여 특정 서비스에만 역할을 전달할 수 있도록 사용자를 제한할 수 있습니다. 예를 들어, 사용자는 Amazon S3 버킷에 로그 데이터를 대신 쓸 수 있도록 CloudWatch를 신뢰하는 서비스 역할을 생성할 수 있습니다. 그런 다음 사용자는 새 서비스 역할에 권한 정책 및 신뢰 정책을 연결해야 합니다. 이 경우, 신뢰 정책은 `cloudwatch.amazonaws.com` 요소에 `Principal`을 지정해야 합니다. 사용자가 CloudWatch에 역할을 전달하도록 허용하는 정책을 보려면 IAM: IAM 역할을 특정 AWS 서비스로 전달 섹션을 참조하세요.

이 조건 키를 사용하면 사용자가 여러분이 지정한 서비스에 대해서만 서비스 역할을 생성하도록 할 수 있습니다. 예를 들어, 앞의 정책을 가진 사용자가 Amazon EC2에 대한 서비스 역할을 생성하려

고 하면 작업이 실패합니다. 해당 사용자에게 Amazon EC2로 역할을 전달할 권한이 없기 때문입니다.

역할을 서비스에 전달한 다음 역할을 다른 서비스에 전달하는 경우가 있습니다. `iam:PassedToService`에는 역할을 전달하는 중간 서비스가 아니라 역할을 수임하는 최종 서비스만 포함됩니다.

> **Note**
>
> 일부 서비스는 이러한 조건 키를 지원하지 않습니다.

iam:PermissionsBoundary

ARN 연산자를 사용합니다.

지정한 정책이 IAM 보안 주체 리소스에 권한 경계로서 연결되어 있는지 확인합니다. 자세한 내용은 IAM 엔터티의 권한 범위 섹션을 참조하세요.

iam:PolicyARN

ARN 연산자를 사용합니다.

관리형 정책이 포함된 요청에서 관리형 정책의 Amazon 리소스 이름(ARN)을 확인합니다. 자세한 내용은 정책에 대한 액세스 제어 단원을 참조하십시오.

iam:ResourceTag/***key-name***

문자열 연산자를 사용합니다.

자격 검증 리소스(사용자 또는 역할)에 연결된 태그가 지정된 키 이름 및 값과 일치하는지 확인합니다.

> **Note**
>
> IAM 및 AWS STS은 `iam:ResourceTag` IAM 조건 키와 `aws:ResourceTag` 전역 조건 키를 모두 지원합니다.

IAM 리소스에 사용자 지정 속성을 키 값 페어의 형태로 추가할 수 있습니다. IAM 리소스의 태그에 대한 자세한 내용은 the section called “IAM 리소스용 태그” 섹션을 참조하세요. `ResourceTag`를

사용하여 AWS 리소스를 비롯한 IAM 리소스에 대한 액세스를 제어할 수 있습니다. 그러나 IAM은 그룹에 대한 태그를 지원하지 않으므로 태그를 사용하여 그룹에 대한 액세스를 제어할 수 없습니다.

이 예제는 **status=terminated** 태그가 있는 사용자만 삭제할 수 있는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 이 정책을 사용하려면 정책 예제의 *#### ## ### ###*를 본인의 정보로 대체합니다. 그런 다음 정책 생성 또는 정책 편집의 지침을 따릅니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "iam:DeleteUser",
        "Resource": "*",
        "Condition": {"StringEquals": {"iam:ResourceTag/status": "terminated"}}
    }]
}
```

AWS OIDC 페더레이션에서 사용 가능한 키

OIDC 페더레이션을 사용하여 OpenID Connect를 준수하는 ID 제공업체(idP)를 통해 AWS 계정의 IAM OIDC(OpenID Connect) ID 공급자에게 인증된 사용자에게 임시 보안 자격 증명을 제공할 수 있습니다. 이러한 공급자의 예로는 GitHub, Amazon Cognito, Login with Amazon, Google 등이 있습니다. Amazon Elastic Kubernetes Service 워크로드에 부여된 서비스 계정 토큰뿐만 아니라 자체 IdP의 자격 증명 토큰 및 액세스 토큰도 사용할 수 있습니다.

AWS OIDC 조건 컨텍스트 키를 사용하여 페더레이션 사용자가 특정 공급자, 앱 또는 사용자와 연결된 리소스에만 액세스할 수 있도록 정책을 작성할 수 있습니다. 이러한 키는 일반적으로 역할에 대한 신뢰 정책에서 사용됩니다. OIDC 공급자(`token.actions.githubusercontent.com`)의 이름 다음에 클레임(`:aud`)을 사용하여 **`token.actions.githubusercontent.com:aud`**와 같이 조건 키를 정의합니다.

일부 OIDC 페더레이션 조건 키는 역할 세션에서 리소스 액세스를 승인하는 데 사용할 수 있습니다. 세션에서 사용 가능 열의 값이 Yes인 경우 정책에서 이러한 조건 키를 사용하여 다른 AWS 서비스에서 사용자가 액세스할 수 있는 대상을 정의할 수 있습니다. 세션에서 클레임을 사용할 수 없는 경우 OIDC 조건 컨텍스트 키는 초기 AssumeRoleWithWebIdentity 인증을 위한 역할 신뢰 정책에서만 사용할 수 있습니다.

IdP를 선택하여 IdP의 클레임이 AWS의 IAM 조건 컨텍스트 키에 어떻게 매핑되는지 확인합니다.

Default

기본값에는 표준 OIDC 클레임과 해당 클레임이 AWS의 AWS STS 조건 컨텍스트 키에 매핑되는 방식이 나열됩니다. 이러한 키를 사용하여 역할에 대한 액세스를 제어할 수 있습니다. 이렇게 하려면 AWS STS 조건 키를 IdP JWT 클레임 열의 값과 비교합니다. IdP가 탭 옵션에 나열되지 않은 경우 이 매핑을 사용합니다.

OIDC JWT ID 토큰에서 기본 구현을 사용하는 IdP의 몇 가지 예로는 GitHub Actions 워크플로와 Google이 있습니다.

AWS STS 조건 키	IdP JWT 클레임	세션에서 사용 가능
amr	amr	예
aud	azp azp의 값이 설정되지 않은 경우 aud 조건 키가 aud 클레임에 매핑됩니다.	예
이메일	이메일	아니요
oaud	aud	아니요
sub	sub	예

GitHub에서 OIDC 조건 컨텍스트 키 사용에 대한 자세한 내용은 GitHub OIDC ID 제공자의 역할 구성 섹션을 참조하세요. Google aud 및 azp 필드에 대한 자세한 내용은 Google ID 플랫폼 OpenID Connect 안내서를 참조하세요.

amr

문자열 연산자를 사용합니다. 이 키는 다수의 값을 갖습니다. 이 말은 정책 내에서 조건 설정 연산자를 사용하여 테스트한다는 것을 의미합니다.

예: `token.actions.githubusercontent.com:amr`

인증 방법 참조에는 사용자에 대한 로그인 정보가 포함됩니다. 키에 추가되는 값은 다음과 같습니다.

- 사용자 인증 전에는 키에 `unauthenticated` 값만 추가됩니다.

- 사용자 인증 후에는 키에 `authenticated` 값과 호출 시 사용된 로그인 공급자 이름(`accounts.google.com`)이 추가됩니다.

aud

문자열 연산자를 사용합니다.

예:

- `accounts.google.com:aud`
- `token.actions.githubusercontent.com:aud`

aud 조건 키를 사용하여 대상이 정책에서 지정한 대상과 일치하는지 확인합니다. 동일한 자격 증명 공급자에 대해 aud 키와 함께 sub 키를 사용할 수 있습니다.

이 조건 키는 다음 토큰 필드에서 설정됩니다.

- azp 필드가 설정되지 않은 경우 애플리케이션의 OAuth 2.0 Google 클라이언트 ID에 대한 aud. azp 필드가 설정되면 aud 필드가 `accounts.google.com:oaud` 조건 키와 일치합니다.
- azp 필드가 설정된 경우 azp. 이러한 경우는 웹 애플리케이션과 Android 앱이 서로 다른 OAuth 2.0 Google 클라이언트 ID를 가지고 있지만 동일한 Google API 프로젝트를 공유하는 하이브리드 앱에서 발생할 수 있습니다.

`accounts.google.com:aud` 조건 키를 사용하여 정책을 작성할 때 앱이 azp 필드를 설정하는 하이브리드 앱인지 여부를 알아야 합니다.

azp 필드가 설정되지 않음

다음 예제 정책은 azp 필드를 설정하지 않는 비 하이브리드 앱에 적용됩니다. 이 경우 Google ID 토큰 aud 필드 값은 `accounts.google.com:aud` 및 `accounts.google.com:oaud` 조건 키 값과 일치합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Federated": "accounts.google.com"},
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
```

```
                    "accounts.google.com:aud": "aud-value",
                    "accounts.google.com:oaud": "aud-value",
                    "accounts.google.com:sub": "sub-value"
                }
            }
        }
    ]
}
```

azp 필드 세트

다음 예제 정책은 azp 필드를 설정하는 하이브리드 앱에 적용됩니다. 이 경우 Google ID 토큰 aud 필드 값은 accounts.google.com:oaud 조건 키 값과 유일하게 일치합니다. azp 필드 값은 accounts.google.com:aud 조건 키 값과 일치합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Federated": "accounts.google.com"},
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "accounts.google.com:aud": "azp-value",
                    "accounts.google.com:oaud": "aud-value",
                    "accounts.google.com:sub": "sub-value"
                }
            }
        }
    ]
}
```

이메일

문자열 연산자를 사용합니다.

예: accounts.google.com:email

이 조건 키는 사용자의 이메일 주소를 확인합니다. 이 클레임의 값은 이 계정에 고유하지 않을 수 있으며 시간이 지남에 따라 변경될 수 있으므로 이 값을 사용자 기록을 확인하는 기본 식별자로 사용해서는 안 됩니다.

oaud

문자열 연산자를 사용합니다.

예: `accounts.google.com:oaud`

이 키는 이 ID 토큰의 용도인 다른 대상(aud)을 지정합니다. 애플리케이션의 OAuth 2.0 클라이언트 ID 중 하나여야 합니다.

sub

문자열 연산자를 사용합니다.

예:

- `accounts.google.com:sub`
- token.actions.githubusercontent.com:sub

이러한 키를 사용하여 해당 주체가 정책에서 지정한 주체와 일치하는지 확인합니다. 동일한 자격 증명 공급자에 대해 sub 키와 함께 aud 키를 사용할 수 있습니다.

다음 역할 신뢰 정책에서 sub 조건 키는 역할을 demo 이름이 지정된 GitHib 브랜치로 제한합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      "Condition": {
         "StringEquals": {
            "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
            "token.actions.githubusercontent.com:sub": "repo:octo-org/octo-
repo:ref:refs/heads/demo"
         }
      }
   ]
}
```

Amazon Cognito

이 탭에서는 Amazon Cognito가 OIDC 클레임을 AWS의 AWS STS 조건 컨텍스트 키에 매핑하는 방법을 설명합니다. 이러한 키를 사용하여 역할에 대한 액세스를 제어할 수 있습니다. 이렇게 하려면 AWS STS 조건 키를 IdP JWT 클레임 열의 값과 비교합니다.

Amazon Cognito에서 사용하는 역할의 경우 `cognito-identity.amazonaws.com`과 클레임을 사용하여 키를 정의합니다.

자격 증명 풀 클레임 매핑에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 기본 공급자 매핑을 참조하세요. 사용자 풀 클레임 매핑에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 ID 토큰 사용을 참조하세요.

AWS STS 조건 키	IdP JWT 클레임	세션에서 사용 가능
amr	amr	예
aud	aud	예
oaud	aud	아니요
sub	sub	예

amr

문자열 연산자를 사용합니다. 이 키는 다수의 값을 갖습니다. 이 말은 정책 내에서 조건 설정 연산자를 사용하여 테스트한다는 것을 의미합니다.

예-`cognito-identity.amazonaws.com:amr`

인증 방법 참조에는 사용자에 대한 로그인 정보가 포함됩니다. 키에 추가되는 값은 다음과 같습니다.

- 사용자 인증 전에는 키에 `unauthenticated` 값만 추가됩니다.
- 사용자 인증 후에는 키에 `authenticated` 값과 호출 시 사용된 로그인 공급자 이름(`cognito-identity.amazonaws.com`)이 추가됩니다.

한 예로, Amazon Cognito 역할의 신뢰 정책에서는 다음 조건에 따라 사용자의 인증 여부를 테스트합니다.

```
"Condition": {
  "StringEquals":
    { "cognito-identity.amazonaws.com:aud": "us-east-2:identity-pool-id" },
  "ForAnyValue:StringLike":
    { "cognito-identity.amazonaws.com:amr": "unauthenticated" }
```

```
}
```

aud

문자열 연산자를 사용합니다.

예-cognito-identity.amazonaws.com:aud

사용자를 인증한 사용자 풀 앱 클라이언트입니다. Amazon Cognito는 액세스 토큰 client_id 클레임에서 동일한 값을 렌더링합니다.

oaud

문자열 연산자를 사용합니다.

예-cognito-identity.amazonaws.com:oaud

사용자를 인증한 사용자 풀 앱 클라이언트입니다. Amazon Cognito는 액세스 토큰 client_id 클레임에서 동일한 값을 렌더링합니다.

sub

문자열 연산자를 사용합니다.

예-cognito-identity.amazonaws.com:sub

인증된 사용자에 대한 고유 식별자(UUID) 또는 제목입니다. 사용자 풀에서 사용자 이름이 고유하지 않을 수 있습니다. 하위 클레임은 특정 사용자를 식별하는 가장 좋은 방법입니다. 동일한 자격 증명 공급자에 대해 sub 키와 함께 aud 키를 사용할 수 있습니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
      "Condition": {
         "StringEquals": {
            "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-abcd-abcd-
abcd-123456790ab",
            "cognito-identity.amazonaws.com:sub": [
               "us-east-1:12345678-1234-1234-1234-123456790ab",
               "us-east-1:98765432-1234-1234-1243-123456790ab"
            ]
         }
      }
   ]
```

```
}
```

Login with Amazon

이 탭에서는 Login with Amazon이 OIDC 클레임을 AWS의 AWS STS 조건 컨텍스트 키에 매핑하는 방법을 설명합니다. 이러한 키를 사용하여 역할에 대한 액세스를 제어할 수 있습니다. 이렇게 하려면 AWS STS 조건 키를 IdP JWT 클레임 열의 값과 비교합니다.

AWS STS 조건 키	IdP JWT 클레임	세션에서 사용 가능
app_id	애플리케이션 ID	예
sub	사용자 ID	예
user_id	사용자 ID	예

app_id

문자열 연산자를 사용합니다.

예-`www.amazon.com:app_id`

이 키는 다른 자격 증명 공급자가 사용하는 `aud` 필드와 일치하는 대상 컨텍스트를 지정합니다.

sub

문자열 연산자를 사용합니다.

예-`www.amazon.com:sub`

이 키를 사용하여 사용자 ID가 정책에서 지정한 ID와 일치하는지 확인합니다. 동일한 자격 증명 공급자에 대해 `sub` 키와 함께 `aud` 키를 사용할 수 있습니다.

user_id

문자열 연산자를 사용합니다.

예-`www.amazon.com:user_id`

이 키는 다른 자격 증명 공급자가 사용하는 `aud` 필드와 일치하는 대상 컨텍스트를 지정합니다. 동일한 자격 증명 공급자에 대해 `user_id` 키와 함께 `id` 키를 사용할 수 있습니다.

Facebook

이 탭에서는 Facebook이 OIDC 클레임을 AWS의 AWS STS 조건 컨텍스트 키에 매핑하는 방법을 설명합니다. 이러한 키를 사용하여 역할에 대한 액세스를 제어할 수 있습니다. 이렇게 하려면 AWS STS 조건 키를 IdP JWT 클레임 열의 값과 비교합니다.

AWS STS 조건 키	IdP JWT 클레임	세션에서 사용 가능
app_id	애플리케이션 ID	예
id	id	예

app_id

문자열 연산자를 사용합니다.

예-`graph.facebook.com:app_id`

이 키는 다른 자격 증명 공급자가 사용하는 aud 필드와 일치하는 대상 컨텍스트를 검증합니다.

id

문자열 연산자를 사용합니다.

예-`graph.facebook.com:id`

이 키를 사용하여 애플리케이션(또는 사이트) ID가 정책에서 지정한 ID와 일치하는지 확인합니다.

OIDC 페더레이션에 대한 자세한 내용

- Amazon Cognito 사용 설명서
- OIDC 페더레이션

SAML 기반 AWS STS 연동에 사용할 수 있는 키

AWS Security Token Service(AWS STS)를 사용하여 SAML 기반 연동으로 작업하는 경우 정책에 조건 키를 추가할 수 있습니다.

SAML 역할 신뢰 정책

역할 신뢰 정책에서는 다음과 같은 키를 추가하여 호출자의 역할 위임 가능 여부를 구성할 수 있습니다. `saml:doc`를 제외한 모든 값은 SAML 어설션에서 가져옵니다. 조건에 따라 정책을 생성하거나 편집할 때 목록의 모든 항목을 IAM 콘솔의 시각적 편집기에서 사용할 수 있습니다. `[]`가 표시된 항목은 지정된 유형의 목록을 값으로 가질 수 있습니다.

saml:aud

문자열 연산자를 사용합니다.

SAML 어설션이 전송되는 엔드포인트 URL입니다. 이 키에 대한 값은 `Audience` 필드가 아닌 어설션의 `SAML Recipient` 필드에서 얻습니다.

saml:commonName[]

문자열 연산자를 사용합니다.

`commonName` 속성입니다.

saml:cn[]

문자열 연산자를 사용합니다.

이 키는 `eduOrg` 속성입니다.

saml:doc

문자열 연산자를 사용합니다.

이 키는 역할 위임 시 사용한 보안 주체를 나타냅니다. 형식은 *`account-ID`*/*`provider-friendly-name`*(예: `123456789012/SAMLProviderName`)을 따릅니다. account-ID 값은 SAML 공급자가 속한 계정을 참조합니다.

saml:edupersonaffiliation[]

문자열 연산자를 사용합니다.

이 키는 `eduPerson` 속성입니다.

saml:edupersonassurance[]

문자열 연산자를 사용합니다.

이 키는 `eduPerson` 속성입니다.

saml:edupersonentitlement[]

문자열 연산자를 사용합니다.

이 키는 eduPerson 속성입니다.

saml:edupersonnickname[]

문자열 연산자를 사용합니다.

이 키는 eduPerson 속성입니다.

saml:edupersonorgdn

문자열 연산자를 사용합니다.

이 키는 eduPerson 속성입니다.

saml:edupersonorgunitdn[]

문자열 연산자를 사용합니다.

이 키는 eduPerson 속성입니다.

saml:edupersonprimaryaffiliation

문자열 연산자를 사용합니다.

이 키는 eduPerson 속성입니다.

saml:edupersonprimaryorgunitdn

문자열 연산자를 사용합니다.

이 키는 eduPerson 속성입니다.

saml:edupersonprincipalname

문자열 연산자를 사용합니다.

이 키는 eduPerson 속성입니다.

saml:edupersonscopedaffiliation[]

문자열 연산자를 사용합니다.

이 키는 eduPerson 속성입니다.

saml:edupersontargetedid[]

문자열 연산자를 사용합니다.

이 키는 eduPerson 속성입니다.

saml:eduorghomepageuri[]

문자열 연산자를 사용합니다.

이 키는 eduOrg 속성입니다.

saml:eduorgidentityauthnpolicyuri[]

문자열 연산자를 사용합니다.

이 키는 eduOrg 속성입니다.

saml:eduorglegalname[]

문자열 연산자를 사용합니다.

이 키는 eduOrg 속성입니다.

saml:eduorgsuperioruri[]

문자열 연산자를 사용합니다.

이 키는 eduOrg 속성입니다.

saml:eduorgwhitepagesuri[]

문자열 연산자를 사용합니다.

이 키는 eduOrg 속성입니다.

saml:givenName[]

문자열 연산자를 사용합니다.

givenName 속성입니다.

saml:iss

문자열 연산자를 사용합니다.

발급자로서 URN으로 표시됩니다.

saml:mail[]

문자열 연산자를 사용합니다.

`mail` 속성입니다.

saml:name[]

문자열 연산자를 사용합니다.

`name` 속성입니다.

saml:namequalifier

문자열 연산자를 사용합니다.

SAML 공급자의 표시 이름을 기준으로 하는 해시 값입니다. 이 값은 다음 값을 순서대로 연결하며 '/' 문자로 구분합니다.

1. `Issuer` 응답 값(`saml:iss`)
2. `AWS` 계정 ID
3. IAM에서 SAML 공급자의 표시 이름(ARN의 마지막 부분)

계정 ID와 SAML 공급자 표시 이름의 연결을 IAM 정책에서 `saml:doc` 키로 사용할 수 있습니다. 자세한 내용은 SAML 기반 페더레이션에서 사용자를 고유하게 식별 단원을 참조하십시오.

saml:organizationStatus[]

문자열 연산자를 사용합니다.

이 키는 `organizationStatus` 속성입니다.

saml:primaryGroupSID[]

문자열 연산자를 사용합니다.

`primaryGroupSID` 속성입니다.

saml:sub

문자열 연산자를 사용합니다.

이것은 클레임의 주체로서 여기에는 조직 내 사용자 개개인을 식별할 수 있는 고유 값이 포함됩니다(예: `_cbb88bf52c2510eabe00c1642d4643f41430fe25e3`).

saml:sub_type

문자열 연산자를 사용합니다.

이 키는 persistent, transient 값을 갖거나 SAML 어설션에서 사용되는 Format 및 Subject 요소의 전체 NameID URI로 구성될 수 있습니다. persistent의 값은 saml:sub의 값이 세션 간 사용자에서도 동일하다는 것을 나타냅니다. 값이 transient인 경우 각 세션마다 사용자의 saml:sub 값이 다릅니다. NameID 요소의 Format 속성에 대한 자세한 내용은 인증 응답에 대한 SAML 어설션 구성 섹션을 참조하세요.

saml:surname[]

문자열 연산자를 사용합니다.

surnameuid 속성입니다.

saml:uid[]

문자열 연산자를 사용합니다.

uid 속성입니다.

saml:x500UniqueIdentifier[]

문자열 연산자를 사용합니다.

이 키는 x500UniqueIdentifier 속성입니다.

eduPerson 및 eduOrg 속성에 관한 일반 정보는 REFEDS Wiki 웹사이트를 참조하세요. eduPerson 속성 목록은 eduPerson Object Class Specification(201602)을 참조하세요.

형식이 목록인 조건 키에는 다수의 값이 추가될 수 있습니다. 목록 값 정책에서 조건을 생성하려면 설정 연산자(ForAllValues, ForAnyValue)를 사용하면 됩니다. 예를 들어, 소속이 "faculty", "staff"("student" 제외)인 사용자를 모두 허용하려면 다음과 같은 조건을 사용할 수 있습니다.

```
"Condition": {
   "ForAllValues:StringLike": {
     "saml:edupersonaffiliation":[ "faculty", "staff"]
   }
}
```

교차 서비스 SAML 기반 AWS STS 페더레이션 컨텍스트 키

일부 SAML 기반 페더레이션 조건 키는 후속 요청에서 다른 서비스 및 AssumeRole 호출의 AWS 작업을 승인하는 데 사용할 수 있습니다. 다음은 페더레이션 보안 주체가 다른 역할을 맡을 때 역할 신뢰 정책에서, 그리고 다른 AWS 서비스의 리소스 정책에서 페더레이션 보안 주체의 리소스 액세스를 승인하

는 데 사용할 수 있는 조건 키입니다. 이러한 키 사용에 대한 자세한 내용은 SAML 2.0 기반 페더레이션에 대하여 단원을 참조하세요.

설명을 보려면 조건 키를 선택합니다.

- saml:namequalifier
- saml:sub
- saml:sub_type

> **Note**
>
> 초기 외부 ID 제공업체(IdP) 인증 응답 이후에는 다른 SAML 기반 페더레이션 조건 키를 사용할 수 없습니다.

AWS STS에서 사용할 수 있는 키

AWS Security Token Service(AWS STS) 작업을 사용하여 수임한 역할에 대한 IAM 역할 신뢰 정책에서는 다음 조건 키를 사용할 수 있습니다.

saml:sub

문자열 연산자를 사용합니다.

이것은 클레임의 주체로서 여기에는 조직 내 사용자 개개인을 식별할 수 있는 고유 값이 포함됩니다(예: `_cbb88bf52c2510eabe00c1642d4643f41430fe25e3`).

sts:AWSServiceName

문자열 연산자를 사용합니다.

이 키를 사용하여 보유자 토큰을 사용할 수 있는 서비스를 지정합니다. 정책에서 이 조건 키를 사용할 때 서비스 보안 주체를 사용하여 서비스를 지정합니다. 서비스 보안 주체는 정책의 `Principal` 요소에 지정할 수 있는 서비스 이름입니다. 예를 들어 `codeartifact.amazonaws.com`은 AWS CodeArtifact 서비스 보안 주체입니다.

가용성 - 이 키는 보유자 토큰을 가져오는 요청에 존재합니다. 보유자 토큰을 얻기 위해 AWS STS를 직접 호출할 수 없습니다. 다른 서비스에서 일부 작업을 수행하면 서비스가 사용자를 대신하여 보유자 토큰을 요청합니다.

일부 AWS 서비스의 경우 프로그래밍 방식으로 리소스에 액세스하기 전에 AWS STS 서비스 보유자 토큰을 가져올 수 있는 권한이 있어야 합니다. 예를 들어 AWS CodeArtifact에서는 보안 주체가 일부 작업을 수행하기 위해 보유자 토큰을 사용해야 합니다. 이 `aws codeartifact get-authorization-token` 명령은 보유자 토큰을 반환합니다. 그런 다음 보유자 토큰을 사용하여 AWS CodeArtifact 작업을 수행 할 수 있습니다. 보유자 토큰에 대한 자세한 내용은 보유자 토큰 사용 섹션을 참조하세요.

이 조건 키를 사용하여 보안 주체가 특정 서비스에 사용할 보유자 토큰을 가져오도록 허용할 수 있습니다.

sts:DurationSeconds

숫자 연산자를 사용합니다.

이 키를 사용하여 보안 주체가 AWS STS 보유자 토큰을 가져올 때 사용할 수 있는 지속 시간(초)을 지정합니다.

가용성 - 이 키는 보유자 토큰을 가져오는 요청에 존재합니다. 보유자 토큰을 얻기 위해 AWS STS를 직접 호출할 수 없습니다. 다른 서비스에서 일부 작업을 수행하면 서비스가 사용자를 대신하여 보유자 토큰을 요청합니다. 이 키는 AWS STS assume-role 작업에 대해 존재하지 않습니다.

일부 AWS 서비스의 경우 프로그래밍 방식으로 리소스에 액세스하기 전에 AWS STS 서비스 보유자 토큰을 가져올 수 있는 권한이 있어야 합니다. 예를 들어 AWS CodeArtifact에서는 보안 주체가 일부 작업을 수행하기 위해 보유자 토큰을 사용해야 합니다. 이 `aws codeartifact get-authorization-token` 명령은 보유자 토큰을 반환합니다. 그런 다음 보유자 토큰을 사용하여 AWS CodeArtifact 작업을 수행 할 수 있습니다. 보유자 토큰에 대한 자세한 내용은 보유자 토큰 사용 섹션을 참조하세요.

sts:ExternalId

문자열 연산자를 사용합니다.

IAM 역할을 수임할 때 보안 주체가 특정 식별자를 제공하도록 요구하려면 이 키를 사용합니다.

가용성 - 이 키는 보안 주체가 AWS CLI 또는 AWS API를 사용하여 역할을 수임하는 동안 외부 ID를 제공할 때 요청에 존재합니다.

다른 계정에서 역할을 맡을 때 필요할 수도 있는 고유한 식별자. 역할이 속한 계정의 관리자가 외부 ID를 제공한 경우에는 해당 값을 `ExternalId` 파라미터에 제공하세요. 이 값은 암호 또는 계정 번호와 같은 어떤 문자열도 가능합니다. 외부 ID의 주된 기능은 혼동된 대리자 문제를 해결하고 방지하는 것입니다. 외부 ID와 혼동된 대리자 문제에 대해 자세히 알아보려면 타사가 소유한 AWS 계정에 액세스 섹션을 참조하세요.

`ExternalId` 값은 최소 2자, 최대 1,224자여야 합니다. 이 값은 공백 없이 영숫자여야 합니다. 이 값은 더하기(+), 등호(=), 쉼표(,) 마침표(.), 기호(@), 콜론(:), 슬래시(/) 및 하이픈(-)과 같은 기호도 포함할 수 있습니다.

sts:RequestContext/context-key

문자열 연산자를 사용합니다.

이 키를 사용하여 요청에 전달된 신뢰할 수 있는 토큰 발급자 서명 컨텍스트 어설션에 포함된 세션 컨텍스트 키-값 페어를 역할 신뢰 정책에 지정된 컨텍스트 키-값과 비교합니다.

가용성 - 이 키는 AWS STS AssumeRole API 작업을 사용해 역할을 수임하는 동안 `ProvidedContexts` 요청 파라미터에 컨텍스트 어설션이 제공되면 요청에 표시됩니다.

이 컨텍스트 키의 형식은 `"sts:RequestContext/context-key":"context-value"`와 같으며 여기서 `context-key` 및 `context-value`는 컨텍스트 키-값 페어입니다. 요청에 전달된 서명된 컨텍스트 어설션에 여러 컨텍스트 키가 포함된 경우 각 키-값 페어에 대해 하나의 컨텍스트 키가 있습니다. 보안 주체가 결과 세션 토큰 내에서 컨텍스트 키를 설정할 수 있도록 하려면 역할 신뢰 정책에서 `sts:SetContext` 작업에 대한 권한을 부여해야 합니다. 이 키와 함께 사용할 수 있는 지원되는 IAM Identity Center 컨텍스트 키에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 IAM Identity Center의 AWS STS 조건 키를 참조하세요.

역할 신뢰 정책에서 이 키를 사용하여 역할을 수임할 때 사용자 또는 사용자 특성에 따라 세분화된 액세스 제어를 적용할 수 있습니다. 역할이 위임되면 역할 수임 요청에서 컨텍스트 제공자가 설정한 세션 컨텍스트 키-값 페어를 포함하는 `AdditionalEventData` 속성 내 AWS CloudTrail 로그에 활동이 나타납니다. 이렇게 하면 여러 보안 주체가 한 역할을 사용할 때 관리자가 역할 세션 간을 구분하는 것이 쉬워집니다. 키-값 페어는 AWS CloudTrail 또는 AWS STS가 아닌 지정된 컨텍스트 제공자가 설정합니다. 이를 통해 컨텍스트 제공자는 CloudTrail 로그 및 세션 정보에 포함되는 컨텍스트를 제어할 수 있습니다.

sts:RequestContextProviders

ARN 연산자를 사용합니다.

이 키를 사용하여 요청의 컨텍스트 제공자 ARN을 역할 신뢰 정책에 지정된 컨텍스트 제공자 ARN과 비교할 수 있습니다.

가용성 - 이 키는 AWS STS AssumeRole API 작업을 사용해 역할을 수임하는 동안 `ProvidedContexts` 요청 파라미터에 컨텍스트 어설션이 제공되면 요청에 표시됩니다.

다음 예제 조건은 요청에 전달된 컨텍스트 제공자 ARN이 역할 신뢰 정책 조건에 지정된 ARN과 일치하는지 확인합니다.

```
      "Condition": {
        "ForAllValues:ArnEquals": {
          "sts:RequestContextProviders": [
            "arn:aws:iam::aws:contextProvider/IdentityCenter"
          ]
        }
      }
```

sts:RoleSessionName

문자열 연산자를 사용합니다.

보안 주체가 역할을 수임할 때 지정하는 세션 이름을 정책에 지정된 값과 비교하려면 이 키를 사용합니다.

가용성 - 이 키는 보안 주체가 AWS Management Console, assume-role CLI 명령 또는 AWS STS `AssumeRole` API 작업을 사용하여 역할을 수임할 때 요청에 존재합니다.

역할 신뢰 정책에서 이 키를 사용하여 사용자가 역할을 수임할 때 특정 세션 이름을 지정하도록 요구할 수 있습니다. 예를 들어 IAM 사용자가 자신의 사용자 이름을 세션 이름으로 지정하도록 요구할 수 있습니다. IAM 사용자가 역할을 수임하면 사용자 이름과 일치하는 세션 이름과 함께 AWS CloudTrail 로그에 활동이 나타납니다. 이렇게 하면 여러 보안 주체가 한 역할을 사용할 때 관리자가 역할 세션 간을 구분하는 것이 쉬워집니다.

다음 역할 신뢰 정책에서는 111122223333 계정의 IAM 사용자가 역할을 수임할 때 IAM 사용자 이름을 세션 이름으로 지정하도록 요구합니다. 이 요구 사항은 조건 키의 `aws:username` 조건 변수를 사용하여 적용됩니다. 이 정책을 통해 IAM 사용자는 정책에 연결된 역할을 수임할 수 있습니다. 이 정책은 임시 자격 증명을 사용하는 사용자가 역할을 수임하는 것을 허용하지 않습니다. `username` 변수는 IAM 사용자에만 사용되기 때문입니다.

> **Important**
>
> 단일 값 조건 키를 변수로 사용할 수 있습니다. 다중 값 조건 키는 변수로 사용할 수 없습니다.

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

```
        {
            "Sid": "RoleTrustPolicyRequireUsernameForSessionName",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
            "Condition": {
                "StringLike": {"sts:RoleSessionName": "${aws:username}"}
            }
        }
    ]
}
```

관리자가 작업에 대한 AWS CloudTrail 로그를 볼 때 세션 이름을 계정의 사용자 이름과 비교할 수 있습니다. 다음 예제에서는 `matjac`이라는 사용자가 `MateoRole`이라는 역할을 사용하여 작업을 수행했습니다. 관리자는 사용자 이름이 `matjac`인 Mateo Jackson에게 연락할 수 있습니다.

```
    "assumedRoleUser": {
        "assumedRoleId": "AROACQRSTUVWRAOEXAMPLE:matjac",
        "arn": "arn:aws:sts::111122223333:assumed-role/MateoRole/matjac"
    }
```

역할을 사용한 크로스 계정 액세스를 허용하는 경우 한 계정의 사용자가 다른 계정의 역할을 수임할 수 있습니다. CloudTrail에 나열되어 있는 수임된 역할 사용자의 ARN에는 해당 역할이 있는 계정이 포함됩니다. 역할을 수임한 사용자의 계정은 포함되지 않습니다. 사용자는 계정 내에서만 고유합니다. 따라서 관리하는 계정의 사용자가 수임하는 역할에 대해서만 CloudTrail 로그를 확인하는 데 이 방법을 사용하는 것이 좋습니다. 사용자는 여러 계정에서 동일한 사용자 이름을 사용할 수 있습니다.

sts:SourceIdentity

문자열 연산자를 사용합니다.

보안 주체가 역할을 수임할 때 지정하는 소스 자격 증명을 정책에 지정된 값과 비교하려면 이 키를 사용합니다.

가용성 - AWS STS assume-role CLI 명령 또는 AWS STS `AssumeRole`API 작업을 사용해 역할을 수임하는 동안 보안 주체가 소스 자격 증명을 제공할 때 이 키가 요청에 표시됩니다.

역할 신뢰 정책에서 이 키를 사용하여 사용자가 역할을 수임할 때 특정 소스 자격 증명을 설정하도록 요구할 수 있습니다. 예를 들어 인력이나 페더레이션 자격 증명에게 소스 자격 증명에 대한 값을 지정하도록 요구할 수 있습니다. 자격 증명 공급자(IdP)는 사용자 이름 또는 전자 메일과 같이 사용

자와 연결된 특성 중 하나를 소스 자격 증명으로 사용하도록 구성할 수 있습니다. 그런 다음 IdP가 소스 자격 증명을 어설션 내의 속성으로 전달하거나 클레임하여 AWS로 전송합니다. 소스 자격 증명 속성의 값은 역할을 수임하는 사용자 또는 애플리케이션을 식별합니다.

사용자가 역할을 수임하면 활동이 설정된 소스 자격 증명 값과 함께 AWS CloudTrail 로그에 나타납니다. 이를 통해 관리자는 AWS의 역할로 누가 어떤 작업을 수행했는지 쉽게 확인할 수 있습니다. 자격 증명이 소스 자격 증명을 설정하도록 허용하는 `sts:SetSourceIdentity` 작업에 대한 사용 권한을 부여 해야 합니다.

`sts:RoleSessionName`과 달리 소스 자격 증명을 설정한 후에는 값을 변경할 수 없습니다. 이것은 소스 자격 증명에 의해 역할에서 수행되는 모든 작업의 요청 컨텍스트에 있습니다. 세션 자격 증명을 사용하여 다른 역할을 수임하는 경우 값이 후속 역할 세션에 유지됩니다. 한 역할에서 다른 역할을 맡는 것을 역할 체인이라고 합니다.

`aws:SourceIdentity` 전역 조건 키를 사용하여 후속 요청에서 소스 자격 증명 값을 기준으로 AWS 리소스에 대한 액세스 권한을 추가로 제어할 수 있습니다.

다음 역할 신뢰 정책은 IAM 사용자 `AdminUser`가 계정 `111122223333`의 역할을 수임하도록 허용합니다. 또한 소스 자격 증명 집합이 `DiegoRamirez`로 설정된 경우에 한해 `AdminUser`가 소스 자격 증명을 설정하는 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAdminUserAssumeRole",
            "Effect": "Allow",
            "Principal": {"AWS": " arn:aws:iam::111122223333:user/AdminUser"},
            "Action": [
                "sts:AssumeRole",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "StringEquals": {"sts:SourceIdentity": "DiegoRamirez"}
            }
        }
    ]
}
```

소스 자격 증명 정보에 대한 자세한 내용은 위임된 역할로 수행한 작업 모니터링 및 제어 섹션을 참조하세요.

sts:TransitiveTagKeys

문자열 연산자를 사용합니다.

이 키를 사용하여 요청의 전이적 세션 태그 키와 정책에 지정된 전이적 세션 태그 키를 비교합니다.

가용성 - 이 키는 임시 보안 자격 증명을 사용하여 요청을 수행할 때 요청에 존재합니다. 여기에는 모든 assume-role 작업 또는 `GetFederationToken` 작업을 사용해 생성된 자격 증명이 포함됩니다.

임시 보안 자격 증명을 사용하여 요청하면 요청 컨텍스트에 `aws:PrincipalTag` 컨텍스트 키가 포함됩니다. 이 키에는 세션 태그, 전이적 세션 태그 및 역할 태그 목록이 포함됩니다. 전이적 세션 태그를 사용하면 세션 자격 증명을 사용하여 다른 역할을 맡은 경우 모든 후속 세션에서 유지됩니다. 한 역할에서 다른 역할을 맡는 것을 역할 체인이라고 합니다.

정책에서 이 조건 키를 사용하여 역할을 맡거나 사용자를 연동할 때 특정 세션 태그를 전이적으로 설정하도록 요구할 수 있습니다.

AWS 서비스에 사용되는 작업, 리소스 및 조건 키

각 AWS 서비스는 IAM 정책에서 사용할 수 있는 작업, 리소스 및 조건 컨텍스트 키를 정의할 수 있습니다. AWS 서비스와 해당 작업, 리소스 및 조건 컨텍스트 키의 전체 목록은 서비스 승인 참조의 작업, 리소스 및 조건 키를 참조하세요.

IAM에 대해 자세히 알아보는 리소스

IAM은 풍부한 기능을 갖춘 제품이며, IAM으로 AWS 계정 및 리소스를 보호하는 방법이 자세히 설명된 리소스가 많이 있습니다.

주제

- ID
- 자격 증명(암호, 액세스 키 및 MFA 디바이스)
- 권한 및 정책
- 연동 및 위임
- IAM 및 기타 AWS 제품
- 일반 보안 사례
- 일반 리소스

ID

ID를 생성하고, 관리하고, 사용하는 방법은 다음 리소스를 참조하세요.

- IAM Identity Center에서 ID 관리 - IAM Identity Center에서 사용자 및 그룹 생성에 대한 절차 정보입니다.
- IAM 자격 증명(사용자, 사용자 그룹 및 역할) - 사용자, 그룹 및 역할에 대한 심층적인 토론입니다.

자격 증명(암호, 액세스 키 및 MFA 디바이스)

AWS 계정 및 IAM 사용자의 암호, 액세스 키 및 MFA 디바이스를 관리하는 방법은 다음 가이드를 검토하세요.

- AWS에서 사용자 암호 관리 - 계정의 IAM 사용자에 대한 암호 관리 옵션을 설명합니다.
- IAM 사용자의 액세스 키 관리 - 액세스 키의 작동 방식과 액세스 키를 사용하여 프로그래밍 방식으로 AWS를 호출하는 방법을 설명합니다. 액세스 키에 대한 다른 더 안전한 대안이 있습니다. 먼저 이를 고려해 보는 것이 좋습니다. 자세한 내용은 AWS 일반 참조 안내서의 장기 액세스 키에 대한 고려 사항 및 대안을 참조하세요.
- IAM의 AWS 다중 인증 - 디바이스에서 암호와 일회용 코드를 둘 다 입력해야 로그인이 허용되도록 계정 및 IAM 사용자를 구성하는 방법을 설명합니다. (이를 이중 인증이라고도 부릅니다.)

Amazon Web Services에 액세스하는 데 사용하는 보안 인증 유형에 대한 일반적인 내용은 AWS 일반 참조 안내서의 AWS 보안 자격 증명을 참조하세요.

권한 및 정책

IAM 정책 내부의 작동 방식과 함께 가장 효과적으로 권한을 부여하는 방법도 알아보세요.

- IAM의 정책 및 권한 - 권한을 정의하는 데 사용되는 정책 언어를 소개합니다. 사용자나 그룹에 또는 일부 AWS 제품의 경우 리소스 자체에 권한을 연결하는 방법을 설명합니다.
- IAM JSON 정책 요소 참조 - 각 정책 언어 요소에 대한 설명과 예제를 소개합니다.
- IAM 정책 검증 - JSON 정책 검증을 위한 리소스를 찾습니다.
- IAM 자격 증명 기반 정책의 예 - 다양한 AWS 제품의 일반적인 작업에 대한 몇 가지 정책 예를 보여 줍니다.
- AWS 정책 생성기 - 목록에서 제품과 작업을 선택하여 사용자 지정 정책을 생성합니다.
- IAM 정책 시뮬레이터 - 정책에서 AWS에 대한 특정 요청을 허용할지 아니면 거부할지 여부를 테스트합니다.

연동 및 위임

다른 곳에서 인증된(로그인한) 사용자에게 AWS 계정의 리소스에 대한 액세스 권한을 부여할 수 있습니다. 여기에는 다른 AWS 계정의 IAM 사용자(위임), 해당 조직의 로그인 프로세스를 통해 인증된 사용자, Login with Amazon, Facebook, Google 또는 기타 OpenID Connect(OIDC) 호환 자격 증명 공급자 등 인터넷 자격 증명 공급자의 사용자가 포함될 수 있습니다. 이 경우 사용자는 AWS 리소스에 액세스할 수 있는 임시 보안 자격 증명을 받게 됩니다.

- 튜토리얼: IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임 - 다른 AWS 계정의 IAM 사용자에게 크로스 계정 액세스 권한을 부여하는 방법을 설명합니다.
- 임시 자격 증명과 관련된 일반적인 시나리오 - AWS 외부에서 인증된 사용자를 AWS로 연동하는 방법을 설명합니다.

IAM 및 기타 AWS 제품

대부분의 AWS 제품은 IAM과 통합되므로 IAM 기능을 사용하여 그러한 제품의 리소스에 대한 액세스를 보호할 수 있습니다. 다음 리소스에서는 IAM 및 가장 인기 있는 일부 AWS 제품의 보안을 다룹니다.

IAM을 사용하는 전체 제품 목록과 각각의 추가 정보 링크는 AWS IAM으로 작업하는 서비스 섹션을 참조하세요.

Amazon EC2에서 IAM 사용

- Amazon EC2 리소스에 대한 액세스 제어 - 사용자가 Amazon EC2 인스턴스, 볼륨 등을 관리할 수 있도록 IAM 기능으로 허용하는 방법을 설명합니다.
- 인스턴스 프로파일 사용 - IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되면서 다른 AWS 제품에 액세스해야 하는 애플리케이션에 안전하게 자격 증명을 제공하는 방법을 설명합니다.

Amazon S3에서 IAM 사용

- Amazon S3 리소스에 대한 액세스 권한 관리 - IAM 정책을 포함하여 버킷 및 객체에 대한 Amazon S3 보안 모델을 다룹니다.
- IAM 정책 작성: Amazon S3 버킷의 사용자별 폴더에 대한 액세스 권한 부여 - 사용자가 Amazon S3의 자체 폴더를 직접 보호하는 방법을 다룹니다. (Amazon S3 및 IAM에 대한 게시물을 더 보려면 블로그 게시물 제목 아래에서 S3 태그를 선택하세요.)

Amazon RDS와 함께 IAM 사용

- AWS Identity and Access Management(IAM)를 사용하여 Amazon RDS 리소스에 대한 액세스 관리 - IAM을 사용하여 데이터베이스 인스턴스, 데이터베이스 스냅샷 등에 대한 액세스 권한을 제어하는 방법을 설명합니다.
- RDS 리소스 수준 권한에 대한 소개 - IAM을 사용하여 특정 Amazon RDS 인스턴스에 대한 액세스를 제어하는 방법을 다룹니다.

Amazon DynamoDB에서 IAM 사용

- IAM을 사용하여 DynamoDB 리소스에 대한 액세스 제어 - 사용자가 DynamoDB 테이블과 인덱스를 관리할 수 있도록 IAM으로 허용하는 방법을 설명합니다.
- 다음 동영상(8:55)에서는 개별 DynamoDB 데이터베이스 항목이나 속성(또는 둘 다)에 대한 액세스 제어를 제공하는 방법을 설명합니다.

  DynamoDB에 대한 세분화된 액세스 제어 시작하기

일반 보안 사례

AWS 계정과 리소스를 보호하는 가장 좋은 방법에 대한 전문적인 팁과 지침을 찾아보세요.

- 보안, 자격 증명 및 규정 준수 모범 사례 - 보안 아키텍처, IAM 사용, 암호화 및 데이터 보안에 대한 권장 사항을 포함하여 AWS 계정 및 제품 전반에서 보안을 관리하는 방법에 대한 리소스를 찾습니다.
- 자격 증명 및 액세스 관리 - AWS Well-Architected 프레임워크는 클라우드에서 워크로드를 설계하고 실행하기 위한 핵심 개념, 설계 원칙 및 아키텍처 모범 사례를 이해하는 데 도움이 됩니다.
- IAM의 보안 모범 사례 - IAM을 사용하여 AWS 계정과 리소스를 보호하는 방법에 대한 권장 사항을 제시합니다.
- AWS CloudTrail 사용 설명서 AWS CloudTrail을 사용하여 AWS에 대한 API 호출 기록을 추적하고 로그 파일에 해당 정보를 저장합니다. 이를 통해 계정의 리소스에 액세스한 사용자와 계정, 호출이 발생한 시기, 요청된 작업 등을 확인할 수 있습니다.

일반 리소스

다음 리소스에서 IAM 및 AWS에 대해 자세히 알아보세요.

- IAM에 대한 제품 정보 - AWS Identity and Access Management 제품에 대한 일반 정보입니다.
- AWS Identity and Access Management에 대한 AWS re:Post –IAM과 관련된 기술 문제를 AWS 커뮤니티와 의논하려면 AWS re:Post를 방문하세요.

- Classes & Workshops(교육 및 워크숍) – 역할 기반의 과정 및 전문 과정은 물론 자습형 실습에 대한 링크를 통해 AWS 기술을 연마하고 실무에 도움이 되는 경험을 쌓을 수 있습니다.
- AWS 개발자 센터 – 튜토리얼을 살펴보고, 도구를 다운로드하고, AWS 개발자 이벤트에 대해 알아봅니다.
- AWSDeveloper Tools(개발자 도구) – AWS 애플리케이션을 개발 및 관리하기 위한 개발자 도구, SDK, IDE 도구 키트 및 명령줄 도구 링크입니다.
- 시작하기 리소스 센터 - AWS 계정을(를) 설정하고 AWS 커뮤니티에 가입하고 첫 번째 애플리케이션을 시작하는 방법을 알아봅니다.
- 실습 튜토리얼 – 단계별 튜토리얼에 따라 AWS에서 첫 번째 애플리케이션을 시작합니다.
- AWSWhitepapers(백서) – AWS 솔루션 아키텍트 또는 기타 기술 전문가가 아키텍처, 보안 및 경제 등의 토픽에 대해 작성한 포괄적 AWS 기술 백서 목록의 링크입니다.

- AWS SupportCenter(센터) – AWS Support 사례를 생성하고 관리할 수 있는 허브입니다. 또한 포럼, 기술 FAQ, 서비스 상태 및 AWS Trusted Advisor 등의 기타 유용한 자료에 대한 링크가 있습니다.
- AWS Support – 클라우드에서 1대 1로 애플리케이션을 구축 및 실행하도록 지원하는 빠른 응답 지원 채널인 AWS Support에 대한 정보가 포함된 기본 웹 페이지입니다.
- Contact Us(문의처) - AWS 결제, 계정, 이벤트, 침해 및 기타 문제에 대해 문의할 수 있는 중앙 연락 창구입니다.
- AWSSite Terms(사이트 약관) – 저작권 및 상표, 사용자 계정, 라이선스 및 사이트 액세스와 기타 토픽에 대한 세부 정보입니다.

HTTP 쿼리 요청을 사용하여 IAM API 호출

목차

쿼리 API를 사용하여 프로그래밍 방식으로 IAM 및 AWS STS 서비스에 액세스할 수 있습니다. 쿼리 API 요청은 수행할 작업을 나타내기 위해 `Action` 파라미터를 포함해야 하는 HTTPS 요청입니다. IAM 및 AWS STS에서는 모든 작업에 대해 GET 및 POST 요청을 지원합니다. 즉, API 사용 시 어떤 작업에는 GET을 사용하고 또 어떤 작업에는 POST를 사용할 필요가 없습니다. 하지만 GET 요청에는 URL 크기 제한이 적용됩니다. 이 제한은 브라우저에 따라 다르지만 일반적으로 2,048바이트입니다. 따라서 더 큰 크기가 필요한 쿼리 API 요청의 경우 POST 요청을 사용해야 합니다.

응답은 XML 문서입니다. 응답에 대한 자세한 정보는 IAM API 참조의 개별 작업 페이지 또는 AWS Security Token Service API 참조를 참조하세요.

> **Tip**
>
> IAM 또는 AWS STS API 작업을 직접 호출하는 대신 AWS SDK 중 하나를 사용할 수 있습니다. AWS SDK는 다양한 프로그래밍 언어 및 플랫폼(Java, Ruby, .NET, iOS, Android 등)을 위한 라이브러리와 샘플 코드로 구성되어 있습니다. SDK를 사용하면 편리하게 IAM 및 AWS에 프로그래밍 방식으로 액세스할 수 있습니다. 예를 들어 SDK는 요청에 암호화 방식으로 서명(아래 참조), 오류 관리 및 자동으로 요청 재시도와 같은 작업을 처리합니다. 다운로드 및 설치 방법을 비롯하여 AWS SDK에 대한 자세한 내용은 Amazon Web Services용 도구 페이지를 참조하세요.

API 작업 및 오류에 대한 자세한 내용은 IAM API 참조 또는 AWS Security Token Service API 참조를 참조하세요.

엔드포인트

IAM 및 AWS STS에는 전역적 엔드포인트가 하나씩 있습니다.

- (IAM)https://iam.amazonaws.com

- (AWS STS)https://sts.amazonaws.com

> **Note**
>
> AWS STS에서는 전역 엔드포인트 외에 리전 엔드포인트로 요청을 보내는 작업도 지원됩니다. 한 리전에서 AWS STS를 사용하려면 먼저 해당 리전에서 본인의 AWS 계정에 대해 STS를 활성화해야 합니다. AWS STS에 대해 추가 리전을 활성화하는 방법은 AWS STS에서 AWS 리전 관리 단원을 참조하십시오.

모든 서비스용 AWS 엔드포인트 및 리전에 대한 자세한 내용은 AWS 일반 참조의 서비스 엔드포인트 및 할당량을 참조하세요.

HTTPS 필요

쿼리 API는 보안 자격 증명과 같이 민감한 정보를 반환하므로 모든 API 요청에 HTTPS를 사용해야 합니다.

IAM API 요청에 서명

액세스 키 ID와 보안 액세스 키를 사용하여 요청에 서명해야 합니다. IAM에서의 일상적인 작업에는 AWS 계정 루트 사용자 자격 증명을 사용하지 않는 것이 좋습니다. IAM 사용자용 자격 증명을 사용하거나 AWS STS를 사용하여 임시 보안 자격 증명을 생성할 수 있습니다.

API 요청에 서명하려면 AWS 서명 버전 4를 사용하는 것이 좋습니다. 서명 버전 4 사용에 대한 자세한 내용은 AWS 일반 참조의 서명 버전 4 서명 프로세스를 참조하세요.

서명 버전 2를 사용해야 할 경우 서명 버전 2 사용에 대한 자세한 내용은 AWS 일반 참조를 참조하세요.

자세한 내용은 다음 자료를 참조하세요.

- AWS 보안 자격 증명 AWS 액세스에 사용되는 자격 증명 유형에 대한 일반적인 정보를 제공합니다.
- IAM의 보안 모범 사례. IAM 서비스를 사용하여 AWS 리소스를 보호하기 위한 제안 사항의 목록을 제공합니다.
- IAM의 임시 보안 자격 증명. 임시 보안 자격 증명을 만들고 사용하는 방법에 관해 설명합니다.

IAM 관련 문서 기록

다음 표에서는 본 IAM 관련 주요 설명서 업데이트를 설명합니다.

변경 사항	설명	날짜
AccessAnalyzerServiceRolePolicy – 권한 추가	IAM Access Analyzer가 AccessAnalyzerServiceRolePolicy의 서비스 수준 권한에 IAM 사용자 및 역할 정책에 대한 정보를 검색할 수 있는 권한에 대한 지원을 추가했습니다.	2024년 5월 30일
AccessAnalyzerServiceRolePolicy – 권한 추가	IAM Access Analyzer가 AccessAnalyzerServiceRolePolicy의 서비스 수준 권한에 Amazon EC2 스냅샷에 대한 퍼블릭 액세스 차단의 현재 상태를 검색할 수 있는 권한에 대한 지원을 추가했습니다.	2024년 1월 23일
AccessAnalyzerServiceRolePolicy – 권한 추가	IAM Access Analyzer가 AccessAnalyzerServiceRolePolicy의 서비스 수준 권한에 DynamoDB 스트림과 테이블을 추가했습니다.	2024년 1월 11일
AccessAnalyzerServiceRolePolicy – 권한 추가	IAM Access Analyzer가 AccessAnalyzerServiceRolePolicy의 서비스 수준 권한에 Amazon S3 디렉터리 버킷을 추가했습니다.	2023년 12월 1일
IAMAccessAnalyzerReadOnlyAccess – 권한 추가	IAM Access Analyzer는 정책 업데이트가 추가 액세스를 허용하는지 여부를 확인할 수 있도록 IAMAccessAnalyzerR	2023년 11월 26일

	eadOnlyAccess에 대한 권한을 추가했습니다. 이 권한은 IAM Access Analyzer에서 정책에 대한 정책 확인을 수행하는 데 필요합니다.	
IAM Access Analyzer에 미사용 액세스 분석기 추가	IAM Access Analyzer는 미사용 액세스 검사를 단순화하여 최소 권한으로 전환하도록 안내합니다. IAM Access Analyzer는 지속적으로 계정을 분석하여 미사용 액세스를 식별하고 조사 결과가 포함된 중앙 집중식 대시보드를 생성합니다.	2023년 11월 26일
IAM Access Analyzer에 사용자 지정 정책 확인 추가	이제 IAM Access Analyzer는 사용자 지정 정책 확인을 제공하여 배포에 앞서 IAM 정책이 보안 표준을 준수하는지 검증합니다.	2023년 11월 26일
AccessAnalyzerServiceRolePolicy – 권한 추가	IAM Access Analyzer는 AccessAnalyzerServiceRolePolicy의 서비스 수준 권한에 IAM 작업을 추가하여 다음 작업을 지원합니다. • 정책 엔터티 나열 • 마지막으로 액세스한 서비스 세부 정보 생성 • 액세스 키 정보 나열	2023년 11월 26일

마지막으로 액세스한 작업 정보 및 60개 이상의 추가 서비스 및 작업에 대한 정책 생성 지원	이제 IAM에서 마지막으로 액세스한 작업 정보를 지원하고 60개 이상의 추가 서비스에 대해 마지막으로 액세스한 정보가 제공된 작업 목록과 함께 작업 수준 정보를 포함한 정책을 생성합니다.	2023년 11월 1일
140개 이상 서비스에 대해 마지막으로 액세스한 작업 정보 지원	이제 IAM에서 140개 이상의 서비스에 대해 마지막으로 액세스한 작업 정보와 함께 마지막으로 액세스한 작업 정보가 제공되는 작업 문서를 제공합니다.	2023년 9월 14일
루트 사용자 및 IAM 사용자를 위한 여러 다중 인증(MFA) 디바이스 지원	이제 FIDO 보안 키, 가상 인증 애플리케이션을 사용하는 소프트웨어 시간 기반 일회용 암호(TOTP) 또는 하드웨어 TOTP 토큰을 포함하여 사용자당 최대 8개의 MFA 디바이스를 추가할 수 있습니다.	2022년 11월 16일
새로운 리소스 유형에 대한 IAM Access Analyzer 지원	IAM Access Analyzer는 다음과 같은 리소스 유형에 대한 지원을 추가했습니다. • Amazon EBS 볼륨 스냅샷 • Amazon ECR 리포지토리 • Amazon EFS 파일 시스템 • Amazon RDS DB 스냅샷 • Amazon RDS DB 클러스터 스냅샷 • Amazon SNS 주제	2022년 10월 25일

U2F 사용 중단 및 WebAuthn/FIDO 업데이트	MFA 옵션으로 U2F에 대한 설명이 제거되고 WebAuthn, FIDO2 및 FIDO 보안 키에 대한 정보가 추가되었습니다.	2022년 5월 31일
IAM의 복원력에 대한 업데이트	이벤트로 인해 AWS 리전 간 통신이 중단될 때 IAM 자격 증명에 대한 액세스를 유지하는 방법에 대한 정보가 추가되었습니다.	2022년 5월 16일
리소스에 대한 새로운 전역 조건 키	이제 리소스가 포함된 AWS Organizations의 계정, 조직 단위(OU) 또는 조직을 기반으로 리소스에 대한 액세스를 제어할 수 있습니다. IAM 정책에서 `aws:ResourceAccount` , `aws:ResourceOrgID` 및 `aws:ResourceOrgPaths` 전역 조건 키를 사용할 수 있습니다.	2022년 4월 27일
AWS SDK를 사용한 IAM용 코드 예제	IAM을 AWS 소프트웨어 개발 키트(SDK)와 함께 사용하는 방법을 보여주는 코드 예제가 추가되었습니다. 예제는 개별 서비스 함수를 호출하는 방법을 보여주는 코드 발췌문과 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 예제로 나뉩니다.	2022년 4월 7일
정책 평가 논리 순서도 업데이트	계정 내에서 요청이 허용되거나 거부되는지 결정 섹션의 정책 평가 논리 순서도와 관련 텍스트에 대한 업데이트입니다.	2021년 11월 17일

보안 모범 사례 업데이트	루트 사용자 보안 인증을 사용하는 대신 관리 사용자를 생성하는 방법에 대한 정보를 추가하고, 사용자 그룹을 사용하여 IAM 사용자에게 권한을 할당하는 모범 사례를 제거했으며, 인라인 정책 대신 관리형 정책을 사용해야 하는 시기를 명확히 했습니다.	2021년 10월 5일
리소스 기반 정책에 대한 정책 평가 논리 주제 업데이트	리소스 기반 정책 및 여러 보안 주체 유형이 동일한 계정에 미치는 영향에 대한 정보가 추가되었습니다.	2021년 10월 5일
단일 값 및 다중 값 조건 키 업데이트	단일 값 및 다중 값 조건 키의 차이점에 대해 자세히 설명합니다. 값 유형이 각 AWS 전역 조건 컨텍스트 키에 추가되었습니다.	2021년 9월 30일
IAM Access Analyzer가 Amazon S3 다중 리전 액세스 포인트 지원	IAM Access Analyzer는 Amazon S3 다중 리전 액세스 포인트를 사용하는 버킷을 포함하여 퍼블릭 및 크로스 계정 액세스를 허용하는 Amazon S3 버킷을 식별합니다.	2021년 9월 2일
AWS 관리형 정책 업데이트 - 기존 정책에 대한 업데이트	IAM Access Analyzer가 기존 AWS 관리형 정책을 업데이트했습니다.	2021년 9월 2일
작업 수준 정책 생성에 대해 추가 서비스 지원	IAM Access Analyzer는 추가로 AWS 서비스에 대한 작업 수준 액세스 활동 정보가 포함된 IAM 정책을 생성합니다.	2021년 8월 24일

크로스 계정 추적을 위한 IAM 정책 생성	이제 IAM Access Analyzer를 사용하여 다른 계정에서의 AWS CloudTrail 추적(예: 중앙 집중식 AWS Organizations 추적)을 사용하는 액세스 활동을 기반으로 세분화된 정책을 생성할 수 있습니다.	2021년 8월 18일

추가 IAM Access Analyzer 정책 확인	IAM Access Analyzer는 IAM 정책에 포함된 조건을 검증하는 새로운 정책 검사를 추가하여 정책 검증을 확장했습니다. 이러한 검사는 정책 설명서의 조건 블록을 분석하고 실행 가능한 권장 사항과 함께 보안 경고, 오류 및 제안 사항을 보고합니다. IAM Access Analyzer에 다음과 같은 정책 검사가 추가되었습니다. • 오류 - 잘못된 서비스 보안 주체 • 오류 - 조건에 태그 키 누락 • 보안 경고 - 서비스에 대해 지원되지 않는 태그 조건 키가 있는 NotAction 거부 • 보안 경고 - 서비스에 대해 지원되지 않는 태그 조건 키가 있어 거부 • 보안 경고 - 쌍으로 연결된 조건 키 누락 • 제안 - 서비스에 대해 지원되지 않는 태그 조건 키가 있는 NotAction 허용 • 제안 - 서비스에 대해 지원되지 않는 태그 조건 키 허용	2021년 6월 29일

마지막으로 액세스한 작업에 대해 추가 서비스 지원	이제 IAM 콘솔에서 IAM 보안 주체가 Amazon EC2, IAM, Lambda 및 Amazon S3 관리 작업과 같은 서비스에 대한 작업을 마지막으로 사용한 시간에 대한 마지막 액세스 작업 정보를 볼 수 있습니다. AWS CLI 또는 AWS API를 사용하여 데이터 보고서를 검색할 수도 있습니다. 이 정보를 사용하여 불필요한 권한을 확인할 수 있으므로 IAM 정책을 미세 조정함으로써 최소 권한의 원칙을 보다 잘 준수할 수 있습니다.	2021년 4월 19일
위임된 역할로 수행된 작업 모니터링 및 제어	관리자는 자격 증명이 AWS CloudTrail에 로그인된 소스 자격 증명을 전달하도록 IAM 역할을 구성할 수 있습니다. 소스 자격 증명 정보를 검토하면 관리자가 수임된 역할 세션에서 누가 또는 어떤 작업을 수행할지 결정할 수 있습니다.	2021년 4월 13일
액세스 활동을 기반으로 IAM 정책 생성	이제 IAM Access Analyzer를 사용하여 AWS CloudTrail에서 확인된 액세스 활동을 기반으로 세분화된 정책을 생성할 수 있습니다.	2021년 4월 7일
IAM Access Analyzer 정책 확인	IAM Access Analyzer는 이제 정책 작성 중에 실행 가능한 권장 사항과 함께 100개 이상의 정책 검사를 제공합니다.	2021년 3월 16일

확장된 정책 검증 옵션	IAM 콘솔에서 사용할 수 있는 확장된 정책 검증, AWS API 및 IAM Access Analyzer의 정책 검사를 사용하는 AWS CLI으로 안전하고 기능적인 JSON 정책을 작성할 수 있습니다.	2021년 3월 15일
IAM 리소스에 태그 지정	이제 태그 키 값 페어를 사용하여 추가 IAM 리소스에 태그를 지정할 수 있습니다.	2021년 2월 11일
IAM 사용자에 대한 기본 암호 정책	AWS 계정에 대한 사용자 지정 암호 정책을 설정하지 않은 경우 이제 IAM 사용자 암호는 기본 AWS 암호 정책을 충족해야 합니다.	2020년 11월 18일
AWS 서비스에 대한 작업, 리소스 및 조건 키 페이지 이동	각 AWS 서비스는 IAM 정책에서 사용할 수 있는 작업, 리소스 및 조건 컨텍스트 키를 정의할 수 있습니다. 이제 서비스 권한 부여 참조에서 AWS 서비스 및 해당하는 작업, 리소스와 조건 컨텍스트 키의 목록을 찾을 수 있습니다.	2020년 11월 16일
IAM 사용자의 더 긴 역할 세션 지속 시간	IAM 사용자는 AWS Management Console에서 역할을 전환할 때 더 긴 역할 세션 지속 시간을 가질 수 있으므로 세션 만료로 인한 중단을 줄일 수 있습니다. 역할에 대해 설정된 최대 세션 지속 시간 또는 IAM 사용자 세션의 남은 시간 중 더 적은 시간이 사용자에게 부여됩니다.	2020년 7월 24일

Service Quotas를 사용하여 IAM 엔터티에 대한 빠른 증가 요청	Service Quotas 콘솔을 사용하여 조정 가능한 IAM 할당량에 대한 할당량 증가를 요청할 수 있습니다. 이제 일부 증가는 Service Quotas에서 자동으로 승인되어 몇 분 내에 계정에서 사용할 수 있습니다. 더 큰 요청이 AWS Support에 제출됩니다.	2020년 6월 25일
이제 IAM에서 마지막으로 액세스한 정보에 Amazon S3 관리 작업 포함	이제는 서비스의 마지막 액세스 정보 외에도 IAM 보안 주체가 Amazon S3 작업을 마지막으로 사용한 시간에 대한 정보를 IAM 콘솔에서 볼 수 있습니다. AWS CLI 또는 AWS API를 사용하여 데이터 보고서를 검색할 수도 있습니다. 이 보고서에는 보안 주체가 마지막으로 액세스하려고 시도한 허용된 서비스 및 작업과 그 시기에 대한 정보가 포함됩니다. 이 정보를 사용하여 불필요한 권한을 확인할 수 있으므로 IAM 정책을 미세 조정함으로써 최소 권한의 원칙을 보다 잘 준수할 수 있습니다.	2020년 6월 3일
보안 장 추가	보안 장은 보안 및 규정 준수 목표에 맞게 IAM 및 AWS STS를 구성하고 충족하는 방법을 이해하는 데 도움이 됩니다. 또한 IAM 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.	2020년 4월 29일

sts:RoleSessionName	이제 보안 주체가 역할을 수임할 때 지정한 세션 이름을 기반으로 권한을 부여하는 정책을 작성할 수 있습니다.	2020년 4월 21일
AWS 로그인 페이지 업데이트	기본 AWS 로그인 페이지에서 로그인할 때 AWS 계정 루트 사용자 또는 IAM 사용자로 로그인하도록 선택할 수 없습니다. 이렇게 하면 페이지의 레이블에 루트 사용자 이메일 주소 또는 IAM 사용자 정보를 제공해야 한다고 표시됩니다. 이 설명서에는 AWS 로그인 페이지를 이해하는 데 도움이 되는 업데이트된 화면 캡처가 포함되어 있습니다.	2020년 3월 4일
aws:ViaAWSService 및 aws:CalledVia 조건 키	이제 서비스가 IAM 보안 주체(사용자 또는 역할)를 대신하여 요청을 수행할 수 있는지 여부를 제한하는 정책을 작성할 수 있습니다. 보안 주체가 AWS 서비스에 요청을 하면 해당 서비스는 보안 주체의 자격 증명을 사용하여 다른 서비스에 대한 후속 요청을 수행할 수 있습니다. 서비스가 보안 주체의 자격 증명을 사용하여 요청하는 경우, `aws:ViaAWSService` 조건 키를 사용하여 일치시킵니다. 특정 서비스가 보안 주체의 자격 증명을 사용하여 요청하는 경우 `aws:CalledVia` 조건 키를 사용하여 일치시킵니다.	2020년 2월 20일

정책 시뮬레이터에서 권한 경계에 대한 지원 추가	이제 IAM 정책 시뮬레이터를 사용하여 IAM 엔터티에 대한 권한 경계의 효과를 테스트할 수 있습니다.	2020년 1월 23일
크로스 계정 정책 평가	이제 AWS에서 크로스 계정 액세스에 대한 정책을 평가하는 방법을 알아볼 수 있습니다. 교차 계정 액세스는 신뢰하는 계정의 리소스에 다른 계정의 보안 주체가 리소스에 액세스할 수 있도록 허용하는 리소스 기반 정책이 포함된 경우 발생합니다. 두 계정 모두에서 요청이 허용되어야 합니다.	2020년 1월 2일
세션 태그	이제 AWS STS에서 역할을 수임하거나 사용자를 연동할 때 태그를 포함할 수 있습니다. `AssumeRole` 또는 `GetFederationToken` 작업을 수행할 때 세션 태그를 속성으로 전달할 수 있습니다. `AssumeRoleWithSAML` 또는 `AssumeRoleWithWebIdentity` 작업을 수행할 때 회사 자격 증명의 속성을 AWS로 전달할 수 있습니다.	2019년 11월 22일

AWS Organizations의 AWS 계정 그룹에 대한 액세스 제어	이제 IAM 정책의 AWS Organizations에서 조직 단위(OU)를 참조할 수 있습니다. Organizations를 사용하여 계정을 OU로 구성하는 경우 리소스에 대한 액세스 권한을 부여하기 전에 보안 주체가 특정 OU에 속하도록 요구할 수 있습니다. 보안 주체에는 AWS 계정 루트 사용자, IAM 사용자, IAM 역할이 포함됩니다. 이렇게 하려면 정책의 `aws:PrincipalOrgPaths` 조건 키에 OU 경로를 지정합니다.	2019년 11월 20일
마지막으로 사용된 역할	이제 역할이 마지막으로 사용된 날짜, 시간 및 리전을 볼 수 있습니다. 이 정보는 또한 계정에서 사용되지 않은 역할을 식별하는 데 도움이 됩니다. AWS Management Console, AWS CLI 및 AWS API를 사용하여 역할이 마지막으로 사용된 시기에 대한 정보를 볼 수 있습니다.	2019년 11월 19일

전역 조건 컨텍스트 키 페이지 업데이트	이제 각 전역 조건 키가 요청 컨텍스트에 포함되는 시기를 알 수 있습니다. 또한 페이지 TOC(목차)를 사용하여 각 키를 보다 쉽게 탐색할 수 있습니다. 이 페이지의 정보는 보다 정확한 정책을 작성하는 데 도움이 됩니다. 예를 들어 직원이 IAM 역할과의 연동을 사용하는 경우 `aws:userId` 키가 아닌 `aws:userName` 키를 사용해야 합니다. `aws:userName` 키는 IAM 사용자에게만 적용되며 역할에는 적용되지 않습니다.	2019년 10월 6일
AWS의 ABAC	태그를 사용하여 속성 기반 액세스 제어(ABAC)가 AWS에서 작동하는 방식 및 이러한 방식을 기존 AWS 권한 부여 모델과 비교하는 방식을 알아봅니다. ABAC 튜토리얼을 사용하여 보안 주체 태그가 있는 IAM 역할이 태그가 일치하는 리소스에 액세스할 수 있도록 허용하는 정책을 생성하고 테스트하는 방법을 알아봅니다. 이 전략을 통해 개인이 자신의 작업에 필요한 AWS 리소스만 보거나 편집할 수 있습니다.	2019년 10월 3일

AWS STS GetAccessKeyInfo 작업	코드에서 AWS 액세스 키를 살펴보면 키가 자신의 계정에 속한 것인지 알 수 있습니다. 액세스 키 ID는 `aws sts get-access-key-info` AWS CLI 명령 또는 `GetAccessKeyInfo` AWS API 작업을 사용해 전달할 수 있습니다.	2019년 7월 24일
IAM에서 마지막으로 Organizations 서비스에 액세스한 정보 보기	이제 IAM 콘솔의 AWS Organizations 섹션에서 AWS Organizations 엔터티 또는 정책에 대해 마지막으로 액세스한 서비스 정보를 볼 수 있습니다. AWS CLI 또는 AWS API를 사용하여 데이터 보고서를 검색할 수도 있습니다. 이 데이터에는 Organizations 계정의 보안 주체가 마지막으로 액세스를 시도한 허용된 서비스와 그 시기에 대한 정보가 있습니다. 이 정보를 사용하여 불필요한 권한을 확인할 수 있으므로 사용자의 정책 또는 Organizations 정책을 미세 조정함으로써 최소 권한의 원칙을 보다 잘 준수할 수 있습니다.	2019년 6월 20일
관리형 정책을 세션 정책으로 사용	역할을 수임할 때 최대 10개의 관리형 정책 ARN을 전달할 수 있습니다. 이를 통해 역할의 임시 자격 증명에 대한 권한을 제한할 수 있습니다.	2019년 5월 7일

전역 엔드포인트에 대한 세션 토큰의 AWS STS 리전 호환성	이제 전역 엔드포인트 토큰의 버전 1 또는 버전 2 사용 여부를 선택할 수 있습니다. 버전 1 토큰은 기본적으로 이용 가능한 AWS 리전에서만 유효합니다. 이러한 토큰은 아시아 태평양(홍콩)과 같이 수동으로 활성화된 리전에서 작동하지 않습니다. 버전 2는 모든 리전에서 유효합니다. 하지만 버전 2 토큰은 더 길고 일시적으로 토큰을 저장하는 시스템에 영향을 미칠 수 있습니다.	2019년 4월 26일
AWS 리전 활성화 및 비활성화 허용	이제 관리자가 아시아 태평양(홍콩) 리전(ap-east-1)을 활성화 및 비활성화할 수 있도록 허용하는 정책을 생성할 수 있습니다.	2019년 4월 24일
IAM 사용자 내 보안 자격 증명 페이지	이제 IAM 사용자는 내 보안 자격 증명 페이지에서 자신의 모든 자격 증명을 관리할 수 있습니다. 이 AWS Management Console 페이지에는 계정 ID 및 정식 사용자 ID와 같은 계정 정보가 표시됩니다. 또한 사용자는 자신의 암호, 액세스 키, X.509 인증서, SSH 키, Git 자격 증명을 보고 편집할 수 있습니다.	2019년 1월 24일
액세스 관리자 API	이제 AWS CLI 및 AWS API를 사용하여 마지막으로 액세스한 서비스 정보를 볼 수 있습니다.	2018년 12월 7일

IAM 사용자 및 역할 태그 지정	이제 IAM 태그를 사용하여 태그 키 값 페어를 통해 사용자 지정 속성을 자격 증명(IAM 사용자 또는 역할)에 추가할 수 있습니다. 태그를 사용하여 리소스에 대한 자격 증명의 액세스를 제어하거나 자격 증명에 연결할 수 있는 태그를 제어할 수도 있습니다.	2018년 11월 14일
U2F 보안 키	이제 AWS Management Console에 로그인할 때 멀티 팩터 인증(MFA) 옵션으로 U2F 보안 키를 사용할 수 있습니다.	2018년 9월 25일
Amazon VPC 엔드포인트에 대한 지원	이제 미국 서부(오레곤) 리전에서 VPC와 AWS STS 간에 프라이빗 연결을 설정할 수 있습니다.	2018년 7월 31일
권한 경계	새 기능을 사용하면 IAM 권한을 관리할 수 있는 권한을 신뢰할 수 있는 직원에게 부여하는 작업이 더 간편해질 뿐 아니라 IAM 관리자 액세스 권한 전체를 부여하지 않아도 됩니다.	2018년 7월 12일
aws:PrincipalOrgID	새로운 조건 키는 IAM 보안 주체의 AWS 조직을 지정하여 AWS 리소스에 대한 액세스를 제어하는 쉬운 방법을 제공합니다.	2018년 5월 17일
aws:RequestedRegion	새로운 조건 키는 IAM 정책을 사용하여 AWS 리전에 대한 액세스를 제어하는 쉬운 방법을 제공합니다.	2018년 25월 4일

IAM 역할에 대한 세션 기간 증가	이제 IAM 역할은 12시간의 세션 기간을 가질 수 있습니다.	2018년 3월 28일
역할 생성 워크플로 업데이트	새로운 워크플로우는 신뢰 관계를 생성하고 권한을 역할에 연결하는 과정을 개선합니다.	2017년 9월 8일
AWS 계정 로그인 프로세스	업데이트된 AWS 로그인 환경을 통해 루트 사용자와 IAM 사용자 모두 AWS Management Console의 홈 페이지에서 Sign In to the Console(콘솔에 로그인) 링크를 사용할 수 있습니다.	2017년 8월 25일
예제 IAM 정책	30 가지의 예제 정책이 넘는 설명서 업데이트 기능.	2017년 8월 2일
IAM 모범 사례	IAM 콘솔의 사용자 섹션에 추가된 정보는 IAM 모범 사례를 쉽게 따라할 수 있게 만듭니다.	2017년 7월 5일
Auto Scaling 리소스	리소스 수준 권한은 Auto Scaling 리소스로의 액세스 및 권한을 제어할 수 있습니다.	2017년 5월 16일
Amazon RDS for MySQL 및 Amazon Aurora 데이터베이스	데이터베이스 관리자는 데이터베이스 사용자를 IAM 사용자 및 역할과 연관시킬 수 있어 사용자가 단일 위치에서 모든 AWS 리소스에 대한 사용자 액세스를 관리할 수 있습니다.	2017년 4월 24일
서비스 연결 역할	서비스 링크된 역할은 AWS 서비스로 권한을 위임하는 더욱 쉽고 보안한 방법을 제공합니다.	2017년 4월 19일

정책 요약	새로운 정책 요약을 통해 IAM 정책의 권한을 더욱 손쉽게 이해할 수 있습니다.	2017년 3월 23일